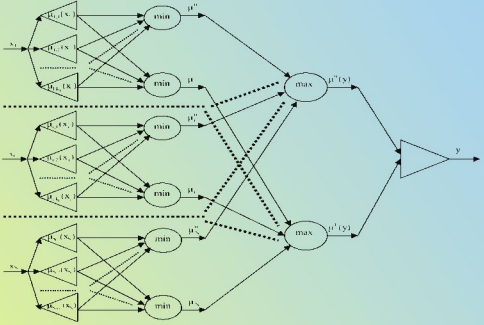
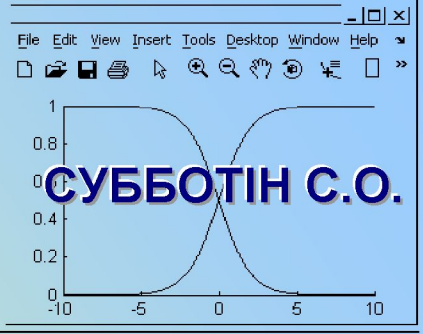
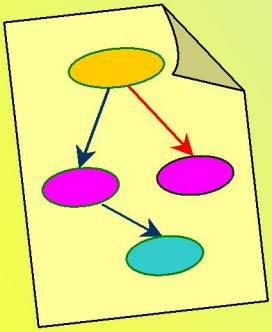


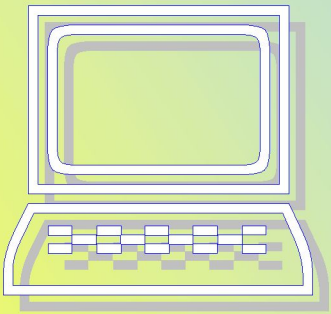


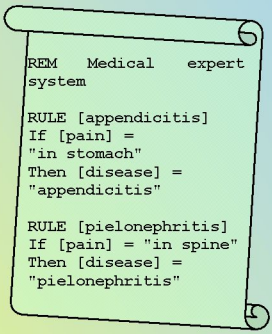
СУББОТІН С.О.

ПОДАННЯ Й ОБРОБКА ЗНАНЬ У СИСТЕМАХ ШТУЧНОГО ІНТЕЛЕКТУ ТА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

навчальний посібник







Запорізький
національний технічний
університет

2008

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Запорізький національний технічний університет

С. О. Субботін

ПОДАННЯ Й ОБРОБКА ЗНАНЬ У СИСТЕМАХ ШТУЧНОГО ІНТЕЛЕКТУ ТА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

Навчальний посібник

Рекомендовано Міністерством освіти і науки України як навчальний посібник для студентів вищих навчальних закладів, які навчаються за напрямом «Програмна інженерія»

Видання здійснено за підтримки Міжнародного проекту «Європейсько-український ступінь магістра з програмної інженерії» (JEP 26182–2005) за програмою Tempus Tacis Європейської Комісії

Запоріжжя
2008

ББК 32.973
С89
УДК 004.93:681.32

Рекомендовано до друку Вченою радою
Запорізького національного технічного університету
(Протокол № 7 від 03.03.2008 р.)

Гриф надано Міністерством освіти і науки України
(лист № 1.4/18-Г-1852 від 16.07.2008 р.)

Рецензенти: доктор технічних наук, професор кафедри штучного інтелекту Харківського національного університету радіоелектроніки *Бодянський Є. В.*;
доктор технічних наук, декан математичного факультету Запорізького національного університету *Гоменюк С. І.*;
доктор технічних наук, професор кафедри технічної кібернетики Національного технічного університету України «Київський політехнічний інститут» *Стенін О. А.*

Субботін С. О.

С89 Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень: Навчальний посібник. – Запоріжжя: ЗНТУ, 2008. – 341 с.

ISBN 978–966–7809–87–4

Книга містить систематизований виклад математичних основ і методів опису, побудови та застосування моделей знань в системах штучного інтелекту та підтримки прийняття рішень. Розглянуто семантичні, фреймові та нейро-нечіткі мережі, продукційні та логічні моделі. Поряд із класичними методами та моделями запропоновано оригінальні авторські розробки, що дозволяють автоматизувати побудову блоків підтримки прийняття рішень інтелектуальних систем. Наведено опис програмних засобів, що реалізують розглянуті методи та моделі. Видання призначено для студентів комп'ютерних спеціальностей вищих навчальних закладів, а також може використовуватися аспірантами, науковими та педагогічними працівниками, практичними фахівцями.

ББК 32.973

ISBN 978–966–7809–87–4

ЗМІСТ

ВСТУП

Постійне збільшення обсягів наявної інформації робить майже неможливою її ручну обробку й освоєння людиною та висуває вимоги щодо створення комп'ютерних засобів для автоматизації подання, збереження, систематизації, пошуку та іншої обробки інформації. При цьому особлива потреба виникає у створенні систем, що здатні узагальнювати та аналізувати інформацію, адаптуватися до її змін, спілкуватися з людиною-користувачем природною мовою, приймати рішення в умовах неповної, ненадійної та суперечливої інформації. Такими системами є *інтелектуальні системи, засновані на знаннях*.

Актуальність створення, вивчення та використання методів, моделей та комп'ютерних систем подання й обробки знань для інтелектуальної підтримки прийняття рішень підтверджується Державною програмою розвитку промисловості на 2003–2011 роки, схваленою Постановою Кабінету Міністрів України № 1174 від 28. 07. 2003 та Указом Президента України №102/2003 від 12. 02. 2003 «Про Концепцію державної промислової політики», що визначають як пріоритет інформатизації створення систем підтримки прийняття рішень та штучного інтелекту; Постановою Кабінету Міністрів України №1896 від 10. 12. 2003, яка передбачає «… розроблення методології інтелектуального аналізу даних … на основі застосування сучасних методів нечіткої логіки, штучного інтелекту та добування знань із баз даних»; Постановою Кабінету Міністрів України №789 від 15 липня 1997 р. «Про першочергові заходи інформатизації», Законом України №75/98–ВР від 04. 02. 1998 «Про Концепцію Національної програми інформатизації», Законом України №76/98–ВР від 04. 02. 1998 «Про затвердження Завдань Національної програми інформатизації на 1998–2000 роки», Постановою Верховної Ради України № 914–XIV від 13. 07. 1999 «Про затвердження Завдань Національної програми інформатизації на 1999–2001 роки», які передбачають, зокрема, «створити діючі зразки та прототипи конкурентоспроможних засобів та систем: методичне та програмне забезпечення проектування і розроблення комп'ютеризованих систем для застосування в управлінні, програмно-технічні засоби підтримки експертного прийняття рішень, високопродуктивні оброблювачі інформації з нейромережевою архітектурою, проблемно-орієнтовані робочі станції, нейрокомп'ютери та нейромережеві технології, програмне забезпечення інформаційно-аналітичної обробки текстових, фактографічних та статистичних даних, конкурентоспроможні інформаційні технології формування аналітичних електронних оглядів і довідок, засоби інтелектуалізації широкого застосування».

Метою даної книги є систематизований виклад основних понять, методів та моделей, що дозволяють створювати основані на знаннях інтелектуальні системи підтримки прийняття рішень у різних сферах людської діяльності.

Поряд із описом відомих методів та моделей розглянуто оригінальні авторські розробки, що дозволяють автоматизувати побудову блоків підтримки прийняття рішень інтелектуальних систем. У книзі також наведено опис програмних засобів, що реалізують розглянуті методи та моделі.

Видання орієнтоване на студентів комп'ютерних спеціальностей вищих навчальних закладів, а також може використовуватися аспірантами, науковими та педагогічними працівниками, практичними фахівцями.

Матеріал, наведений у книзі, призначений для вивчення та апробований автором при читанні курсів «Математичні основи подання знань», «Системи штучного інтелекту», «Інтелектуальні агенти», «Сучасний штучний інтелект», «Сучасні бази даних та інтелектуальний аналіз даних», «Організація баз даних і знань» у Запорізькому національному технічному університеті, а також курсів «Експертні системи» та «Системи штучного інтелекту» у Класичному приватному університеті (Гуманітарному університеті «Запорізький інститут державного та муніципального управління»). Книга також може використовуватися при вивченні окремих розділів дисциплін «Дискретні структури», «Програмне забезпечення інтелектуальних систем», «Лінгвістичне забезпечення інтелектуальних систем», «Нейроінформатика та еволюційні алгоритми», «Теорія прийняття рішень», «Логічне програмування» та інших.

Більш детальна інформація про використання матеріалу книги у навчальному процесі, а також електронні варіанти праць автора та інші корисні посилання на літературні джерела та рекомендовані програмні засоби доступні на веб-сайті автора за адресою: `http://csit.narod.ru`.

Терміни та назви методів, визначення яких наводиться у наступному тексті виділено *курсивом*. Назви підзаголовків всередині підрозділів виділено ***жирним курсивом***. Тексти програм виділено `моноширинним шрифтом`.

Для спрощення пошуку навчальних завдань, прикладів і літературних джерел за кожною темою використовується така *система умовних позначень*:

 – контрольні питання,

 – приклади та додаткові ілюстрації до розділів,

 – завдання, що мають бути виконані вручну,

 – завдання, що мають бути виконані з використанням комп'ютера,

 – завдання для підготовки рефератів та доповідей,

 – посилання на літературні джерела за темою розділу.

1. ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ, ЗАСНОВАНІ НА ЗНАННЯХ

1.1 Інтелектуальні системи

Штучний інтелект – це галузь комп'ютерних наук, що займається автоматизацією розумного поводження агентів, які одержують у результаті актів сприйняття інформацію про навколишнє середовище і виконують дії, що реалізують функцію від результатів сприйняття і попередніх дій.

Метою штучного інтелекту є розробка комп'ютерних систем, що мають можливості, які традиційно пов'язуються з людським розумом: розуміння мови, навчання, здатність міркувати, вирішувати проблеми, планувати і т. д.

Основні напрями досліджень в галузі штучного інтелекту виділяють такі:

– *Подання знань і робота з ними* – створення спеціалізованих моделей і мов для подання знань в ЕОМ, а також програмних і апаратних засобів для їхнього перетворення (поповнення, логічної обробки і т. д.).

– *Планування доцільного поводження* – дослідження зі створення методів формування цілей і вирішення задач планування дій агента, що функціонує в складному зовнішньому середовищі.

– *Спілкування людини з ЕОМ* – задачі створення мовних засобів, що дозволяють ефективно взаємодіяти з ЕОМ непрограмуючому користувачу. Ведуться дослідження в області синтаксису і семантики природних мов, способів збереження знань про мову в пам'яті машини і побудови спеціальних процесорів, що здійснюють переклад текстової інформації у внутрішнє машинне подання.

– *Розпізнавання образів і навчання* – дослідження зі сприйняття зорової, слухової і інших видів інформації, методів її обробки, формування відповідних реакцій на впливи зовнішнього середовища і способів адаптації штучних систем до середовища шляхом навчання.

Інтелектуальною системою називають кібернетичну систему, призначену для вирішення інтелектуальних задач.

Інтелектуальна задача – це задача, точний алгоритмізований метод вирішення якої апріорі є невідомий. При цьому під *рішенням задачі* розуміється будь-яка діяльність, пов'язана з розробкою планів і виконанням дій, необхідних для досягнення визначеної мети.

Властивість інтелектуальності, що істотно відрізняє інтелектуальні системи від інших кібернетичних систем, характеризується набором таких ознак.

1. Наявність у системі власної внутрішньої моделі зовнішнього світу, що забезпечує індивідуальність, самостійність системи в оцінці вхідного запиту, можливість значеннєвої (семантичної) і прагматичної інтерпретації запиту відповідно до власних знань і вироблення відповіді (реакції), семантично і прагматично правильної з точністю до адекватного моделювання зовнішнього світу (предметної області).

2. Здатність системи поповнювати наявні знання, засвоювати нові знання, навчатися, здійснюючи вбудовування нової інформації в систему подання знань.

3. Здатність системи виділяти значні якісні характеристики ситуації.

4. Здатність до дедуктивного виведення, генерації рішення, що у явному і готовому вигляді не міститься в самій системі.

5. Здатність до прийняття рішень на основі нечіткої, неточної, недостатньої або погано визначеної інформації і застосування формалізмів подань, що допомагають програмісту справлятися з цими недоліками.

6. Еволюційність і адаптивність моделей штучного інтелекту: набуття знань системою здійснюється за допомогою навчання, заснованого на адаптації (пристосуванні) її структури і параметрів у процесі еволюційного розвитку до умов зовнішнього середовища, що змінюються.

У залежності від *сфери застосування* інтелектуальні системи можна поділити на системи загального призначення і спеціалізовані системи.

Інтелектуальні системи загального призначення – системи, що не тільки виконують задані процедури, але на основі метапроцедур пошуку генерують і виконують процедури рішення нових конкретних задач. Технологія використання таких систем полягає в наступному. Користувач (експерт) формує знання (дані і правила), що описують обране застосування. Потім на підставі цих знань, заданої мети і вихідних даних метапроцедури системи генерують і виконують процедуру вирішення конкретної задачі. Дану технологію називають *технологією систем, заснованих на знаннях*, або *технологією інженерії знань*. Вона дозволяє фахівцю, що не знає програмування, розробляти гнучкі прикладні системи. Найбільше широко використовуваним типом інтелектуальних систем загального призначення є оболонки експертних систем.

Спеціалізовані інтелектуальні системи – вирішують фіксований набір задач, визначений при проектуванні системи. Для використання таких систем потрібно наповнити їх даними, що відповідають обраному застосуванню.

1.2 Проблемні області та їхні властивості

Проблемна область (предметна область) – сукупність взаємозалежних відомостей, необхідних і достатніх для вирішення даної інтелектуальної задачі. Знання про предметну область включають описи об'єктів, явищ, фактів, подій, а також відношень між ними.

Уявно предметна область складається з реальних або абстрактних об'єктів, що звуться *сутностями*. Сутності предметної області знаходяться у визначених *відношеннях* (асоціаціях) одна до одної, які також можна розглядати як сутності і включати в предметну область. Між сутностями спостерігаються різні *відношення подоби*. Сукупність подібних сутностей складає *клас сутностей*, що є новою сутністю предметної області.

Для вирішення інтелектуальних задач необхідно використовувати знання з конкретної предметної області, подані в певній стандартній формі і скласти програму їхньої обробки.

Класифікація проблемних середовищ може бути зроблена за такими вимірами.

– *Середовище, яке повністю* або *частково спостерігається.*

Повністю спостережне – середовище, у якому датчики агента надають йому доступ до повної інформації про стан середовища в кожний момент часу, необхідної для вибору агентом дії.

Частково спостережне – середовище, у якому через датчики, що створюють шум і є неточними, або через те, що окремі характеристики його стану просто відсутні в інформації, отриманій від датчиків, агент не може мати доступ до повної інформації про стан середовища в кожний момент часу.

– *Детерміноване* або *стохастичне середовище.*

Детерміноване – середовище, наступний стан якого цілком визначається поточним станом і дією, виконаною агентом.

Стохастичне – середовище, наступний стан якого цілком не визначається поточним станом і дією, виконаною агентом.

– *Епізодичне* або *послідовне середовище*.

В *епізодичному середовищі* досвід агента складається з нерозривних епізодів, кожний з яких містить у собі сприйняття середовища агентом, а потім виконання однієї дії, при цьому наступний епізод не залежить від дій, виконаних у попередніх епізодах.

У *послідовному середовищі* поточне рішення може вплинути на всі майбутні рішення.

– *Статичне* або *динамічне середовище.*

Динамічне середовище для агента – середовище, що може змінитися в ході того, як агент обирає чергову дію.

Статичне середовище для агента – середовище, що не може змінитися в ході того, як агент обирає чергову дію. Статичність області означає незмінність вихідних даних, що її описують. При цьому похідні дані (виведені з вихідних) можуть і з'являтися заново, і змінюватися (не змінюючи, однак, вихідних даних).

Напівдинамічне середовище для агента – середовище, що з часом не змінюються, а змінюються тільки показники продуктивності агента в середовищі.

– *Дискретне* або *неперервне середовище*. Розходження між дискретними і неперервними варіантами середовища може відноситися до стану середовища, способу обліку часу, а також сприйняттям і діям агента.

– *Одноагентне середовище* (коли в середовищі діє тільки один агент) або *мультиагентне середовище* (коли діють два агенти і більше).

Крім того, предметні області можна характеризувати такими аспектами: числом і складністю сутностей; їхніх атрибутів і значень атрибутів; зв'язністю сутностей та їхніх атрибутів; повнотою знань; точністю знань (знання точні або правдоподібні; правдоподібність знань подається певним числом або висловленням).

Задачі, що вирішуються інтелектуальними системами у проблемній області, класифікують:

– *за ступенем зв'язності правил*: *зв'язні* (задачі, що не вдається розбити на незалежні задачі) та *малозв'язні* (задачі, що вдається розбити на деяку кількість незалежних підзадач);

– *з точки зору розробника*: *статичні* (якщо процес вирішення задачі не змінює вихідні дані про поточний стан предметної області) і *динамічні* (якщо процес вирішення задачі змінює вихідні дані про поточний стан предметної області);

– *за класом вирішуваних задач*: розширення, довизначення, перетворення.

1. *Задачі розширення* – задачі, у процесі вирішення яких здійснюється тільки збільшення інформації про предметну область, що не призводить ні до зміни раніше виведених даних, ані до вибору іншого стану області. Типовою задачею цього класу є задача класифікації.

2. *Задачі довизначення* – задачі з неповною або неточною інформацією про реальну предметну область, мета вирішення яких – вибір з множини альтернативних поточних станів предметної області того, що є адекватний вихідним даним. У випадку неточних даних альтернативні поточні стани виникають як результат ненадійності даних і правил, що призводить до різноманіття різних доступних висновків з тих самих вихідних даних. У випадку неповних даних альтернативні стани є результатом довизначення області, тобто результатом припущень про можливі значення відсутніх даних.

3. *Задачі перетворення* – задачі, які здійснюють зміни вихідної або виведеної раніше інформації про предметну область, що є наслідком змін або реального світу, або його моделі.

Задачі розширення і довизначення є статичними, а задачі перетворення – динамічними.

1.3 Системи, засновані на знаннях

При використанні традиційних структурних мов програмування необхідні знання вміщувалися безпосередньо в прикладну програму й утворювали з нею одне ціле. Однак такий підхід ускладнює розуміння того, яким чином використовуються знання і яку роль вони виконують, тобто знання, закладені в програму, і сама програма їхньої обробки виявляються жорстко пов'язаними між собою і дозволяють одержувати тільки ті висновки з наявних знань, що передбачені програмою їхньої обробки. Для вирішення цієї проблеми використовують системи, засновані на знаннях.

Система, заснована на знаннях – система програмного забезпечення, основними структурними елементами якої є *бази знань* і *механізм (машина) логічних виведень*. У системи, засновані на знаннях, також входять як модулі: підсистема набуття знань, підсистема пояснень та інтерфейсна підсистема.

У системі, заснованій на знаннях, знання подаються в конкретній формі в базі знань, що дозволяє їх легко визначати, модифікувати і поповнювати; функції вирішення задач реалізуються автономним механізмом логічних виведень, що робляться на знаннях, які зберігаються в базі. Саме вибір методів подання й одержання знань визначає архітектуру системи знань і на практиці виражається у відповідній організації бази знань і схеми керування машиною виведення.

Невід'ємними характеристиками систем, заснованих на знаннях, із практичної точки зору вважаються їхня здатність пояснювати лінію суджень та можливість набуття і нарощування знань.

Система, заснована на знаннях, може бути описана такою *ієрархією рівнів* (у порядку зменшення рівнів ієрархії).

1. *Рівень знань* (knowledge level) – пов'язаний зі змістом інформації, а також способами її використання і визначає можливості інтелектуальної системи. Самі знання не залежать від формалізмів, використовуваних для їхнього подання, а також виразності обраної мови програмування. На рівні знань зважуються питання про те, які запити є припустимими в системі, які об'єкти і відношення відіграють важливу роль у даній предметній області, як додати в систему нові знання, чи будуть факти згодом змінюватися, як у системі будуть реалізовані розсуди про знання, чи має дана предметна область добре зрозумілу систематику, чи є в ній незрозуміла або неповна інформація.

2. *Рівень символів* (symbol level) – пов'язаний з конкретними формалізмами, застосовуваними для подання знань у процесі вирішення задач. На цьому рівні здійснюється вибір конкретного способу подання знань і визначається мова подання для бази знань, зокрема, логічні або продукційні правила. Відділення рівня символів від рівня знань дозволяє програмісту вирішувати проблеми виразності, ефективності і простоти програмування, що не відносяться до більш високих рівнів поводження системи.

3. *Рівень алгоритмів і структур даних* – визначає структури даних для подання знань і алгоритми їхньої обробки.

4. *Рівень мов програмування* – визначає використовуваний стиль програмування. Хоча гарний стиль програмування припускає поділ конкретних властивостей мови програмування і вищестоящих рівнів, специфіка задач штучного інтелекту вимагає їхнього глибокого взаємозв'язку.

5. *Рівень компонування* – визначає архітектуру і функціональність операційної системи.

6. *Рівень апаратних засобів* – визначає архітектуру апаратних засобів, обсяг пам'яті і швидкодію процесора. Багаторівневий підхід дозволяє програмісту відволіктися від складності, що відноситься до нижніх рівнів, і сконцентрувати свої зусилля на питаннях, що відповідають даному рівню абстракції. Такий підхід дозволяє виділити теоретичні основи штучного інтелекту й абстрагуватися від деталей конкретної реалізації або мови програмування. Він дозволяє модифікувати реалізацію, підвищуючи її ефективність, або виконати портирування на іншу платформу, не торкаючись поводження системи на більш високих рівнях.

1.4 Експертні системи

Експертна система – це програмний засіб, що використовує експертні знання у певній предметній області з метою ефективного вирішення задач у предметній області, яка цікавить користувача, на рівні середнього професіонала (експерта).

Всі експертні системи є системами, заснованими на знаннях і програмами штучного інтелекту, але не навпаки. Інтелектуальні системи – найбільш загальний клас систем, які демонструють інтелектуальне поводження вмілим застосуванням евристик; системи, засновані на знаннях – підклас інтелектуальних систем, що роблять знання предметної області явними і відокремлюють їх від іншої частини системи; експертні системи – підклас систем, заснованих на знаннях, що застосовують експертні знання до складних задач реального життя.

Експертні системи використовуються для вирішення так званих *неформалізованих задач*, загальним для яких є те, що: задачі недостатньо добре розуміються або вивчені; задачі не можуть бути задані в числовій формі, але можуть бути досліджені за допомогою механізму символічних суджень; цілі не можна виразити в термінах точно визначеної цільової функції; не існує відомого алгоритмічного рішення задачі; якщо алгоритмічне рішення є, то його не можна використовувати через обмеженість ресурсів (час, пам'ять). Крім того неформалізовані задачі мають помилковість, неповноту, неоднозначність і суперечливість як вихідних даних, так і знань про розв'язувану задачу.

1.4.1 Властивості експертних систем

Властивості експертних систем, що відрізняють їх від звичайних програм:

– накопичення й організація знань про предметну область у процесі побудови й експлуатації експертної системи;

– явність і доступність знань;

– застосування для вирішення проблем високоякісного досвіду, що подає рівень мислення найбільш кваліфікованих експертів у даній області, який веде до творчих, точних і ефективних рішень;

– моделювання не стільки фізичної (чи іншої) природи визначеної проблемної області, скільки механізму мислення людини стосовно до вирішення задач у цій проблемній області;

– наявність прогностичних можливостей, за яких експертна система видає відповіді не тільки для конкретної ситуації, але і показує, як змінюються ці відповіді в нових ситуаціях, з можливістю докладного пояснення яким чином нова ситуація призвела до змін;

– забезпечення такої нової якості, як *інституціональна пам'ять*, за рахунок бази знань, що входить до складу експертної системи та розроблена в ході взаємодій з фахівцями організації, і являє собою поточну політику цієї

групи людей. Цей набір знань стає зведенням кваліфікованих думок і постійно поновлюваним довідником найкращих стратегій і методів, використовуваних персоналом. Провідні спеціалісти ідуть, але їхній досвід залишається;

– можливість використання для навчання і тренування керівників, забезпечуючи нових службовців великим багажем досвіду і стратегій, за якими можна вивчати політику, що рекомендується, і методи;

– явний поділ засобів керування і даних;

– слабка детермінованість керування;

– використання при вирішенні задач евристичних і наближених методів, які, на відміну від алгоритмічних, не завжди гарантують успіх. *Евристика* є *правилом впливу* (rule of thumb), що у машинному вигляді подає деяке знання, набуте людиною в міру накопичення практичного досвіду вирішення аналогічних проблем. Такі методи є приблизними в тому змісті, що, по-перше, вони не вимагають вичерпної вихідної інформації, і, по-друге, існує визначений ступінь упевненості (чи непевності) у тому, що пропоноване рішення є вірним;

– здатність до символьних суджень: здатність подавати знання в символьному вигляді і переформулювати символьні знання;

– прийняття рішень на основі правил і логічного виведення;

– організація способу керування ходом виконання з використанням машини логічного виведення;

– *самосвідомість* – здатність досліджувати свої судження (тобто перевіряти їхню правильність) і пояснювати свої дії;

– здатність до навчання на своїх помилках;

– можливість застосування неповні чи неправильні вхідні дані;

– *компетентність* – здатність досягати експертного рівня рішень (в конкретній предметній області мати той же рівень професіоналізму, що й експерти-люди), бути вмілою (застосовувати знання ефективно і швидко, уникаючи, як і люди, непотрібних обчислень), мати *адекватну робастність* (здатність лише поступово знижувати якість роботи у міру наближення до меж діапазону або компетентності припустимої надійності даних);

– *логічна адекватність* – здатність подання знань експертної системи розпізнавати усі відмінності, що закладаються у вихідні сутності;

– *евристична потужність* – наявність поряд з виразною мовою подання деякого засобу використання подань, сконструйованих і інтерпретованих таким чином, щоб з їхньою допомогою можна було вирішити проблему;

– *природність нотації* – зручність і простота виразів, якими формально описуються знання в експертній системі, зрозумілість їхнього змісту навіть тим, хто не знає, яким чином комп'ютер інтерпретує ці вирази;

– *логічна прозорість* – здатність експертної системи пояснити методику прийняття рішення, яка обумовлює те, наскільки просто персоналу з'ясувати, що робить програма і чому;

– *глибина* – здатність експертної системи працювати в предметній області, що містить важкі задачі, і використовувати складні правила (використовувати складні конструкції правил або велику їхню кількість);

– *корисність* – здатність експертної системи в ході діалогу визначати потреби користувача, виявляти й усувати причини невдач у роботі, а також вирішувати поставлені задачі;

– *гнучкість* – здатність системи налагоджуватися на різних користувачів, а також враховувати зміни в кваліфікації одного й того ж самого користувача;

– *зручність роботи* – природність взаємодії з експертною системою (спілкування в звичному виді, що втомлює користувача), її гнучкість і стійкість системи до помилок (здатність не виходити з ладу при помилкових діях недосвідченого користувача).

1.4.2 Класифікація експертних систем

Виділяють такі *види експертних систем*.

1. *За метою створення*: для навчання фахівців, для вирішення задач, для автоматизації рутинних робіт, для тиражування знань експертів.

2. *За основним користувачем*: для не фахівців в галузі експертизи, для фахівців, для учнів.

3. *За типами розв'язуваних задач*:

– *інтерпретуючі системи* – призначені для формування опису ситуацій за результатами спостережень або даними, одержуваними від різного роду сенсорів. Приклади: розпізнавання образів і визначення хімічної структури речовини;

– *прогнозуючі системи* – призначені для логічного аналізу можливих наслідків заданих ситуацій або подій. Приклади: прогнозування погоди і ситуацій на фінансових ринках;

– *діагностичні системи* – призначені для виявлення джерел несправностей за результатами спостережень за поведінкою контрольованої системи (технічної або біологічної). У цю категорію входить широкий спектр задач у всіляких предметних областях – медицині, механіці, електроніці і т. д.;

– *системи проектування* – призначені для структурного синтезу конфігурації об'єктів (компонентів проектованої системи) при заданих обмеженнях. Приклади: синтез електронних схем, компонування архітектурних планів, оптимальне розміщення об'єктів в обмеженому просторі;

– *системи планування* – призначені для підготовки планів проведення послідовності операцій, що призводить до заданої мети. Приклади: задачі планування поведінки роботів і складання маршрутів пересування транспорту;

– *системи моніторингу* – аналізують поведінку контрольованої системи і, порівнюючи отримані дані з критичними точками заздалегідь складеного плану, прогнозують імовірність досягнення поставленої мети. При-

клади: контроль руху повітряного транспорту і спостереження за станом енергетичних об'єктів;

– *налагоджувальні системи* – призначені для вироблення рекомендацій з усунення несправностей у контрольованій системі. До цього класу відносяться системи, що допомагають програмістам у налагодженні програмного забезпечення, і консультуючі системи;

– *системи надання допомоги при ремонті устаткування* – виконують планування процесу усунення несправностей у складних об'єктах, наприклад, у мережах інженерних комунікацій;

– *навчальні системи* – проводять аналіз знань студентів за визначеним предметом, відшукують пробіли в знаннях і пропонують засоби для їхньої ліквідації;

– *системи контролю* – забезпечують адаптивне керування поведінкою складних людино-машинних систем, прогнозуючи появу можливих збоїв і плануючи дії, необхідні для їхнього попередження. Приклади: керування повітряним транспортом, воєнними діями і діловою активністю в сфері бізнесу.

4. *За ступенем складності структури*:

– *поверхневі системи* – подають знання про область експертизи у вигляді правил (умова → дія). Умова кожного правила визначає зразок деякої ситуації, при дотриманні якої правило може бути виконано. Пошук рішення полягає у виконанні тих правил, зразки яких зіставляються з поточними даними. При цьому передбачається, що в процесі пошуку рішення послідовність формованих у такий спосіб ситуацій не обірветься до одержання рішення, тобто не виникне невідомої ситуації, що не зіставиться з жодним правилом;

– *глибинні системи* – крім можливостей поверхневих систем, мають здатність при виникненні невідомої ситуації визначати за допомогою деяких загальних принципів, справедливих для області експертизи, які дії варто виконати.

5. *За типом використовуваних методів і знань*:

– *традиційні системи* – використовують в основному неформалізовані методи інженерії знань і неформалізовані знання, отримані від експертів;

– *гібридні системи* – використовують методи інженерії знань і формалізовані методи, а також дані традиційного програмування та математики.

6. *За видами використовуваних даних і знань*: з детермінованими і невизначеними знаннями. Під невизначеністю знань і даних розуміються їхня неповнота, ненадійність, нечіткість.

7. *За способом формування рішення*:

– *аналізуючі системи* – вибір рішення здійснюється з множини відомих рішень на основі аналізу знань;

– *синтезуючі системи* – рішення синтезується з окремих фрагментів знань.

8. *За способом урахування часової ознаки*:

– *статичні системи* – призначені для вирішення задач з незмінними в процесі рішення даними і знаннями;

– *динамічні системи* – допускають зміни даних і знань у процесі рішення.

9. *За рівнем складності*:

– *прості системи*: поверхневі, традиційні (рідше гібридні) системи, виконані на персональних ЕОМ, з комерційною вартістю від 100 до 25 тисяч доларів, з вартістю розробки від 50 до 300 тисяч доларів, з часом розробки від 3 міс. до одного року, що містять від 200 до 1000 правил;

– *складні системи*: глибинні, гібридні системи, виконані або на символьних ЕОМ, або на потужній універсальній ЕОМ, або на інтелектуальній робочій станції, з комерційною вартістю від 50 тисяч до 1 мільйона доларів, із середньою вартістю розробки 5–10 мільйонів доларів, часом розробки від 1 до 5 років, що містять від 1,5 до 10 тисяч правил.

10. *За стадією існування* (ступенем пропрацьованності і налагодженості):

– *демонстраційний прототип* – система, що вирішує частину необхідних задач, демонструючи життєздатність методу інженерії знань. При наявності розвитих інструментальних засобів для розробки демонстраційного прототипу потрібно в середньому приблизно 1–2 міс., а при відсутності – 12–18 міс. Демонстраційний прототип працює, маючи 50–100 правил;

– *дослідницький прототип* – система, що вирішує всі необхідні задачі, але хитлива в роботі та не є цілком перевіреною. На доведення системи до стадії дослідницького прототипу йде 3–6 міс. Дослідницький прототип звичайно має 200–500 правил, що описують проблемну область;

– *діючий прототип* – надійно вирішує всі задачі, але для вирішення складних задач може знадобитися занадто багато часу та (або) пам'яті. Для доведення системи до стадії діючого прототипу потрібно 6–12 міс., при цьому кількість правил збільшується до 500–1000.

– *система промислової стадії* – забезпечує високу якість вирішення всіх задач при мінімумі часу і пам'яті. Звичайно процес перетворення діючого прототипу в промислову систему полягає в розширенні бази знань до 1000–1500 правил і переписуванні програм з використанням більш ефективних інструментальних засобів. Для доведення системи від початку розробки до стадії промислової системи потрібно 1–1,5 року;

– *комерційна система* – система, придатна не тільки для власного використання, але і для продажу різним споживачам. Для доведення системи до комерційної стадії потрібно 1,5–3 роки та 0,3–5 млн. доларів. При цьому в базі знань системи – 1500–3000 правил.

11. *За поколінням*:

– *системи першого покоління* – статичні поверхневі системи;

– *системи другого покоління* – статичні глибинні системи (іноді до другого покоління відносять також гібридні системи);

– *системи третього покоління* – динамічні системи, що, як правило, є глибинними і гібридними.

12. *За узагальненим показником – класом:*

– *класифікуючі системи* – вирішують задачі розпізнавання ситуацій. Основним методом формування рішень у таких системах є дедуктивне логічне виведення;

– *довизначальні системи* – використовуються для вирішення задач з не цілком визначеними даними і знаннями. У таких системах виникають задачі інтерпретації нечітких знань і вибору альтернативних напрямків пошуку в просторі можливих рішень. Як методи обробки невизначених знань можуть використовуватися байєсівський імовірнісний підхід, коефіцієнти впевненості, нечітка логіка;

– *трансформуючі системи* – відносяться до синтезуючих динамічних експертних систем, у яких передбачається повторюване перетворення знань у процесі вирішення задач. У системах даного класу використовуються різні способи обробки знань: генерація і перевірка гіпотез, логіка припущень і умовчань (коли за неповними даними формуються подання про об'єкти визначеного класу, що згодом адаптуються до конкретних умов ситуацій, що змінюються), використання метазнань (більш загальних закономірностей) для усунення невизначеностей у ситуаціях;

– *мультиагентні системи* – динамічні системи, засновані на інтеграції декількох різнорідних джерел знань, що обмінюються між собою одержуваними результатами в ході вирішення задач. Системи даного класу мають можливості реалізації альтернативних міркувань на основі використання різних джерел знань і механізму усунення протиріч, розподіленого вирішення проблем, що декомпозуються на паралельно розв'язувані підзадачі із самостійними джерелами знань, застосування різних стратегій виведення рішень у залежності від типу розв'язуваної проблеми, обробки великих масивів інформації з баз даних, використання математичних моделей і зовнішніх процедур для імітації розвитку ситуацій.

1.4.3 Життєвий цикл та методологія розробки експертних систем

У розробці й експлуатації експертної системи беруть участь такі фахівці.

– *Експерт* – висококваліфікований фахівець у проблемній області, задачі якої повинна вирішувати експертна система, який визначає знання, що характеризують проблемну область, забезпечує повноту і правильність введених в систему знань.

– *Інженер зі знань* (когнітолог) – фахівець з розробки експертних систем, який виступає в ролі проміжного буфера між експертом і базою знань. Він допомагає експерту виявити і структурувати знання, необхідні для роботи експертної системи, здійснює вибір того інструментального засобу, що найбільше підходить для даної проблемної області, визначає спосіб подання знань у цьому інструментальному засобі, виділяє і програмує (традиційними засобами) стандартні функції (типові для даної проблемної області), що будуть використовуватися в знаннях, які вводяться експертом.

– *Програміст* – фахівець з розробки інструментальних засобів, що здійснює їхнє створення і сполучення з тим середовищем, у якому вони будуть використовуватися.

– *Користувач* – людина, що використовує вже побудовану експертну систему. Термін користувач є трохи неоднозначним. Звичайно він позначає *кінцевого користувача*. Однак користувачем може бути програміст, що відлагоджує засіб побудови експертної системи, інженер зі знань, що уточнює існуючі в системі знання, експерт, що додає в систему нові знання, оператор, що заносить у систему поточну інформацію.

Склад і взаємодію учасників побудови й експлуатації експертних систем зображено на рис. 1.1.

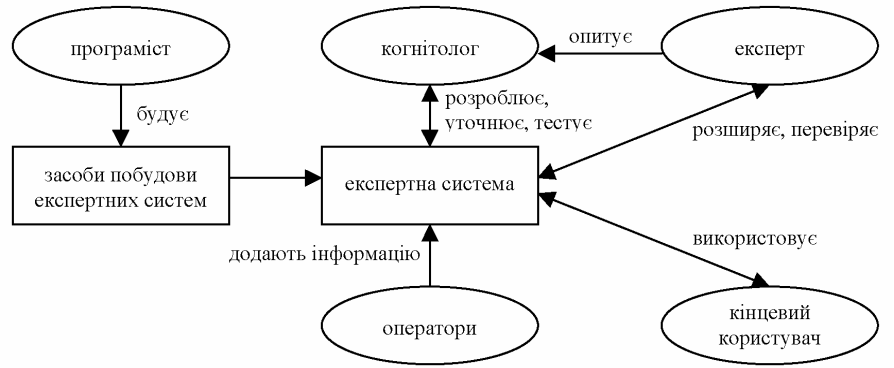


Рисунок 1.1 – Взаємозв'язки основних учасників
побудови та експлуатації експертних систем

Засіб побудови експертної системи – це програмний засіб, використовуваний інженером зі знань або програмістом для побудови експертної системи. Цей інструмент відрізняється від звичайних мов програмування тим, що забезпечує зручні способи подання складних високорівневих понять. Важливо розрізняти інструмент, що використовується для побудови експертної системи, і саму експертну систему. Інструмент побудови включає як мову, використовувану для доступу до знань, що містяться в системі, і їхнього подання, так і підтримуючі засоби – програми, що допомагають користувачам взаємодіяти з компонентом експертної системи, яка вирішує проблему.

Концепція «швидкого прототипу» використовується для розробки експертних систем і полягає в тому, що розробники не намагаються відразу створити кінцевий продукт. На початковому етапі вони створюють прототип (прототипи) експертної системи, що повинний задовольняти двом суперечливим вимогам:

з одного боку, вирішувати типові задачі конкретного застосування, а з іншого боку – час і трудомісткість його розробки повинні бути дуже незначними, щоб можна було максимально запаралелити процес накопичення і відлагодження знань (здійснюваний експертом) із процесом вибору (розробки) програмних засобів (здійснюваним інженером зі знань і програмістом). Для задоволення зазначених вимог при створенні прототипу, як правило, використовуються різноманітні інструментальні засоби, що прискорюють процес проектування.

Прототип повинний продемонструвати придатність методів інженерії знань для даного застосування. У випадку успіху експерт за допомогою інженера зі знань розширює знання прототипу про проблемну область. При невдачі може знадобитися розробка нового прототипу або розробники можуть прийти до висновку про непридатність методів інженерії знань для даного застосування. В міру збільшення знань прототип може досягти такого стану, коли він успішно вирішує всі задачі даного застосування. Перетворення прототипу в кінцевий продукт звичайно приводить до перепрограмування експертної системи на мовах низького рівня, що забезпечують як підвищення її швидкодії, так і зменшення необхідної пам'яті.

Методологія розробки експертних систем включає такі етапи (див. рис. 1.2).

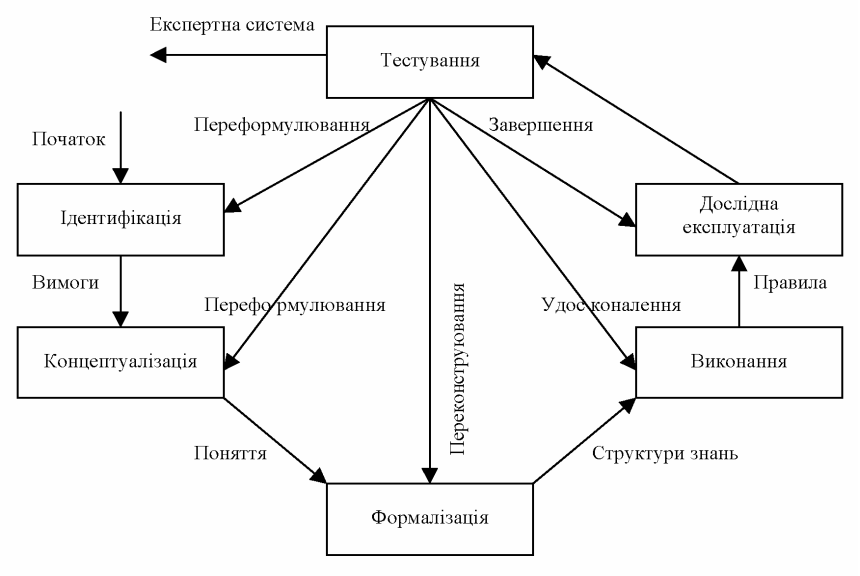


Рисунок 1.2 – Етапи розробки експертних систем

1. *Етап ідентифікації:* визначаються задачі, що підлягають вирішенню, складаються їхні неформальні описи (загальні характеристики задачі;

підзадачі, що виділяються усередині даної задачі; ключові поняття, характеристики і відношення; вхідні і вихідні дані; можливий вид рішення; знання, релевантні розв'язуваній задачі; ситуації, що перешкоджають вирішенню задачі; приклади рішення задачі; критерії оцінки якості рішень), виявляються цілі розробки (формалізація неформальних знань експертів; поліпшення якості рішень, прийнятих експертом; автоматизація рутинних аспектів роботи експерта або користувача; тиражування знань експерта), ресурси (джерела знань, трудомісткість, час розробки, обчислювальні засоби й обсяг фінансування), експерти, інженери зі знань і категорії користувачів, визначаються форми взаємин учасників розробки.

2. *Етап концептуалізації*: експерт та інженер зі знань виконують змістовний аналіз проблемної області, виявляють використовувані поняття і їхні взаємозв'язки (типи використовуваних відношень: ієрархія, причина – наслідок, частина – ціле і т. п.), визначають гранулярність (рівень деталізації) знань, визначають особливості задачі (типи доступних даних; вихідні і вихідні дані, підзадачі загальної задачі), визначають методи вирішення задач (використовувані стратегії і гіпотези; процеси, використовувані в ході рішення задачі; типи обмежень, що накладаються на процеси, використовувані в ході рішення; склад знань, використовуваних для рішення задачі; склад знань, використовуваних для пояснення рішення).

3. *Етап формалізації*: визначаються способи подання й інтерпретації усіх видів знань, формалізуються (подаються формальною мовою) основні поняття і відношення, подається структура простору станів і характер методів пошуку в ньому, моделюється робота системи, вибираються програмні засоби розробки, оцінюється адекватність цілям і повнота системи зафіксованих понять, методів рішення, засобів подання і маніпулювання знаннями.

4. *Етап виконання* (реалізація): здійснюється набуття знань системою, яке розділяють на витяг знань з експерта, організацію знань, що забезпечує ефективну роботу системи, і подання знань у вигляді, зрозумілому експертній системі. Мета цього етапу – створення одного або декількох прототипів, що вирішують необхідні задачі.

5. *Етап тестування*: експерт (та інженер зі знань) в інтерактивному режимі, використовуючи діалогові та пояснювальні засоби, перевіряє компетентність експертної системи на великій кількості репрезентативних задач. Процес тестування продовжується доти, поки експерт не вирішить, що система досягла необхідного рівня компетентності.

6. *Етап дослідної експлуатації*: перевіряється придатність експертної системи для кінцевих користувачів, яка визначається в основному зручністю роботи із системою та її корисністю. За результатами цього етапу може знадобитися істотна модифікація експертної системи. Після успішного завер-

шення етапу дослідної експлуатації і використання різними користувачами експертна система може класифікуватися як комерційна.

Модифікація експертної системи здійснюється майже постійно в ході її створення. Виділяють такі види модифікації системи:

– *Удосконалення прототипу* – здійснюється в процесі циклічного проходження через етапи виконання і тестування для налагодження правил і процедур виведення. Цикли повторюються доти, поки система не буде поводитися очікуваним чином. Зміни, здійснювані при удосконаленні, залежать від обраного способу подання і класу задач, розв'язуваних системою. Якщо в процесі удосконалення бажане поводження не досягається, то необхідно здійснити більш серйозні модифікації архітектури системи і бази знань;

– *Переконструювання подання* – перегляд обраного раніше способу подання знань, здійснюваний у результаті повернення від етапу тестування до етапу формалізації;

– *Переформулювання понять*, використовуваних у системі – проектування всієї системи практично заново, здійснюване в результаті повернення на етапи концептуалізації й ідентифікації після невдачі на етапі тестування.

1.4.4 Структура та функціонування експертної системи

Ідеальна статична експертна система (рис. 1.3) містить такі основні компоненти, як: машина логічного виведення (вирішувач, інтерпретатор правил), база знань, підсистема набуття знань, підсистема пояснення рішень, інтерфейсна підсистема (діалоговий компонент), робоча пам'ять (база даних). Такі експертні системи використовуються в тих застосуваннях, де можна не враховувати зміни навколишнього світу за час рішення задачі.

База знань – частина експертної системи, що містить подання знань, яке стосується визначеної предметної області. У статичній частині бази знань зберігаються довгострокові знання, що описують розглянуту предметну область у вигляді *загальних фактів* (фраз без умов, що містять твердження, які завжди є абсолютно вірними) і *правил* (тверджень, істинність яких залежить від деяких умов, що утворюють тіло правила), які описують доцільні перетворення фактів цієї області з метою породження нових фактів або гіпотез.

Робоча пам'ять (база даних) – динамічна частина бази знань, що змінює свій стан під впливом правил, призначена для збереження вихідних даних (фактів, що описують поточну ситуацію) і проміжних даних розв'язуваної в поточний момент задачі (фактів, що були встановлені до визначеного моменту в результаті виведення, яке полягає в застосуванні правил до наявних фактів).

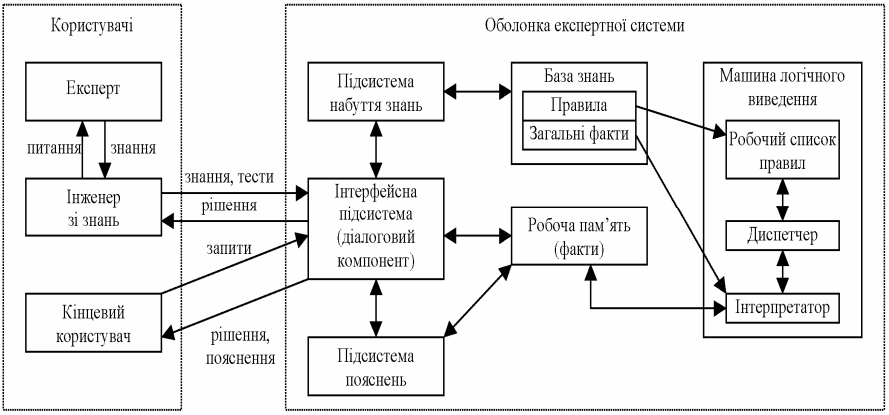


Рисунок 1.3 – Схема ідеальної статичної експертної системи

Машина логічного виведення (механізм логічного виведення, вирішувач) – основна частина експертної системи, яка, використовуючи інформацію з бази знань, на основі стратегії, тісно пов'язаної зі способом подання знань в експертній системі і характером розв'язуваних задач, генерує рекомендації з вирішення задачі та містить:

– *інтерпретатор* – компонент, який, вишиковуючи правила в ланцюжок для досягнення поставленої користувачем мети, послідовно визначає, які правила можуть бути активовані в залежності від умов, що у них містяться, вибирає одне з застосовних у даній ситуації правил і виконує його;

– *диспетчер* – компонент, що установлює порядок застосування активованих правил;

– *робочий список правил* – створений машиною логічного виведення і впорядкований за пріоритетами список правил, шаблони яких задовольняють фактам або об'єктам, що знаходяться в робочій пам'яті.

Оболонка експертної системи – програма, що забезпечує взаємодію між базою знань та машиною логічного виведення. Кінцевий користувач взаємодіє з оболонкою через інтерфейсну підсистему, передаючи їй запити. Остання активізує машину логічного виведення, яка звертається до бази знань, витягає з неї і генерує в процесі логічного виведення знання, необхідні для відповіді на конкретне питання, і передає сформовану відповідь користувачу або як рішення проблеми, або у формі рекомендації чи поради.

Інтерфейсна підсистема (діалоговий компонент) розподіляє ролі користувачів і експертної системи, а також організує їхню взаємодію в процесі кооперативного вирішення задачі за допомогою перетворення запитів користувачів у внутрішню мову подання знань експертної системи і перетворення повід-

омлень системи, поданих внутрішньою мовою, у повідомлення мовою, звичною для користувача (обмеженою природною мовою або мовою графіки).

Підсистема набуття знань – автоматизує процес наповнення експертної системи знаннями експертом або інженером зі знань через редактор бази знань без залучення інженера зі знань до вирішення задачі явного кодування знань або автоматично витягає знання з наборів даних у процесі навчання на основі дерев рішень, методів виділення асоціативних правил, штучних нейронних або нейро-нечітких мереж.

Редактор бази знань – складова частина підсистеми набуття знань, що являє собою транслятор з деякої підмножини природної мови, використовуваної інженером зі знань і експертом, у спеціальний код, орієнтований на роботу механізму логічного виведення.

Підсистема пояснень – дозволяє контролювати хід суджень експертної системи і пояснювати її рішення або їхню відсутність з указівкою використаних знань, а також виявляти неоднозначності і протиріччя в базі знань експертної системи, що полегшує експерту тестування системи і підвищує довіру користувача до отриманого результату, а також дозволяє навчати користувача рішенню відповідних задач.

Динамічні експертні системи – більш високий клас програмних засобів у порівнянні зі статичними експертними системами, що враховують динаміку зміни навколишнього світу за час виконання програми. У порівнянні зі статичною експертною системою в динамічну вводяться ще два компоненти: підсистема моделювання зовнішнього світу і підсистема сполучення з зовнішнім світом (рис. 1.4).

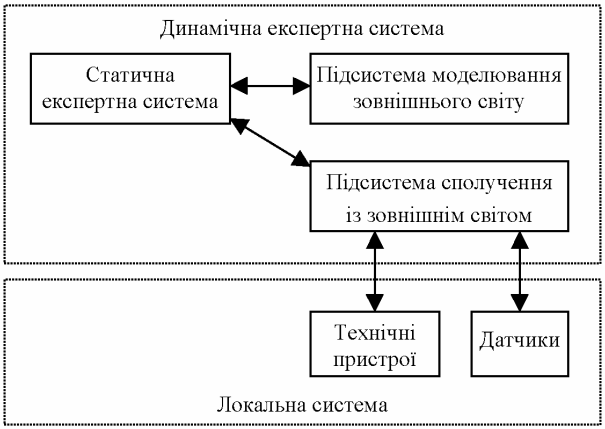


Рисунок 1.4 – Динамічна експертна система

Динамічна експертна система здійснює зв'язки з зовнішнім світом через систему контролерів і датчиків. Крім того компоненти бази знань і механізму виведення істотно змінюються, щоб відбити тимчасову логіку подій, які відбуваються в реальному світі.

Режими роботи експертної системи виділяють такі.

– *Режим набуття знань*: спілкування з експертною системою здійснює експерт за посередництвом інженера зі знань. Експерт описує проблемну область у вигляді сукупності фактів і правил. Факти визначають об'єкти, їхні характеристики і значення, що існують в області експертизи. Правила визначають способи маніпулювання даними, характерні для розглянутої проблемної області. Експерт, використовуючи компонент набуття знань, наповняє систему знаннями, що дозволяють експертній системі в режимі рішення самостійно (без експерта) вирішувати задачі з проблемної області.

Важливу роль у режимі набуття знань грає пояснювальний компонент. Саме завдяки йому експерт на етапі тестування локалізує причини невдалої роботи експертної системи, що дозволяє експерту цілеспрямовано модифікувати старі або вводити нові знання. Звичайно пояснювальний компонент повідомляє таке: як правила використовують інформацію користувача; чому використовувалися або не використовувалися дані чи правила; які були зроблені висновки і т. п. Усі пояснення робляться, як правило, обмеженою природною мовою або мовою графіки.

– *Режим консультації* (рішення): спілкування з експертною системою здійснює кінцевий користувач, якого цікавить результат та (або) спосіб одержання рішення. Користувач у залежності від призначення експертної системи може не бути фахівцем у даній проблемній області, у цьому випадку він звертається до системи за порадою, не вміючи одержати відповідь самостійно, або бути фахівцем, у цьому випадку він звертається до системи, щоб або прискорити процес одержання результату, або покласти на систему рутинну роботу. У режимі консультації дані про задачу користувача обробляються діалоговим компонентом. Після обробки дані надходять у робочу пам'ять. На основі вхідних даних з робочої пам'яті, загальних даних про проблемну область і правил з бази знань вирішувач (інтерпретатор) формує рішення задачі. На відміну від традиційних програм експертна система в режимі рішення задачі не тільки виконує запропоновану послідовність операцій, але і попередньо формує її. Якщо відповідь системи є незрозумілою користувачу, то він може зажадати пояснення того, як відповідь отримана.

1.4.5 Переваги і недоліки експертних систем

Перевагами експертних систем є:

– сталість: знання експертної системи зберігаються протягом невизначено довгого часу і нікуди не зникають, у той час як людська компетенція слабшає із часом, перерва у діяльності людини-експерта може серйозно відбитися

на її професійних якостях, крім того експерти-люди можуть піти на пенсію, звільнитися з роботи або вмерти, тобто їхні знання можуть бути втрачені;

– легкість передачі або відтворення: передача знань від однієї людини до іншої – довгий і дорогий процес, передача штучної інформації – це простий процес копіювання програми або файлу даних;

– підвищена доступність: експертна система – засіб масового виробництва експертних знань, що дозволяє багатьом користувачам одержати доступ до експертних знань;

– можливість одержання й об'єднання експертних знань з багатьох джерел: за допомогою експертних систем можуть бути зібрані знання багатьох експертів і притягнуті до роботи над задачею, виконуваної одночасно і безупинно у будь-яку годину дня і ночі; рівень експертних знань, скомбінованих шляхом об'єднання знань декількох експертів, може перевищувати рівень знань окремо узятого експерта-людини;

– стійкість і відтворюваність результатів: експерт-людина може приймати в тотожних ситуаціях різні рішення через емоційні фактори або утому, у той час, як результати експертних систем стабільні і являють собою незмінно правильні, позбавлені емоцій і повні відповіді за будь-яких обставин;

– низька вартість: експерти, особливо висококваліфіковані, обходяться дуже дорого, у той час, як експертні системи, навпаки, є порівняно недорогими – їхня розробка є дорогою, але вони є дешевими в експлуатації: вартість надання експертних знань у розрахунку на окремого користувача істотно знижується;

– зменшена небезпека: експертні системи можуть використовуватися в таких варіантах середовища, що можуть виявитися небезпечними для людини;

– швидкий відгук: експертна система може реагувати швидше і бути більш готовою до роботи, ніж експерт-людина, особливо в деяких екстремальних ситуаціях, де може знадобитися більш швидка реакція, ніж у людини;

– підвищена надійність: застосування експертних систем дозволяє підвищити ступінь довіри до того, що прийнято правильне рішення, шляхом надання ще однієї обґрунтованої думки людині-посереднику за наявності неузгоджених думок між декількома експертами-людьми;

– можливість пояснення рішень: експертна система здатна докладно пояснити свої рішення, що привели до визначеного висновку, а людина може виявитися занадто втомленою, не схильною до пояснень або нездатною робити це постійно;

– можливість застосування в якості інтелектуальної навчальної програми: експертна система може діяти як інтелектуальна навчальна програма, передаючи учню на виконання приклади програм і пояснюючи, на чому засновані судження системи;

– можливість застосування у якості інтелектуальної бази даних: експертні системи можуть використовуватися для доступу до баз даних за допомогою інтелектуального способу доступу;

– формалізація і перевірка знань: у процесі розробки експертної системи знання експертів-людей перетворяться в явну форму для введення в комп'ютер, у результаті чого вони стають явно відомими і з'являється можливість перевіряти знання на правильність, несуперечність і повноту.

Недоліки експертних систем:

– експертні системи погано вміють: подавати знання про часові та просторові відношення, розмірковувати, виходячи зі здорового глузду, розпізнавати межі своєї компетентності, працювати із суперечливими знаннями;

– інструментальні засоби побудови експертних систем погано вміють: виконувати набуття знань, уточнювати бази знань, працювати зі змішаними схемами подання знань;

– побудова експертних систем не під силу кінцевому користувачу, який не володіє експертними знаннями про проблемну область;

– необхідність залучення людини-експерта з проблемної області, що є носієм знань; неможливість повного відмовлення від експерта-людини;

– можливі труднощі взаємодії експерта зі спеціалістом-когнітологом, який шляхом діалогу з експертом оформляє отримані від експерта знання в обраному формалізмі подання знань;

– необхідність повної переробки програмного інструментарію, у випадку, якщо наявна оболонка експертної системи та / або використовувана нею модель подання знань погано підходять для обраної проблемної області, задачі;

– тривалість процесів витягу знань з експерта, їхньої формалізації, перевірки на несуперечність і усунення протиріч.

Експертні системи залучають значні грошові інвестиції і людські зусилля. Спроби вирішити занадто складну, малозрозумілу чи, іншими словами, не відповідну наявній технології проблему можуть привести до дорогих і ганебних невдач.

Використання експертної системи є доцільним, якщо розробка експертної системи можлива, виправдана і доречна.

Розробка експертної системи можлива, якщо: задача не вимагає загальнодоступних знань, задача вимагає тільки інтелектуальних навичок, експерти можуть описати свої методи природною мовою, існують справжні експерти, експерти одностайні щодо рішень, задача не є занадто важкою, задача є цілком зрозумілою, проблемна область є добре структурованою і не вимагає розмірковувань на основі здорового глузду – широкого спектру загальних відомостей про світ та спосіб його функціонування, котрі знає та вміє використовувати будь-яка нормальна людина.

Розробка експертної системи виправдана, хоча б в одному з випадків, якщо: одержання рішення є високорентабельним, втрачається людський досвід, експертів мало, досвід потрібний одночасно у багатьох місцях, досвід необхідно застосовувати у ворожих людині умовах, людський досвід відсутній у ситуаціях, де він є необхідним.

Застосування експертної системи розумно, якщо: задача вимагає оперування символами, задача не може бути вирішена традиційними обчислювальними методами і вимагає евристичних рішень, задача не є занадто простою, задача становить практичний інтерес, задача має розміри, що допускають реалізацію.

1.5 Логічне виведення

Логічне виведення – процес одержання з вихідних фактів за заданими правилами нових фактів, що логічно випливають з вихідних.

Типи логічного виведення виділяють такі.

– *Дедукція* – логічний розсуд, у якому висновки повинні випливати з відповідних ним посилок.

– *Індукція* – логічне виведення від окремого випадку до загального – один з основних методів машинного навчання, у якому комп'ютери навчаються без утручання людини. До цих методів відноситься кластер-аналіз і нейронні та нейро-нечіткі мережі, що самоорганізуються.

– *Інтуїція* – метод, не заснований на перевіреній теорії. Відповідь являє собою усього лише припущення, можливо, сформульоване шляхом підсвідомого розпізнавання якогось основного образа. Логічне виведення такого типу ще не реалізовано в експертних системах.

– *Евристика* – емпіричні (отримані на основі досвіду) правила виведення.

– *Метод породження і перевірки* – метод проб і помилок. Часто використовується в сполученні з плануванням для досягнення максимальної ефективності.

– *Абдукція* – метод формування суджень у зворотному напрямку, від істинного висновку до посилок, що могли привести до одержання цього висновку.

– *Судження, застосовувані за замовчуванням* – судження, що допускають можливість під час відсутності конкретних знань приймати за замовчуванням загальноприйняті чи загальновідомі знання.

– *Аутоепістемічні судження* – самопізнання, або міркування про те, яким людині уявляється деякий об'єкт, його властивість.

– *Немонотонні судження* – судження, застосовувані в тих умовах, коли раніше отримані знання можуть виявитися неправильними після одержання нового свідчення.

– *Судження за аналогією* – логічне виведення виходячи з наявності ознак, подібних до ознак іншої ситуації.

1.5.1 Дедуктивне логічне виведення

Звичайне (булеве) логічне виведення висновків на основі відомих фактів і правил широко використовується в експертних системах і базується на таких тавтологіях:

– *модус поненс* (modus ponens): $(A \wedge (A \rightarrow B)) \rightarrow B$. Модус поненс виводить висновок «*B* є істинно», якщо відомо, що «*A* є істинно» й існує правило «Якщо *A*, то *B*», де *A* та *B* – чіткі логічні твердження.

– *модус толенс* (modus tollens): $((\neg B) \wedge (A \rightarrow B)) \rightarrow \neg A$. Модус толенс виводить висновок «*A* є хибним», якщо відомо, що «*B* є хибним» та існує правило «Якщо *A*, то *B*».

– *силогізм* (syllogism): $((A \rightarrow B) \wedge (B \rightarrow C)) \rightarrow (A \rightarrow C)$. Силогізм виводить висновок «Якщо *A*, то *C*» («З *A* випливає *C*»), якщо відомо, що «Якщо *A*, то *B*» («З *A* випливає *B*») і «Якщо *B*, то *C*» («З *B* випливає *C*»).

– *контрапозиція* (contraposition): $(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$. Контрапозиція виводить висновок «Якщо *B* є хибним, то *A* є хибним», якщо відомо, що «Якщо *A*, то *B*».

Найбільш відомими є методи виведення висновків: прямий і зворотний.

Прямий метод виведення висновків (метод висхідного виведення, метод прямого ланцюжка суджень, classic forward-chaining reasoning), заснований на використанні правила виведення модус поненс, що дозволяє визначити істинність висновку правила при відомій істинності його умови. Прямий метод виведення реалізується за допомогою перетворення окремих фактів проблемної області в конкретні значення істинності умов правил. Після цього перетворення ті з правил, для яких стають істинними відповідні умови, генерують висновки своїх правих частин. Ці висновки приймаються як істинні і стають новими фактами, що можуть бути використані як умови в розглянутій базі правил. При цьому правила, для яких є істинними умови, називають *активними*.

Процес виведення прямим методом має рекурсивний характер і може бути зупинений або у випадку відсутності нових активних правил, або у випадку одержання висновку, що є цільовим у контексті рішення вихідної проблеми. Подібне підтвердження цільового висновку характеризує успіх процесу виведення, оскільки тільки в цьому випадку використання системи правил характеризує рішення поставленої проблеми.

Ланцюжок прямого виведення висновків (forward chaining) – шлях суджень, що будується, відштовхуючись від фактів (умов, про які відомо, що вони задовольняються), до гіпотез (стану проблеми, що випливає з цих умов).

Зворотний метод виведення висновків (метод спадного виведення, classic backward-chaining reasoning), заснований на використанні правила модус толенс, що дозволяє визначити хибність умови правила при відомій хибності його висновку.

Зворотний метод виведення в експертних системах реалізується в модифікованому вигляді за допомогою дослідження можливості застосування правил для підтвердження деяких заздалегідь заданих висновків: заперечення висновку заміняється питанням про його істинність. Символічно це записується у вигляді: $(B? \wedge (A \rightarrow B)) \rightarrow A?$, що означає, що у випадку істинності імплікації $A \rightarrow B$ до-

статньою умовою істинності формули *B* є істинність формули *A*. Таким чином, якщо метою виведення є доказ істинності висновку *B*, то для цього досить довести істинність умови *A*, розглянутої як підціль. Тому зворотний метод служить обґрунтуванням достатніх умов для істинності висновків правил.

Процес зворотного виведення починається з підстановки окремих цікавлячих нас висновків у праві частини відповідних правил, які у цьому випадку стають активними. Після аналізу кожного з активних правил фіксуються умови, що підтверджують ці правила. Ці умови приймаються як істинні і стають новими фактами, що можуть бути використані в якості нових цільових висновків у розглянутій базі правил.

Процес виведення зворотним методом має рекурсивний характер і може бути зупинений або у випадку відсутності нових активних правил, або у випадку одержання підтвердження умов, що є істинними чи відомими фактами проблемної області.

Ланцюжок зворотного виведення (backward chaining) – шлях суджень, що будується, відштовхуючись від заданої цілі (гіпотез, що подають цільовий стан системи) до умов, при яких можливе досягнення цієї цілі (до фактів).

1.5.2 Індуктивне логічне виведення

Метод ДСМ (метод Джона Стюарта Міля) запропонований у середині XIX століття і є методом індуктивного виведення. Способи встановлення причинно-наслідкових відношень, запропоновані Мілем, ґрунтуються на ідеях виявлення подібності та розходження в ситуаціях, що спостерігаються. Здатність уловлювати подібність і виділяти розходження – фундаментальна здатність, властива, очевидно, усім живим істотам. Спираючись на цю здатність, Міль сформулював такі *принципи індукції*.

1. *Принцип єдиного розходження*: якщо після введення якого-небудь фактора з'являється (чи після його видалення зникає) відоме явище, причому ми не вводимо і не видаляємо ніякої іншої обставини, що могла б мати вплив, то зазначений фактор складає причину явища. Цей принцип можна проілюструвати схемою:

$$
\begin{aligned}
&A, B, C \to D,\\
&A, B, C \to D,\\
&\ldots\ldots\ldots\\
&A, B, C \to D,\\
&B, C \not\to D,
\end{aligned}
$$

де знак «$\to$» трактується як поява D при наявності A, B, C. При достатній кількості експериментів принцип єдиного розходження дозволяє стверджувати, що A є причиною, а D – наслідком.

2. *Принцип єдиної подібності*: якщо всі обставини явища, крім одної, можуть бути відсутні, не знищуючи цим явища, та ця обставина є причиною даного явища. Схема принципу така:

$$A, B, C \to D,$$
$$A, B, C \to D,$$
$$\ldots\ldots\ldots\ldots$$
$$A, B \to D,$$
$$A, C \to D,$$
$$\ldots\ldots\ldots\ldots$$
$$A \to D.$$

З цієї схеми випливає, що A та D пов'язані причинно-наслідковим відношенням.

3. *Принцип єдиного залишку*: якщо відняти з якого-небудь явища ту його частину, що є наслідком відомих причин, то залишок явища є наслідком інших причин. Розглянемо схему:

$$A, B, C \to D, E$$
$$A, B, C \to D, E$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$B, C \to E.$$

Після того як із прикладів A, B, $C \to D$, E було «відняте» причинно-наслідкове відношення $A \to D$, були отримані спостереження B, $C \to E$, на підставі яких можна припустити, що B та C є можливими причинами явища E. Для подальшого уточнення потрібно перевірити, чи приводить виключення B до появи E. Якщо так, то причиною явища E слугує C, у противному випадку – B. Можливо також, що явище E обумовлене одночасною наявністю B та C, тобто поява деякого елемента ситуації може визначатися не окремими факторами, а їхньою сукупністю.

Схеми Міля справедливі лише за умови, що в описі ситуації присутня повна множина фактів і явищ, що спостерігаються.

Нехай задана множина причин $A = \{A_1, A_2, ..., A_p\}$, множина наслідків $B = \{B_1, B_2, ..., B_m\}$ і множина оцінок $Q = \{q_1, q_2, ..., q_r\}$.

Вираз виду $A_i \to B_j$ називається позитивною гіпотезою, що виражає твердження «A_i є причиною B_j, з оцінкою вірогідності q_k». Негативною гіпотезою називається вираз $A_i \not\to B_j$, що формулюється «A_i не є причиною B_j з оцінкою вірогідності q_k». Позитивні гіпотези будемо позначати $h^{+}_{i,j,k}$, негативні – $h^{-}_{i,j,k}$.

Серед значень виділимо два спеціальних, котрі можна інтерпретувати як «хибність» (0) та «істина» (1). Гіпотези з цими оцінками можна розглядати як явища, істинність або хибність яких твердо встановлено. Інші значення між 0 та 1 будемо позначати раціональними числами k/n, де $k = 1, ..., n–1$, а n характеризує число прикладів.

Узагальнений ДСМ-метод включає такі кроки.

Крок 1. На основі вихідної множини позитивних і негативних прикладів (спостережень) формується набір гіпотез, що записуються в матриці M^+ та M^-. Гіпотези формуються на основі виявлення подібності і розходження в прикладах. Матриці мають вигляд:

$$M^+ = \begin{array}{c|c|c|c} & B_i & \dots & B_w \\ \hline A_l & h^+_{l,j,k} & \dots & h^+_{l,w,m} \\ \dots & \dots & \dots & \dots \\ A_r & h^+_{r,i,s} & \dots & h^+_{r,w,d} \end{array}; \quad M^- = \begin{array}{c|c|c|c} & B_j & \dots & B_v \\ \hline A_x & h^-_{x,j,k} & \dots & h^-_{x,v,t} \\ \dots & \dots & \dots & \dots \\ A_z & h^-_{z,j,m} & \dots & h^-_{z,v,f} \end{array}.$$

Крок 2. До вихідної множини прикладів додаються нові спостереження, що можуть або підтверджувати висунуті гіпотези, або спростовувати їх, при цьому оцінки гіпотез змінюються в такий спосіб. Якщо деяка гіпотеза $h_{i,j,k}$ мала оцінку $q_k = k/n$, то з появою нового приклада (n+1) проводиться перевірка на підтвердження цієї гіпотези. У випадку позитивної відповіді оцінка $q_k = (k+1)/(n+1)$, інакше $q_k = (k–1)/(n+1)$. У процесі накопичення інформації оцінки висунутих гіпотез можуть наближатися до 1 або 0. Зміна оцінок може також мати коливальний характер, що, як правило, веде до виключення таких гіпотез з множин M^+ або M^-.

Крок 3. Циклічне додавання прикладів, що супроводжується зміною оцінок вірогідності гіпотез з періодичною зміною множин M^+ та M^-.

Крок 4. Завершення процесу індуктивного виведення при виконанні умов закінчення циклу. Як такі умови можуть використовуватися міри близькості значень q_i до 0 або 1, а також додаткові умови, що можуть бути пов’язані з обмеженням часу (кількості нових прикладів) виведення і т. п.

У сучасних модифікаціях ДСМ-методу використовується виведення за аналогією, враховується контекст реалізації причинно-наслідкових відношень, застосовуються нечіткі описи фактів і т. д.

1.5.3 Абдуктивне логічне виведення

Абдукція є необгрунтованим правилом виведення $((A \to B) \wedge B) \to A$ і означає, що висновок не є обов’язково істинним для кожної інтерпретації, при якій істинні передумови. Абдуктивні судження часто називають найкращим поясненням даних B.

Традиційна логіка має обмеження на використання в умовах неповної і невизначеної інформації і є монотонною.

Монотонна логіка заснована на множині аксіом, прийнятих за істинні, з яких виводяться наслідки. При цьому додавання в систему нової інформації

може викликати тільки збільшення множини істинних тверджень, що приводить до проблем при моделюванні суджень, заснованих на довірі і припущеннях.

Немонотонна логіка вирішує проблему моделювання суджень, заснованих на довірі і припущеннях за рахунок того, що, на відміну від математичних аксіом, міри довіри і висновки можуть мінятися в міру накопичення інформації.

Система немонотонних суджень керує ступенем невизначеності, роблячи найбільш визначені припущення в умовах невизначеної інформації. Потім виконується виведення на основі цих припущень, прийнятих за істинні. Пізніше міра довіри може змінитися і зажадати повторного перегляду усіх висновків, виведених з її використанням.

До методів абдуктивного виведення відносять стенфордську теорію коефіцієнта впевненості, нечітку логіку (див. розділ 3), теорію Демпстера-Шафера, метод Байєса, метод Нейлора.

Стенфордська теорія коефіцієнта впевненості заснована на неформальних оцінках фактів і висновків.

Коефіцієнт упевненості (коефіцієнт довіри, вірогідності, certainty factor) – це число, що означає імовірність або ступінь упевненості, з яким можна вважати даний факт або правило достовірним або справедливим.

Для визначення коефіцієнта упевненості використовуються методи математичної статистики, а також суб'єктивні оцінки експерта.

Коефіцієнти упевненості використовують у продукційних моделях (п. 2.3.8).

Коефіцієнт упевненості антецедента CF(*X*, *Y*), що містить посилки *X* та *Y*, обчислюється за формулою:

$$CF(X,Y)=\begin{cases}1, CF(X)=1 \text{ або } CF(Y)=1;\\ CF(X)+CF(Y)-CF(X)CF(Y),\ CF(X)>0,\ CF(Y)>0;\\ \dfrac{CF(X)+CF(Y)}{1-\min\{|CF(X)|,|CF(Y)|\}},\ CF(X)CF(Y)\le 0,\ CF(X)\ne\pm1, CF(Y)\ne\pm1;\\ CF(X)+CF(Y)+CF(X)CF(Y),\ CF(X)<0,\ CF(Y)<0;\\ -1, CF(X)=-1 \text{ або } CF(Y)=-1,\end{cases}$$

де *CF*(*X*) – коефіцієнт упевненості посилки *X*, *CF*(*Y*) – коефіцієнт упевненості посилки *Y*.

Коефіцієнт упевненості консеквента CF(*C*), визначається на основі коефіцієнта впевненості антецедента *CF*(*X*, *Y*) і коефіцієнта впевненості правила *CF*(*R*) як *CF*(*C*) = *CF*(*X*, *Y*)*CF*(*R*).

Коефіцієнт впевненості приймає значення в діапазоні [–1, +1]. Якщо він дорівнює +1, то це означає, що при дотриманні всіх обговорених умов укладач правила абсолютно упевнений у правильності висновку, а якщо він дорівнює –1, то виходить, що при дотриманні всіх обговорених умов існує абсолютна впевненість у помилковості цього висновку. Відмінні від +1 пози-

тивні значення коефіцієнта вказують на ступінь впевненості в правильності висновку, а негативні значення – на ступінь впевненості в його помилковості.

Коефіцієнт упевненості є комбінованою оцінкою. Його основне призначення полягає в тому, щоб керувати ходом виконання програми при формуванні суджень, керувати процесом пошуку мети в просторі станів (якщо коефіцієнт упевненості гіпотези виявляється в діапазоні [–0,2, +0,2], то пошук блокується), ранжирувати набір гіпотез після обробки всіх ознак.

Якщо обидві гіпотези підтверджують висновок або, навпаки, обидві гіпотези його спростовують, то коефіцієнт упевненості їхньої комбінації зростає за абсолютною величиною. Якщо ж одна гіпотеза підтверджує висновок, а інша його спростовує, то наявність знаменника у відповідному виразі згладжує цей ефект. Якщо виявилося, що гіпотез декілька, то їх можна по черзі пропускати через цю формулу, причому, оскільки вона має властивість комутативності, то порядок, у якому обробляються гіпотези, значення не має.

Незважаючи на відсутність строгого теоретичного обґрунтування, коефіцієнти впевненості знаходять широке застосування в експертних системах продукційного типу завдяки простоті сприйняття й інтерпретації одержуваних результатів, які непогано узгоджуються з реальністю.

Теорія Демпстера–Шафера розроблена з метою узагальнення імовірнісного підходу до опису невизначеності та пов'язана зі спробою звільнитися від догмата аксіом теорії ймовірностей при описі суб'єктивної віри людей.

Розглянемо *фрейм розрізнення* – кінцеву множину можливостей θ, які взаємно виключають одна одну.

На множині всіх підмножин θ як на множині елементарних подій задамо базисний розподіл ймовірностей *m*, визначений на множині 2^{θ} значень з інтервалу [0, 1], такий, що: $m(\varnothing) = 0$ та

$$\sum_{A_i \subseteq \theta} m(A_i) = 1,$$

де $m(A_i)$ – міра довіри, приписана гіпотезі A_i.

Міра загальної довіри, приписана *A*, визначається співвідношенням:

$$\mathrm{Bel}(A) = \sum_{B \subseteq A} m(B),$$

де $\mathrm{Bel}(A) \in [0, 1]$ – функція довіри, що характеризує віру суб'єкта в істинність події *A*, називана нижньою імовірністю.

Міра правдоподібності події *A*, називана верхньою імовірністю, визначається як: $Pl(A)=1–\mathrm{Bel}(\mathrm{not}(A))$.

Теорія Демпстера-Шафера заснована на двох ідеях. Перша – одержання ступеня довіри для даної задачі із суб'єктивних свідчень про пов'язані

з нею проблеми та друга – використання правила поєднання свідчень, якщо вони засновані на незалежних спостереженнях.

Правила Демпстера дозволяють обчислити нове значення функції довіри за двома її значеннями, що базуються на різних спостереженнях.

Метою правила є приписати деяку міру довіри m різним підмножинам A множини θ; m іноді називають імовірнісною функцією чутливості (probability density function) підмножини θ.

Реально свідчення підтримують не всі елементи θ. В основному підтримуються різні підмножини A множини θ. Більш того, оскільки елементи θ передбачаються як взаємовиключні, доказ на користь одного з них може впливати на довіру іншим елементам. У системі Демпстера-Шафера ці взаємодії враховуються прямо шляхом безпосереднього маніпулювання множинами гіпотез. Величина $m_n(A)$ означає ступінь довіри, пов'язаний з підмножиною гіпотез A, а n подає число джерел свідчень.

Правило Демпстера має вигляд:

$$m_n(A) = \frac{\sum_{X \cap Y = A} m_{n-2}(X) m_{n-1}(Y)}{1 - \sum_{X \cap Y = \varnothing} m_{n-2}(X) m_{n-1}(Y)}.$$

X та Y поширюються на всі підмножини в θ, перетинанням яких є A. Якщо в таблиці перетинань буде виявлено порожній елемент, то виконується нормалізація: визначається значення k як сума всіх ненульових значень, присвоєних у множені θ, потім установлюють $m_n(\varnothing) = 0$, а значення m_n для всіх інших множин гіпотез поділяються на $(1 - k)$.

Метод виведення Байєса заснований на припущенні про наявність практично для будь-якої події апріорних ймовірностей того, що дана подія є істинною. Задача полягає в зміні імовірнісних оцінок цієї події з появою інформації про настання деякої іншої події, називаних апостеріорними ймовірностями.

Апріорна імовірність події (безумовна імовірність, prior probability) – це імовірність, привласнена події при відсутності знання, що підтримує її настання, тобто це імовірність події, що передує якій-небудь основі. Апріорна імовірність позначається P(подія).

Апостеріорна імовірність події (умовна імовірність, posterior probability) – це імовірність події при деякій заданій основі. Вона позначається P(подія | основа).

Правило добутку пов'язує апостеріорні й апріорні імовірності:

$$P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A) .$$

Правило Байєса дозволяє обчислювати невідомі імовірності з відомих умовних ймовірностей, засновано на правилі добутку і має вигляд:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \text{ або } P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A)}.$$

Теорема Байєса в загальному виді дозволяє визначити апостеріорну імовірність, залежить від апріорної імовірності і від інформації, що надійшла:

$$P(A_i \mid B) = \frac{P(B \mid A_i)P(A_i)}{\sum_{j=1}^{N} P(B \mid A_j)P(A_j)}, i = 1, 2, ..., N,$$

де $\{A_i\}$ – послідовність непересічних подій (гіпотез), B – довільна подія (симптом), для якої $P(B) > 0$, N – число можливих гіпотез, $P(A_i|B)$ – апостеріорна імовірність гіпотези A_i за наявності симптому B; $P(B|A_i)$ – імовірність появи симптому B за наявності гіпотези A_i; $P(A_i)$ – апріорна імовірність гіпотези A_i. Імовірності $P(B|A_i)$ та $P(A_i)$, $i = 1, ..., N$, задаються експертом і не змінюються в процесі вирішення задачі.

Метод виведення Нейлора полягає в приписуванні кожному симптому ціни, що відбиває роль у процесі виведення, і задавання в першу чергу того питання, для якого ціна виявляється найбільшою.

Судження проводяться за такою послідовністю кроків.

Крок 1. Оцінити апріорні імовірності $P(A_i)$ для всіх гіпотез.

Крок 2. Обчислити ціни симптомів.

Ціна симптому визначається за формулою:

$$C(B_j) = \sum_{i=1}^{N} \left| P(A_i \mid B_j) - P(A_i \mid B_j) \right|,$$

$$P(A_i \mid B_j) = \frac{P(B_j \mid A_i)P(A_i)}{P(B_j)}, \quad P(A_i \mid B_j) = \frac{1 - P(B_j \mid A_i)P(A_i)}{1 - P(B_j)},$$

$$P(A_i) = P(B_j|A_i)P(A_i) + P(B_j|\neg A_i)P(\neg A_i),\ P(\neg B_j) = 1 - P(B_j),$$

де $C(B_j)$ – ціна симптому B_j – сума максимальних змін ймовірностей подій, що можуть відбутися у всіх N гіпотезах, до яких цей симптом може бути застосовним.

Крок 3. Знайти симптом з максимальною ціною:

$$B_m: m = \arg \max_{j=1,2,...,N} C(B_j).$$

Крок 4. Поставити запитання користувачу для симптому B_m і одержати на нього відповідь R_m, $R_m \in \{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$, де –5 відповідає «ні», 0 – «не знаю», а 5 – «так».

Крок 5. Обчислити апостеріорні імовірності для актуальних гіпотез:

$$P(A_i \mid R_m) = \begin{cases} 0{,}2R_m(P(A_i \mid B_m) - P(A_i)) + P(A_i), 0 \le R_m \le 5; \\ -0{,}2R_m(P(A_i \mid \neg B_m) - P(A_i)) + P(A_i), -5 \le R_m \le 5. \end{cases}$$

Установити: $P(A_i) = P(A_i|R_m)$.

Крок 6. Перерахувати $C(B_j)$ для нових значень $P(A_i)$.

Крок 7. Для кожної гіпотези A_i знайти значення поточної мінімальної імовірності гіпотези $P_{\min}(A_i)$ і поточної максимальної імовірності гіпотези $P_{\max}(A_i)$.

Крок 8. Знайти найбільший з можливих досяжних максимумів ймовірностей для всіх гіпотез:

$$PM = \max_{i=1,2,\ldots,N} P_{\min}(A_i).$$

Крок 9. Якщо існує такий номер k, для якого $P_{\max}(A_k) > PM$, то перейти до кроку 3; у противному випадку – вибрати гіпотези:

$$A_m : m = \arg \max_{i=1,2,\ldots,N} P(A_i),$$

як найбільш ймовірний результат.

Крок 10. Видати як результат гіпотези A_m і завершити роботу.

1.5.4 Пошук у просторі станів

При проектуванні інтелектуальної системи серйозну увагу має бути приділено тому, як здійснюється доступ до знань і як вони використовуються при пошуку рішення.

Пошук – процес формування системою послідовності дій, що дозволяють їй досягти своїх цілей. Перш ніж система зможе приступити до пошуку рішень, повинна бути сформульована мета, а потім ця мета може використовуватися для формулювання задачі.

Процес вирішення задач пошуку рішень в інтелектуальних системах, заснованих на знаннях, як правило, являє собою перебір досить великої кількості різних варіантів рішень, кожне з яких можна зіставити деякому стану.

Стан – це колекція характеристик, що можуть використовуватися для визначення стану або статусу об’єкта.

Простір станів – це множина станів, за допомогою якої подаються переходи між станами, у яких може бути об’єкт. У результаті переходу об’єкт попадає з одного стану до іншого.

Простір станів зручно подавати у вигляді гіперграфа.

Гіперграф складається з множини вершин (вузлів) N і множини гіпердуг H.

Вузол – це облікова структура даних, застосовувана для подання дерева пошуку. Кожен вузол має батьківський вузол, містить дані про стан і має різні допоміжні поля.

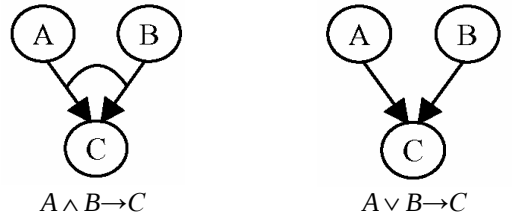
Листовий вузол – вузол, що не має нащадків у орієнтованому графі.

Периферія – колекція вузлів, що були сформовані, але ще не розгорнуті. Кожен елемент периферії являє собою лист.

Гіпердуга задається упорядкованою парою, у якій перший елемент є окремою вершиною з N, а другий – підмножиною множини N. Гіпердуги також

називають *k-коннекторами*, де *k* – потужність множини породжених вершин.

Граф ТА/АБО (And/Or-Graph) – окремий випадок гіперграфа, у якому вузли з'єднані не окремими дугами, а множиною дуг. Щоб подати різні відношення графічно, на графах ТА/АБО розрізняють вузли: ТА (and) і АБО (or). (рис. 1.5).



$A \wedge B \rightarrow C$ $A \vee B \rightarrow C$

Рисунок 1.5 – ТА/АБО-графи

ТА-вузли графа – вузли, що подають імплікацію передумов, зв'язаних оператором ТА. Дуги, що ведуть до цих вузлів, з'єднуються кривою. Крива, що з'єднує дуги, означає, що для доказу вузла повинні бути істинними усі його передумови, з'єднані дугою.

АБО-вузли графа – вузли, що подають імплікацію передумов, зв'язаних оператором АБО. Дуги, які ведуть до цих вузлів, не з'єднуються кривою, що вказує на те, що істина кожної з передумов є достатньою умовою для істинності висновку.

Задачі пошуку в просторі станів можна сформулювати в термінах трьох найважливіших компонентів:

– *вихідний стан проблеми*;

– *тест завершення* — перевірка, чи досягнуто необхідний кінцевий стан або знайдено рішення проблеми;

– множина операцій, які можна використовувати для зміни поточного стану проблеми.

Простір пошуку може бути визначено такими під-аспектами:

– *розміром простору пошуку* – дає узагальнену характеристику складності задачі. Виділяють малі (до 10! станів) і великі (понад 10! станів) простори пошуку;

– *глибиною простору пошуку* – характеризується середнім числом послідовно застосовуваних правил, що перетворюють вихідні дані в кінцевий результат;

– *шириною простору пошуку* – середнім числом правил, придатних до виконання в поточному стані.

Стратегія пошуку в просторі станів – порядок, у якому відбувається розгортання станів. Стратегія пошуку повинна бути виражена у вигляді функції, що вибирає певним чином з периферії наступний вузол, який підлягає розгортанню. Хоча даний підхід концептуально є нескладним, він може ви-

явитися дорогим з обчислювальної точки зору, оскільки функцію, передбачену в цій стратегії, можливо, прийдеться застосувати до кожного елемента в зазначеній множині для вибору найкращого з них. Тому часто передбачається, що колекцію вузлів реалізовано у вигляді черги.

Стратегія прямого пошуку (forward chaining) – відповідає руху від вихідних вершин графа до цільової вершини.

Стратегія зворотного пошуку (backward chaining) – відповідає руху від цільової вершини до вихідних вершин. Якщо вершин-цілей мало, а вихідних багато, то зворотний пошук є більш природним і ефективним.

Стратегія двонаправленого пошуку (bi-directional chaining) – поєднує прямий пошук (рух від вихідних вершин до цільової) і зворотний пошук (рух від цільової вершини до вихідної) та намагається досягти деякого загального для обох пошуків стану, зупиняючись після того, як два процеси пошуку зустрінуться на середині. У стратегії двонаправленого пошуку передбачається перевірка в одному чи в обох процесах пошуку кожного вузла перед його розгортанням для визначення того, чи не знаходиться він на периферії іншого дерева пошуку; у випадку позитивного результату перевірки вважається, що рішення знайдено. Перевірка приналежності вузла до іншого дерева пошуку може бути виконана за постійний час за допомогою хеш-таблиці.

Часова складність двонаправленого пошуку визначається як $O(b^{d/2})$, де b – коефіцієнт розгалуження, а d – глибина дерева пошуку. У пам'яті необхідно зберігати принаймні одне з дерев пошуку, для того, щоб можна було виконати перевірку приналежності до іншого дерева, тому просторова складність також визначається як $O(b^{d/2})$. Такі вимоги до простору є одним з найбільш істотних недоліків двонаправленого пошуку. Цей метод є повним при кінцевому коефіцієнті розгалуження b і оптимальним при однакових вартостях етапів, якщо обидва процеси пошуку здійснюються в ширину; інші сполучення методів можуть характеризуватися відсутністю повноти, або оптимальності того й іншого.

Схема пошуку на ТА/АБО-графі – спосіб руху по графу в напрямку, заданому стратегією пошуку. Розрізняють схеми *сліпого (неінформованого)* і *спрямованого (інформованого) пошуків* на графі, пов'язані з перебором альтернативних вершин-підцілей і організацією повернення.

На практиці сліпі методи пошуку використовуються рідко, оскільки вони пов'язані з великими витратами ресурсів через комбінаторний вибух. Спрямовані методи пошуку використовують як оцінювання придатних альтернатив, так і точок повернення, тобто забезпечують керування поверненням.

Після вибору стратегії пошуку (на основі даних або від мети), оцінки графа і вибору методу пошуку подальший хід рішення буде залежати від конкретної задачі.

Результатом застосування будь-якого методу рішення задачі є або невдале завершення, або одержання рішення. Деякі методи можуть входити в нескінченний цикл і не повертати ніякого результату.

Методи пошуку на графах використовують ідеї пошуку з поверненнями.

Метод пошуку з поверненнями (backtracking) – це метод систематичної перевірки різних шляхів у просторі станів при якому формується список недосліджених станів Open, для того щоб мати можливість повернутися до кожного з них; підтримується список переглянутих «невдалих» станів Closed, щоб відгородити метод від перевірки марних шляхів; підтримується список вузлів поточного шляху, що повертається по досягненні мети; кожен новий стан перевіряється на входження в ці списки, щоб запобігти зацикленню.

Пошук з поверненнями запускається з початкового стану і проводиться по деякому шляху доти, поки не досягне мети або не упреться в тупик. Якщо ціль досягнуто, пошук завершується, і як рішення задачі повертається шлях до цілі. Якщо ж пошук привів у тупикову вершину, то метод повертається в найближчу з пройдених вершин і досліджує всі її вершини-сестри, а потім спускається по одній з гілок, що ведуть від вершини-сестри. Цей процес описується таким рекурсивним правилом.

Якщо вихідний стан не задовольняє вимогам цілі, то зі списку його нащадків вибираємо першого нащадка, і до цієї вершини рекурсивно застосовуємо процедуру пошуку з поверненнями. Якщо в результаті пошуку з поверненнями в підграфі з коренем першого нащадка ціль не виявлено, то повторюємо процедуру для вершини-сестри – другого нащадка вихідного стану. Ця процедура продовжується доти, поки один з нащадків розглянутої вершини-сестри не виявиться цільовим вузлом, або не з'ясується, що розглянуто уже всіх можливих нащадків (усіх вершин-сестер). Якщо ж жодна з вершин-сестер вихідної вершини не привела до цілі, то повертаємося до предка вихідної вершини і повторюємо процедуру з вершиною-сестрою вихідної вершині і т. д.

Продуктивність методів пошуку оцінюють за допомогою таких показників:

– *повнота* – визначає гарантію виявлення методом рішення, якщо воно існує;

– *оптимальність* – властивість забезпечення методом знаходження оптимального рішення;

– *часова складність* – оцінка часу, за який метод знаходить рішення;

– *просторова складність* – оцінка обсягу пам'яті, необхідного для здійснення пошуку.

Часова і просторова складність завжди аналізуються з урахуванням визначеного критерію виміру складності задачі. Типовим критерієм для комп'ютерних наук є розмір графа простору станів, оскільки цей граф розглядається як явно задана структура даних, що є вхідною для програми пошуку. У штучному інтелекті, де граф подається неявно за допомогою початкового стану і функції визначення нащадка і часто є нескінченним, складність виражається в термінах трьох величин: b – коефіцієнт розгалуження або максимальна кількість нащадків будь-якого вузла, d – глибина найповерхневого цільового вузла та m – максимальна довжина будь-якого шляху в просторі станів. Часова складність часто виміряється в термінах

кількості вузлів, вироблюваних у процесі пошуку, а просторова складність – у термінах максимальної кількості вузлів, збережених у пам'яті.

Пошуковий метод є припустимим, якщо для будь-якого графа він завжди вибирає оптимальний шлях до рішення.

Ефективність методів пошуку визначається вартістю пошуку, що звичайно залежить від часової складності, але може також включати вираз для оцінки використання пам'яті, або сумарною вартістю, у якій поєднуються вартість пошуку і вартість шляху знайденого рішення.

Стратегії неінформованого (сліпого) пошуку – стратегії, що не використовують додаткову інформацію про стани, крім тієї, котра подана у визначенні задачі. Вони здатні тільки виробляти нові стани-нащадки і відрізняти цільовий стан від нецільового.

Метод породження і перевірки (generate-and-test).

Крок 1. Генерувати новий стан, модифікуючи існуючий.

Крок 2. Перевірити, чи не є стан кінцевим (рішенням). Якщо це так, то завершити роботу, інакше перейти до кроку 1.

Метод породження і перевірки має два основних варіанти: пошук у глибину та пошук у ширину. Відрізняються варіанти порядком формування станів на кроці 1.

Метод пошуку в ширину (*breadth-first search*) – це стратегія, у якій простір станів послідовно проглядається по рівнях: на кожному рівні, у свою чергу, послідовно проглядаються стани, подані вузлами цього рівня один за іншим, і тільки якщо станів на даному рівні більше немає, метод переходить до наступного рівню (див. рис. 1.6 – пунктиром показаний порядок перегляду вузлів).

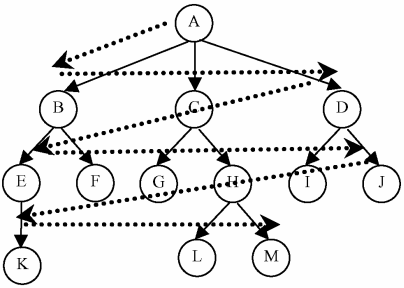


Рисунок 1.6 – Граф, що демонструє роботу методу пошуку в ширину

Нехай *s* – вузол початкового стану, Open – список, що містить обрані, але необроблені вузли; Closed – список, що містить оброблені вузли. Тоді метод пошуку в ширину буде полягати у виконанні таких кроків.

Крок 1. Ініціалізація. Установити: Open = {*s*}, Closed = ∅.

Крок 2. Якщо Open=∅, то припинити виконання – шляху до цільового стану на графі не існує; у противному випадку – перейти до кроку 3.

Крок 3. Видалити з Open крайній ліворуч стан *x*.

Крок 4. Якщо *x* – ціль, то закінчити пошук і сформувати результат – шлях, породжений простежуванням покажчиків від вузла *x* до вузла *s*; у противному випадку – виконати кроки 4.1–4.4.

Крок 4.1 Згенерувати стан-нащадок *x*.

Крок 4.2 Помістити *x* у список Closed.

Крок 4.3 Виконати перевірку на цикл – виключити нащадок *x*, якщо він вже є в списку Open або Closed.

Крок 4.4 Помістити інші нащадки в правий кінець списку Open, створюючи в такий спосіб чергу.

Крок 5. Перейти до кроку 2.

На кожній ітерації методу пошуку в ширину генеруються всі дочірні вершини стану *x* і записуються в список Open, що діє як черга й обробляє дані в порядку надходження: стани додаються в список праворуч, а видаляються ліворуч. У такий спосіб у пошуку беруть участь стани, що знаходяться в списку Open довше всього, забезпечуючи пошук у ширину. Дочірні стани, що були вже записані в списки Open або Closed, відкидаються.

Коли ціль знайдено, метод може відновити шлях рішення, прослідкуючи його в зворотному напрямку від мети до початкового стану по батьківських станах. Оскільки пошук у ширину знаходить кожний стан за найкоротшим шляхом і зберігає першу версію кожного стану, цей шлях від початку до цілі є найкоротшим. Методи, що мають таку властивість, називають *розв'язними* (admissible).

Пошук у ширину є повним, оптимальним при однакових вартостях етапів і характеризується часовою і просторовою складністю $O(b^{d+1})$, де b – кількість вузлів на рівні, d – кількість рівнів. У зв'язку з такою просторовою складністю в більшості випадків він стає практично не застосовним. При пошуку в ширину найбільш складною проблемою в порівнянні зі значним часом виконання є забезпечення потреб у пам'яті. Узагалі говорячи, задачі пошуку з експонентною складністю неможливо вирішити за допомогою неінформованих методів у всіх екземплярах цих задач, крім самих невеликих.

Метод пошуку за критерієм вартості – модифікація пошуку в ширину, що є оптимальною при будь-якій функції визначення вартості етапу. При пошуку за критерієм вартості враховується не кількість етапів, що є у шляху, а тільки їхня сумарна вартість. Якщо вартості всіх етапів є рівними, такий пошук ідентичний пошуку в ширину. Він є повним, якщо коефіцієнт розгалуження b є кінцевим, і оптимальним, якщо вартість кожного кроку перевищує деяке позитивне граничне значення e – ця умова означає, що вартість шляху завжди зростає в міру проходження цим шляхом.

Метод розгортає вузли в порядку зростання вартості шляху, віддаючи

пріоритет розгортанню вузла з найменшою вартістю шляху. Тому перший цільовий вузол, обраний для розгортання, являє собою оптимальне рішення.

Пошук за критерієм вартості направляється з урахуванням вартостей шляхів, а не значень глибини в дереві пошуку, тому його складність не може бути легко охарактеризована в термінах *b* і *d*.

Часова і просторова складність цього методу в найгіршому випадку складає $O(b^{\lfloor 1+C^*/e \rfloor})$, тобто може бути набагато більшою, ніж b^d, де C^* – вартість оптимального рішення, *e* – вартість кожної дії. Це пов'язано з тим, що процедури пошуку за критерієм вартості можуть і часто виконують перевірку великих дерев, які складаються з дрібних етапів, перш ніж перейти до дослідження шляхів, у котрі входять великі, але, можливо, більш корисні етапи. Безумовно, якщо усі вартості етапів є рівними, то $O(b^{\lfloor 1+C^*/e \rfloor}) = b^d$.

Метод пошуку в глибину (depth-first search) – це стратегія, у якій переглядаються стани на одному шляху – розгортається найглибший вузол у поточній периферії дерева пошуку, у результаті чого пошук безпосередньо переходить на найглибший рівень дерева пошуку, на якому вузли не мають нащадків. У міру того, як ці вузли розгортаються, вони видаляються з периферії, тому надалі пошук переходить до наступного найповерхневого вузлу, що усе ще має недосліджених нащадків (див. рис. 1.7 – пунктиром показаний порядок перегляду вузлів).

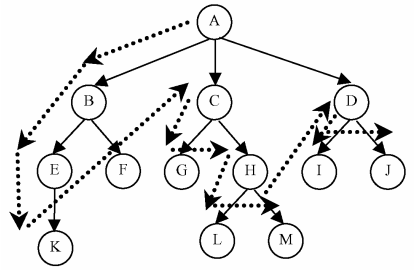


Рисунок 1.7 – Граф, що демонструє роботу методу пошуку в глибину

Нехай *s* – вузол початкового стану, Open – список, що містить обрані, але необроблені вузли; Closed – список, що містить оброблені вузли. Тоді метод пошуку в глибину буде полягати у виконанні таких кроків.

Крок 1. Ініціалізація. Установити: Open = {*s*}, Closed = ∅.

Крок 2. Якщо Open=∅, то припинити виконання – шляху до цільового стану на графі не існує; у противному випадку – перейти до кроку 3.

Крок 3. Видалити з Open крайній ліворуч стан *x*.

Крок 4. Якщо *x* – ціль, то закінчити пошук і сформувати результат –

шлях, породжений простежуванням покажчиків від вузла *x* до вузла *s*; у противному випадку – виконати кроки 4.1–4.4.

Крок 4.1 Згенерувати стан-нащадок *x*.

Крок 4.2 Помістити *x* у список Closed.

Крок 4.3 Виконати перевірку на цикл – виключити нащадок *x*, якщо він вже є в списку Open або Closed.

Крок 4.4 Помістити інші нащадки в лівий кінець списку Open, створюючи в такий спосіб стек.

Крок 5. Перейти до кроку 2.

При пошуку в глибину після дослідження стану спочатку необхідно оцінити всіх його нащадків та їхніх нащадків, а потім досліджувати кожну з вершин-сестер. Пошук у глибину за можливістю поглиблюється в область пошуку. Якщо подальші нащадки стану не знайдені, розглядаються вершини-сестри. У цьому методі стани-нащадки додаються і видаляються з лівого кінця списку Open, тобто список Open реалізований як стек магазинного типу. При організації списку Open у вигляді стека перевага віддається самим «молодим» (нещодавно згенерованим станам).

Пошук у глибину не є ані повним, ані оптимальним, і характеризується часовою складністю $O(b^m)$ і просторовою складністю $O(bm)$, де b – коефіцієнт розгалуження, m – максимальна глибина будь-якого шляху в просторі станів.

Пошук у глибину швидко проникає в глибини простору. Якщо відомо, що шлях рішення буде довгим, то пошук у глибину не буде витрачати час на пошук великої кількості поверхневих станів на графі.

Пошук у глибину ефективний для областей пошуку з високим ступенем зв'язаності, оскільки йому не потрібно запам'ятовувати усі вузли даного рівня в списку Open.

Недоліком пошуку в глибину є те, що в ньому може бути зроблено неправильний вибір і перехід у тупикову ситуацію, пов'язану з проходженням униз по дуже довгому (чи навіть нескінченному) шляху, при тому, що інший варіант міг би привести до рішення, яке знаходиться недалеко від кореня дерева пошуку.

Метод пошуку з поверненнями – один з варіантів пошуку в глибину, що використовує менше пам'яті. При пошуку з поверненнями щоразу формується тільки один нащадок, а не всі нащадки; у кожному частково розгорнутому вузлі запам'ятовується інформація про те, який нащадок має бути сформований наступним. Таким чином, потрібно тільки $O(m)$ пам'яті, а не $O(bm)$. При пошуку з поверненнями застосовується ще один прийом, що дозволяє заощаджувати пам'ять (і час); ідея його полягає в тому, що при формуванні нащадка повинний безпосередньо модифікуватися опис поточного стану, а не здійснюватися його попереднє копіювання. При цьому потреба в пам'яті скорочується до обсягу, необхідного для збереження тільки одного опису стану і $O(m)$ дій. Але для успішного застосування даного прийому потрібно мати можливість скасовувати кожну мо-

дифікацію при поверненні, виконуваному для формування наступного нащадка. При вирішенні задач з об'ємними описами стану застосування зазначених методів модифікації станів стає найважливішим фактором успіху.

Метод пошуку з обмеженням глибини – проста модифікація рекурсивного методу пошуку в глибину. При пошуку з обмеженням глибини на пошук у глибину накладається встановлена межа глибини l. Це означає, що вузли на глибині l розглядаються таким чином, якби вони не мали нащадків.

Функція RDLS(вузол node, межа глибини l) повертає рішення result або індикатор невдачі: failure (указує на відсутність рішення), cutoff (свідчить про те, що на заданій межі глибини рішення немає).

Крок 1. Установити прапор відсутності рішення на поточній глибині: cutoff_occurred = хибність.

Крок 2. Якщо вузол node є цільовим, то зупинення, повернути як рішення вузол node.

Крок 3. Якщо глибина вузла node дорівнює l, то зупинення, повернути як рішення індикатор невдачі cutoff.

Крок 4. Згенерувати нащадків вузла node. Для кожного нащадка successor вузла node виконувати кроки 4.1–4.2.

Крок 4.1 Установити: result = RDLS(successor, l).

Крок 4.2 Якщо result = cutoff, то установити: cutoff_occurred = істина, у противному випадку – якщо result ≠ failure, то зупинення, повернути як рішення result.

Крок 5. Якщо cutoff_occurred = істина, то зупинення, повернути індикатор невдачі cutoff; у противному випадку – зупинення, повернути індикатор невдачі failure.

Початковий виклик функції: RDLS(початковий стан node, l).

Застосування межі глибини дозволяє вирішити проблему нескінченного шляху. На жаль, у цьому підході також вводиться додаткове джерело неповноти, якщо буде обране значення $l < d$, іншими словами, якщо найповерхнева ціль виходить за межі глибини. Така ситуація цілком ймовірна, якщо значення d невідомо. Крім того, пошук з обмеженням глибини буде неоптимальним при виборі значення $l > d$. Його часова складність дорівнює $O(b^l)$, а просторова складність – $O(bl)$. Пошук у глибину може розглядатися як окремий випадок пошуку з обмеженням глибини, при якому $l = \infty$.

Метод пошуку в глибину з ітеративним поглибленням – загальна стратегія, що дозволяє знайти найкращу межу глибини. Це досягається шляхом поступового збільшення межі глибини від 0 із кроком 1 доти, поки не буде знайдено ціль. Така подія відбувається після того, як межа глибини досягає значення d – глибини самого поверхневого цільового вузла. На кожній ітерації метод виконує пошук у глибину з урахуванням поточного граничного числа рівнів. При цьому при переході від однієї ітерації до іншої інформація про простір станів не зберігається. Метод припускає виконання таких кроків.

Крок 1. Установити поточний рівень глибини: $d = 0$.

Крок 2. Установити: result = RDLS(початковий вузол, d).

Крок 3. Якщо result ≠ cutoff, то зупинення, повернути як рішення result.

Крок 4. Установити: $d = d+1$. Перейти до кроку 2.

Пошук з ітеративним поглибленням повторно застосовує пошук з обмеженням глибини, реалізований функцією RDLS, при послідовному збільшенні меж. Він завершує свою роботу після того, як виявляється рішення, або функція пошуку з обмеженням глибини повертає значення failure, а це означає, що рішення не існує.

Цей пошук є повним при кінцевому коефіцієнті розгалуження b і оптимальним при однакових вартостях етапів і характеризується часовою складністю $O(b^d)$ і просторовою складністю $O(bd)$.

У пошуку з ітеративним поглибленням поєднуються переваги пошуку в глибину і пошуку в ширину. Як і пошук у глибину, він характеризується дуже скромними вимогами до пам'яті. Як і пошук у ширину, він є повним, якщо коефіцієнт розгалуження кінцевий, і оптимальним, якщо вартість шляху являє собою неубутну функцію глибини вузла.

Ітеративне поглиблення є кращим методом неінформованого пошуку за тих умов, коли є великий простір пошуку, а глибина рішення є невідомою.

Оскільки сліпий пошук можливий тільки в невеликому просторі варіантів, є необхідним деякий спосіб спрямованого пошуку.

Стратегії інформованого (евристичного) пошуку – стратегії, що дозволяють визначити, чи є один нецільовий стан більш перспективним у порівнянні з іншим. Ці стратегії використовують при пошуку шляху на графі в просторі станів деякі знання, специфічні для конкретної предметної області.

Найкраще розглядати евристику як деяке правило впливу, що, хоча і не гарантує успіху (як детермінований метод або процедура прийняття рішення), у більшості випадків виявляється дуже корисним.

У загальному вигляді евристичний пошук можна подати як послідовність таких етапів.

Етап 1. Обрати з множини можливих дій деяку дію:

- з урахуванням відповідності до цілі:
  - зменшення деякого небажаного розходження,
  - безпосереднє вирішення тієї чи іншої підзадачі;
- з урахуванням досвіду:
  - повторення минулого,
  - виявлення ключової дії;
- з урахуванням необхідних умов:
  - рішення, обумовлене аналізом ситуації,
  - виключення нездійсненного варіанта;
- з урахуванням фактора випадковості:

– перевага віддається розмаїтості.

Етап 2. Здійснити обрану дію і змінити поточну ситуацію.

Етап 3. Оцінити ситуацію:

- за аналогією:
  – відома сама задача,
  – відома підзадача;
- за величиною відстані до цілі:
  – відстань між двома ситуаціями,
  – кількість зусиль, затрачуваних на пошук;
- за математичним критерієм:
  – складання переліку необхідних та / або достатніх для одержання даного рішення характеристик,
  – чисельна оцінна функція,
  – верхні і нижні межі,
  – сума вартостей, обраних придатним способом;
- за очікуваним виграшем (критерій, пов'язаний з минулим досвідом):
  – простота ситуації,
  – коефіцієнт розширення пошуку,
  – інший критерій (складність задачі, затрачуваний на її рішення час і т. д.).

Етап 4. Відкинути некорисні ситуації.

Етап 5. Якщо досягнуто кінцеву ситуацію – кінець; у противному випадку – вибрати нову, вихідну ситуацію і почати спочатку:

- рухатися тільки вперед:
  – систематичний розвиток останньої породженої ситуації;
- виконувати всі дії паралельно:
  – почергове виконання всіх дій;
- як вихідну обирати найбагатообіцяючу ситуацію:
  – у відношенні оцінної функції,
  – у відношенні незначного числа вхідних у неї дій;
- йти на компроміс між:
  – глибиною пошуку,
  – оцінкою ситуації.

Метод пошуку зі сходженням до вершини (пошук екстремуму, hill climbing) – проста форма евристичного пошуку, що є фундаментальним методом локального пошуку. У процесі пошуку використовується певна *оцінна функція*, за допомогою якої можна грубо оцінити, наскільки гарним або поганим є поточний стан. При цьому оцінюють не тільки поточний стан пошуку, але і його нащадків. Для подальшого пошуку вибирається найкращий нащадок; при цьому про його братів і батьків просто забувають. Пошук припиняється, коли досягається стан, що краще ніж кожний з його нащадків.

Метод сходження до вершини, можна сформулювати в такий спосіб.

Крок 1. Знаходячись у даній точці простору станів, застосувати правила породження нової множини можливих рішень.

Крок 2. Якщо один з нових станів є рішенням проблеми, припинити процес. У противному випадку перейти в той стан, що характеризується найвищим значенням оцінної функції. Повернутися до кроку 1.

Пошук зі сходженням до вершини іноді називають *жадібним локальним пошуком*, оскільки в процесі його виконання відбувається захоплення самого гарного сусіднього стану без попередніх міркувань про те, куди варто відправитися далі. Якщо кількість найкращих нащадків вузла більше одного, то метод звичайно передбачає випадковий вибір нащадка серед них.

У методі зі сходженням до вершини не здійснюється прогнозування за межами станів, що є безпосередньо сусідніми стосовно поточного стану.

Жадібні методи часто показують дуже високу продуктивність, але застосування цієї стратегії наштовхується на такі труднощі:

– необхідність формулювання оцінної функції, яка б адекватно відбивала якість поточного стану;

– тенденція зупинятися в *локальному максимумі* – рішенні, з якого можливий тільки спуск, тобто погіршення стану. Як тільки метод досягає стану, що має кращу оцінку, ніж його нащадки, він зупиняється. Якщо цей стан не є рішенням задачі, а тільки локальним максимумом, то такий метод неприйнятний для даної задачі. Це значить, що рішення може бути оптимальним на обмеженій множині, але через форму всього простору, можливо, ніколи не буде обране найкраще рішення.

Оскільки в цій стратегії дані про попередні стани не зберігаються, то вона без механізмів повернення або інших прийомів відновлення не може відрізнити локальний максимум від глобального, у наслідок чого метод не може бути відновлений із точки, що призвела до невдачі.

Існують методи наближеного вирішення цієї проблеми, наприклад, випадкове збурювання оцінки. Однак гарантовано вирішувати задачі з використанням даної стратегії пошуку не можна.

Незважаючи на ці обмеження, метод пошуку зі сходженням до вершини може бути досить ефективним, якщо оцінна функція дозволяє уникнути локального максимуму і зациклення методу.

Метод пошуку за першим найкращим збігом (жадібний пошук, best-first search) – форма евристичного пошуку, що використовує оцінну функцію, за допомогою якої можна порівнювати стани в просторі станів.

Кожному стану n відповідає значення оцінної функції $f = g(n) + h(n)$, де $g(n)$ – глибина стану n у просторі пошуку, оцінка якого утримує метод від завзятого проходження по невірному шляху, $h(n)$ – деяка евристична оцінка відстані від стану n до цілі, що веде метод пошуку до евристично найбільш перспективних станів.

Метод порівнює не тільки ті стани, у які можливий перехід з поточного

стану, але і всі, до яких можна дістати. Такий метод вимагає великих обчислювальних ресурсів, але його ідея полягає в тому, щоб брати до уваги не тільки найближчі стани, тобто локальну обстановку, а переглянути як можна більшу ділянку простору станів і бути готовим, у разі потреби, повернутися туди, де він уже був, і піти іншим шляхом, якщо найближчі претенденти не обіцяють істотного прогресу в досягненні мети. Саме ця можливість відмовитися від частини пройденого шляху заради глобальної мети і дозволяє знайти більш ефективний шлях.

Метод використовує списки збережених станів: список Open відслідковує поточний стан пошуку, а в Closed записуються вже перевірені стани. На кожному кроці метод записує в список Open стан з урахуванням деякої евристичної оцінки його близькості до цілі. Таким чином, на кожній ітерації розглядаються найбільш перспективні стани зі списку Open. Стани в списку Open сортуються у відповідності зі значеннями оцінної функції *f*. Зберігаючи всі стани в списку Open, метод зможе уберегти себе від невдалого завершення. У жадібному методі пошуку при використанні пріоритетної черги можливе відновлення точки локального максимуму.

Метод припускає виконання таких кроків.

Крок 1. Установити: Open = {*S*}, Closed = ∅, де *S* – початковий стан.

Крок 2. Якщо Open = ∅, то перейти до кроку 9, у противному випадку – перейти до кроку 3.

Крок 3. Видалити перший стан *X* зі списку Open.

Крок 4. Якщо *X* – цільовий стан, то зупинення, повернути шлях від *S* до *X*; у противному випадку – перейти до кроку 5.

Крок 5. Згенерувати нащадків *X*.

Крок 6. Для кожного нащадка *X* виконувати кроки 6.1–6.3.

Крок 6.1 Якщо нащадок не міститься в списках Open та Closed, то зіставити нащадку евристичне значення і додати нащадка в список Open.

Крок 6.2 Якщо нащадок міститься в списку Open і нащадок був досягнутий по найкоротшому шляху, то записати цей стан у список Open.

Крок 6.3 Якщо нащадок міститься в списку Closed і нащадок був досягнутий по найкоротшому шляху, то видалити стан зі списку Closed і додати нащадок у список Open.

Крок 7. Помістити *X* у список Closed.

Крок 8. Переупорядкувати стани в списку Open відповідно до евристики «кращі ліворуч».

Крок 9. Зупинення. Повернути відмову – список Open порожній.

На кожній ітерації метод видаляє перший елемент зі списку Open. Досягнувши мети, він повертає шлях, що веде до рішення. Помітимо, що кожен стан зберігає інформацію про попередній стан, щоб згодом відновити його і дозволити методу знайти найкоротший шлях до рішення.

Якщо перший елемент у списку Open не є рішенням, то метод використовує

правила перевірки відповідності й операції, щоб згенерувати всіх можливих нащадків даного елемента. Якщо нащадок уже знаходиться в списку Open або Closed, то метод вибирає найкоротший із двох можливих шляхів досягнення цього стану.

Потім метод обчислює евристичну оцінку станів у списку Open і сортує список відповідно до цих евристичних значень. При цьому кращі стани ставляться на початок списку. Помітимо, що через евристичну природу оцінювання наступний стан повинний перевірятися на будь-якому рівні простору станів. Відсортований список Open часто називають *пріоритетною чергою* (priority queue).

На відміну від пошуку екстремуму, що не зберігає пріоритетну чергу для відбору наступних станів, даний метод відновлюється після помилки і знаходить цільовий стан.

Компонент $g(n)$ функції оцінки додає жадібному пошуку властивості пошуку в ширину. Він запобігає можливості заблукання через помилкову оцінку: якщо евристика безупинно повертає гарні оцінки для станів на шляху, що не веде до мети, то значення g буде рости і домінувати над h, повертаючи процедуру пошуку до найкоротшого шляху. Це рятує метод від зациклення.

Пошук за першим найкращим збігом нагадує пошук у глибину в тому відношенні, що цей метод надає переваги на шляху до мети постійному крокуванню по єдиному шляху, але метод повертається до попередніх вузлів після влучення в тупик. Даний метод страждає від тих же недоліків, що і пошук у глибину: він не є оптимальним, до того ж він – неповний (оскільки здатний відправитися по нескінченному шляху та так і не повернутися, щоб випробувати інші можливості). При цьому в найгіршому випадку оцінки часової і просторової складності складають $O(b^m)$, де m – максимальна глибина простору пошуку. Однак гарна евристична функція дозволяє істотно скоротити таку складність. Величина цього скорочення залежить від конкретної задачі і від якості евристичної функції.

Метод рекурсивного пошуку за першим найкращим збігом (Recursive Best-First Search – RBFS) – простий рекурсивний метод, у якому робляться спроби імітувати роботу стандартного пошуку за першим найкращим збігом, але з використанням тільки лінійного простору. Метод може бути описаний як така функція.

Функція RBFS (вузол node, межа f-вартості f_{limit}) – повертає рішення result або індикатор невдачі «відмова» і нову межу f-вартості f_{limit}.

Крок 1. Якщо вузол node – цільовий, то повернути result = node.

Крок 2. Сформувати для вузла node множину вузлів-нащадків Successors.

Крок 3. Якщо Successors = ∅, то повернути: result = «відмова» та $f_{\text{limit}} = \infty$.

Крок 4. Для кожного вузла s у Successors установити: $f(s)=\max(g(s)+h(s), f(\text{node}))$.

Крок 5. Установити: best = вузол з найменшим f-значенням у множині

Successors.

Крок 6. Якщо f(best) > f_{limit}, то повернути result = «відмова» та $f_{\text{limit}} = f$(best).

Крок 7. Знайти alternative – друге після найменшого f-значення для елементів у множині Successors.

Крок 8. Установити: result, f(best) = RBFS (best, min(f_{limit}, alternative)).

Крок 9. Якщо result ≠ «відмова», то повернути result.

Перший виклик функції: RBFS (node = початковий вузол, $f_{\text{limit}} = \infty$).

Цей метод контролює f-значення найкращого альтернативного шляху, доступного з будь-якого предка поточного вузла. Якщо поточний вузол перевищує дану межу, то поточний етап рекурсії скасовується і рекурсія продовжується з альтернативного шляху. Після скасування даного етапу рекурсії відбувається заміна f-значення кожного вузла уздовж даного шляху найкращим f-значенням його дочірнього вузла. Завдяки цьому запам'ятовується f-значення найкращого листового вузла з забутого піддерева і тому в деякий наступний момент часу може бути прийняте рішення про те, чи варто знову розгортати це піддерево.

Метод RBFS є оптимальним, якщо евристична функція $h(n)$ є припустимою. Його просторова складність дорівнює $O(bd)$, але охарактеризувати часову складність досить важко: вона залежить і від точності евристичної функції, і від того, наскільки часто відбувалася зміна найкращого шляху в процесі розгортання вузлів. Метод RBFS страждає від недоліку, пов'язаного з занадто частим повторним формуванням вузлів.

Метод пошуку A^* – різновид пошуку за першим найкращим збігом. Метод припускає використання для кожного вузла n на графі простору станів *оцінної функції* виду $f(n) = g(n) + h(n)$, де $g(n)$ відповідає відстані на графі від вузла n до початкового стану, $h(n)$ – оцінка відстані від n до вузла, що подає кінцевий (цільовий) стан. Чим менше значення оцінної функції $f(n)$, тим «краще», тобто вузол n лежить на більш короткому шляху від вихідного стану до цільового.

Ідея методу полягає в тому, щоб за допомогою $f(n)$ відшукати найкоротший шлях на графі від вихідного стану до цільового. Звідси випливає, що якщо $h(n)$ – нижня оцінка дійсної відстані до цільового стану, тобто якщо $h(n)$ ніколи не дає завищеної оцінки відстані, то метод A^* завжди відшукає оптимальний шлях до цілі за допомогою оцінної функції $f(n)$.

Нехай s – вузол початкового стану; g – вузол цільового стану; Open – список, що містить, обрані, але необроблені вузли; Closed – список, що містить оброблені вузли. Тоді метод A^* буде полягати у виконанні таких кроків.

Крок 1. Установити: Open = {s}, Closed = ∅.

Крок 2. Якщо Open = ∅, то припинити виконання – шляху до цільового стану на графі не існує; у противному випадку – перейти до кроку 3.

Крок 3. Видалити зі списку Open вузол n, для якого $f(n) < f(m)$ для будь-якого вузла m, що вже знаходиться у списку Open, і перенести його в список Closed.

Крок 4. Сформувати список чергових вузлів, у які можливий перехід з вузла n, і видалити з нього усі вузли, що утворять петлі; з кожним із вузлів, що залишилися зв'язати покажчик на вузол n.

Крок 5. Якщо в сформованому списку чергових вузлів присутній вузол g, то завершити виконання і сформувати результат – шлях, породжений простежуванням покажчиків від вузла g до вузла s; у противному випадку для кожного чергового вузла n_i, включеного в список, виконати таку послідовність операцій, що задається кроками 5.1–5.3.

Крок 5.1 Обчислити $f(n_i)$.

Крок 5.2 Якщо n_i не є присутнім ані в списку Open, ані в списку Closed, то додати його в список Open, приєднати до нього оцінку $f(n_i)$ і установити зворотний покажчик на вузол n.

Крок 5.3 Якщо n_i вже є присутнім у списку Open або в списку Closed, то порівняти нове значення new = $f(n_i)$ з колишнім old=$f(n_i)$. Якщо old<new, то припинити обробку нового вузла. Якщо new<old, замінити новим вузлом колишній у списку, причому, якщо колишній вузол був у списку Closed, перенести його в список Open.

Крок 6 Перейти до кроку 2.

Всі методи A^* є припустимими. Пошук A^* є оптимальним, за умови, що $h(n)$ являє собою припустиму евристичну функцію, тобто за умови, що $h(n)$ ніколи не переоцінює вартість досягнення мети.

Метод A^* з ітеративним поглибленням (Iterative-Deepening A^* – IDA*) створений для скорочення потреб у пам'яті для пошуку A^* на основі ідеї ітеративного поглиблення в контексті евристичного пошуку. Основне розходження між IDA* і стандартним методом ітеративного поглиблення полягає в тому, що застосовуваною умовою зупинення розгортання слугує f-вартість (g+h), а не глибина; на кожній ітерації цим зупинним значенням є мінімальна f-вартість будь-якого вузла, що перевищує зупинне значення, досягнуте в попередній ітерації. Метод IDA* є практично застосовним для рішення багатьох задач з одиничними вартостями етапів і дозволяє уникнути істотних витрат, пов'язаних з підтримкою відсортованої черги вузлів.

Метод IDA* є небагато менш ефективним у порівнянні з методом. RBFS. Обидва ці методи піддаються потенційному експонентному збільшенню складності, пов'язаної з пошуком у графах, оскільки вони не дозволяють визначати наявність повторюваних станів, відмінних від тих, котрі знаходяться в поточному шляху. Тому дані методи здатні багато разів досліджувати один і той самий стан. Методи IDA* та RBFS страждають від того недоліку, що в них використовується занадто мало пам'яті. Між ітераціями IDA* зберігає тільки одне – поточну межу f-вартості. RBFS зберігає в пам'яті більше інформації, але кількість використовуваної в ньому пам'яті виміряється лише значенням $O(bd)$: навіть якби було доступно більше пам'яті, RBFS не здатний нею скористатися.

1.5.5 Пояснення процесу прийняття рішень

Подання інформації про поводження експертної системи в процесі формування ланцюжка логічного виведення при пошуку рішення є важливим з таких причин:

– користувачі, що працюють із системою, мають потребу в підтвердженні того, що в кожному конкретному випадку рішення, до якого прийшла програма, в основному є коректним;

– інженери зі знань, що мають справу з формуванням бази знань, повинні переконатися, що сформульовані ними знання застосовано правильно, у тому числі й у випадку, коли існує прототип;

– експерти у предметній області бажають простежити хід виведення рішень і спосіб використання тих відомостей, які з їхніх слів були введені в базу знань, що дозволить судити, наскільки коректно відомості застосовуються в даній ситуації;

– програмісти, що супроводжують, налагоджують і модернізують систему, мають потребу у інструменті, що дозволяє заглянути в «нутро» системи на рівні більш високому, ніж виклик окремих мовних процедур.

– менеджер системи, який використовує експертну технологію, що зрештою несе відповідальність за наслідки рішення, прийнятого програмою, також має потребу в підтвердженні, що ці рішення є досить обґрунтованими.

Відсутність достатньої прозорості поводження системи не дозволить експерту вплинути на її продуктивність або дати пораду, як можна її підвищити. Простежування й оцінка поводження системи – задача досить складна і для її вирішення необхідні спільні зусилля експерта і фахівця з інформатики.

Для відображення послідовності дій експертної системи і застосовуваних знань при побудові логічного виведення необхідні динамічні пояснення, тобто пояснення повинні генеруватися в процесі виведення рішення. Для підвищення інтелектуальності експертної системи варто передбачити різні типи питань, на які користувач може одержати відповідь з різною детальністю пояснень.

Види пояснень виділяють такі:

– *каузальні пояснення* – найбільш важливі і розкривають причинно-наслідкові зв'язки між розглянутими поняттями, а також дозволяють направляти процес вирішення задачі на розуміння сутностей. Наприклад, при побудові умовиводів на підставі відношень спільності і відношень перетворення користувачу необхідно пояснити, які логічні зв'язування використовувалися при побудові даного висновку і чому саме вони використовувалися системою;

– *пояснення через закон* – показують, на підставі якого закону, теорії або моделі побудовано висновок. При цьому пояснення будується на дедукції;

– *функціональні пояснення* (цільові, мотиваційні пояснення) будуються за принципом «*X* необхідно для проходження *Y*», тобто виконувані системою кроки розуміються стосовно до поставленої мети. Для цього необхідно закла-

сти в експертну систему знання експерта про предметну область і знання про те, які відношення і чому активізуються в тому або іншому випадку.

Методи формування пояснень – призначені для пояснення вихідних даних у світлі закладених в експертну систему знань і одержання відповідей на питання про те, як і чому система прийшла до деякого висновку, чому вона розмірковувала саме так; чому рекомендація системи є правильною і т. п.

Виділяють такі методи формування пояснень:

– *Метод фіксації ситуацій* – є найбільш простим і заснований на збереженні заздалегідь заготовлених текстів для фіксованого набору ситуацій. Ситуації, що вимагають пояснення, виділяються при проектуванні експертної системи. З кожною ситуацією зв'язується деякий текст, видаваний системою, якщо має місце відповідна ситуація. При використанні такого методу формування пояснення керування їхнім породженням найчастіше здійснюється експертною системою; при цьому підсистема пояснення не виділяється в окремий блок. Простота фіксації ситуацій є безсумнівною перевагою методу, а недоліки пов'язані з тим, що набір ситуацій, які пояснюються, звичайно жорстко фіксується при розробці системи і його розширення вимагає додаткового програмування. Текст, що пояснює, алгоритмічно не залежить від ситуації, яка призводить до семантичних неузгодженостей між ними при внесенні змін;

– *Метод трасування* – використовує при породженні пояснень *слід* (*трасу*) *судження*, проведеного експертною системою. Метод широко поширився при розробці блоків пояснення в системах, заснованих на поданні евристичного знання у формі продукцій. Слід цілеспрямованого судження являє собою ланцюжок правил, які застосовувалися. Можливості користувача зі вказівки того, що саме його цікавить, є обмеженими. Користувач може одержати пояснення того, навіщо потрібно досягнення деякої цілі (йому повідомляється правило, у якому ця ціль знаходиться в позиції умови), а також пояснення того, як досягнення даної мети відбивається на постановці інших підцілей (йому повідомляється правило, у якому дана ціль знаходиться в позиції наслідку).

1.6 Приклади та ілюстрації

Приклад 1. Надати опис, побудувати словник та сценарії для предметної області «Діагностика внутрішніх хвороб».

Назва предметної області: Діагностика внутрішніх хвороб.

Внутрішні хвороби є такими, що складно визначаються як для неспеціаліста, так і для професіонала. Їхня діагностика вимагає врахування великої кількості показників, що характеризують стан пацієнта. Кількість відомих хвороб внутрішніх органів та їх різновидів також є значною. Тому дуже актуальною є розробка експертної системи для діагностики внутрішніх хвороб.

Мета експертної системи – визначення діагнозу пацієнта за значеннями симптомів, що характеризують його стан.

Вхідними параметрами є симптоми пацієнта: температура тіла, місце болю, вид болю.

Вихідним параметром є діагноз пацієнта.

Словник предметної області наведено у табл. 1.1.

Узагальненнями для різних значень, що поступають на входи, є такі класи: апендицит, пієлонефрит.

Позитивний сценарій:

Якщо температура тіла – нормальна
та місце болю – у животі
та вид болю – слабкий,
то діагноз – хронічний апендицит.

Негативний сценарій:

Якщо температура тіла – нормальна
та місце болю – у животі
та вид болю – сильний,
то діагноз – невідомий.

Приклад 2. Нехай відомо, що у пацієнта: температура тіла є нормальною (із коефіцієнтом впевненості 1), місце болю – у животі (із коефіцієнтом впевненості 0,8), вид болю – слабкий (із коефіцієнтом впевненості 0,6) та відомо, що правило позитивного сценарію з попереднього прикладу, характеризується коефіцієнтом впевненості 0,9. Визначити коефіцієнт впевненості консеквента (наслідку) правила для пацієнта на основі стенфордської теорії коефіцієнта впевненості.

Коефіцієнти впевненості посилок антецедента з умови прикладу: $CF(X_1) = 1$, $CF(X_2) = 0{,}8$, $CF(X_3) = 0{,}6$.

Коефіцієнт упевненості першої та другої посилок антецедента визначаємо з відповідної формули: $CF(X_1, X_2) = 1$.

Коефіцієнт упевненості $CF(X_1, X_2)$ та третьої посилки визначаємо з відповідної формули: $CF(CF(X_1, X_2), X_3) = 1$.

Коефіцієнт впевненості правила з умови прикладу: $CF(R) = 0{,}9$.

Коефіцієнт упевненості консеквента $CF(C) = CF(X,Y)CF(R) = 0{,}9$.

Приклад 3. За допомогою правила Байєса оцінити імовірність діагнозу «гострий апендицит» за наявності симптому «підвищена температура», якщо відомо, що апріорна імовірність діагнозу «апендицит» – 0,001, апріорна імовірність наявності симптому «підвищена температура» – 0,1, а умовна імовірність наявності симптому «підвищена температура» за наявності діагнозу «апендицит» – 0,8.

Позначимо діагноз «апендицит» як подію A, а симптом «підвищена температура» як подію B. Тоді $P(A) = 0{,}001$, $P(B) = 0{,}1$, $P(B|A) = 0{,}8$. Необхідно знайти $P(A|B)$.

Використовуючи правило Байєса $P(A|B) = P(B|A)P(A) / P(B)$, отримаємо: $P(A|B) = 0{,}8 \cdot 0{,}001 / 0{,}1 = 0{,}008$.

Таблиця 1.1 – Словник предметної області

Назва об'єкта (події)	Визначення сутності	Назва атрибута (властивості) об'єкта	Допустимі значення атрибута
Пацієнт	людина, що проходить медичне обстеження	Симптоми	
		Діагноз	
Симптоми	показники, що відображують стан пацієнта	Температура тіла	Нормальна
			Підвищена
		Місце болю	У животі
			У попереку
		Вид болю	Сильний
			Слабкий
Температура тіла	характеристика температури тіла пацієнта у ^oC	Нормальна	
		Підвищена	
Нормальна	температура тіла, що менше 37^oC		
Підвищена	температура тіла, що більше або дорівнює 37^oC		
Місце болю	область тіла пацієнта, де відчувається біль	У животі	
		У попереку	
Вид болю	Умовний рівень відчуття болю	Сильний	
		Слабкий	
Діагноз	вид стану пацієнта	Гострий апендицит	
		Хронічний апендицит	
		Гострий пієлонефрит	
		Хронічний пієлонефрит	
Гострий апендицит	хвороба кишок, що проявляється у запаленні апендиксу		
Хронічний апендицит	застаріле запалення апендиксу		
Гострий пієлонефрит	хвороба нирок у стадії загострення		
Хронічний пієлонефрит	застаріла хвороба нирок		

? 1.7 Контрольні питання

1. Що таке штучний інтелект, яка в нього мета?
2. Які основні напрями досліджень в галузі штучного інтелекту ви знаєте?
3. Що таке інтелектуальна система, інтелектуальна задача?
4. Що таке властивість інтелектуальності?
5. Що таке інтелектуальні системи загального призначення?
6. Що таке спеціалізовані інтелектуальні системи?
7. Які ви знаєте проблемні області та їхні властивості?
8. Класифікація проблемних середовищ.
9. Класифікація задач, що вирішуються інтелектуальними системами.
10. Що таке система, заснована на знаннях?
11. Актуальність інтелектуальних системи, заснованих на знаннях.
12. Ієрархія рівнів систем, заснованих на знаннях.
13. Що таке експертна система?
14. Які властивості мають експертні системи?
15. Класифікація експертних систем.
16. Життєвий цикл та методологія розробки експертних систем.
17. Хто такий експерт, інженер зі знань, програміст, користувач?
18. Що таке засіб побудови експертної системи?
19. У чому полягає концепція «швидкого прототипу»?
20. У чому полягають структура та функціонування експертної системи?
21. Що таке база знань, факт, правило, робоча пам'ять, машина логічного виведення, оболонка експертної системи, інтерфейсна підсистема, підсистема набуття знань, редактор бази знань, підсистема пояснень?
22. Що таке динамічна експертна система?
23. Які ви знаєте режими роботи експертної системи?
24. Які ви знаєте переваги і недоліки експертних систем?
25. Що таке логічне виведення?
26. Які ви знаєте типи логічного виведення?
27. Що таке дедукція, індукція, абдукція?
28. У чому полягає звичайне (булеве) логічне виведення?
29. У чому полягає прямий метод виведення?
30. У чому полягає зворотний метод виведення?
31. У чому полягає метод Джона Стюарта Міля?
32. Що таке монотонна логіка, немонотонна логіка?
33. У чому полягають Стенфордська теорія коефіцієнта впевненості, теорія Демпстера–Шафера, метод виведення Байєса, метод виведення Нейлора?
34. Що таке пошук у просторі станів?
35. Що таке граф ТА/АБО?
36. Які ви знаєте стратегії пошуку в просторі станів?

37. У чому полягає метод пошуку з поверненнями?

38. Продуктивність та ефективність методів пошуку.

39. Які ви знаєте стратегії неінформованого (сліпого) пошуку?

40. У чому полягає метод породження і перевірки?

41. У чому полягають метод пошуку в ширину, метод пошуку за критерієм вартості?

42. У чому полягають метод пошуку в глибину, метод пошуку з поверненнями, метод пошуку з обмеженням глибини, метод пошуку в глибину з ітеративним поглибленням?

43. Які ви знаєте стратегії інформованого (евристичного) пошуку?

44. У чому полягає метод пошуку зі сходженням до вершини?

45. Що таке оцінна функція?

46. У чому полягають метод пошуку за першим найкращим збігом, метод рекурсивного пошуку за першим найкращим збігом, методи пошуку *A** та *A** з ітеративним поглибленням?

47. Що таке пояснення процесу прийняття рішень?

48. Які ви знаєте види пояснень?

49. Які ви знаєте методи формування пояснень?

50. Склад розроблювачів експертної системи, роль і задачі кожного з них.

51. Які властивості предметної області (об'єкта автоматизації) є передумовою для створення експертної системи.

52. Області людської діяльності в яких застосовуються експертні системи.

53. Чим експертні системи відрізняються від звичайних програмних додатків та типових програм штучного інтелекту?

54. Чи може програма, яка не використовує методи штучного інтелекту, мати такі ж властивості як експертна система?

55. У чому різниця між експертною системою та системою, заснованою на знаннях?

1.8 Практичні завдання

✍ *Завдання 1.* Обрати та проаналізувати предметну область, виділити основні об'єкти області та відношення між ними в контексті поставленої мети побудови бази знань.

Наприклад, можна обрати одну з таких тем.

1. Діагностика захворювань (органів дихання, кровообігу, травлення, зору, кістково-м'язової системи та ін.).

2. Діагностика несправностей пристроїв (телевізора, комп'ютера, автомобіля та ін.).

3. Розпізнання видів істот (комах, тварин, рослин та ін.) за ознаками.

4. Індивідуальний підбір (косметики, одягу, взуття, мобільного телефону, відеомагнітофону, музичного центру, відеокамери, телевізора,

комп'ютера, периферійного пристрою ЕОМ, автомобіля, житла, туристичної путівки, спеціальності для абітурієнтів, виду лікування, закладу освіти та ін.).

5. Розміщення об'єктів (станків, приміщень в споруді, що будується).

6. Планування (ремонтів, бізнес-процесів).

7. Комплектація комп'ютерних систем за критеріями: функції, швидкодія, ціна, ефективність.

Розробити короткий опис предметної області, що містить формулювання мети і призначення бази знань, актуальність створення експертної системи для предметної області. Виділити входи та виходи для експертної системи. Побудувати словник для предметної області, яка аналізується, що повинен містити такі стовпці: назва об'єкта (події), визначення сутності, назва атрибута (властивості) об'єкта, допустимі значення атрибута. Виконати узагальнення у предметній області, виділити (за можливістю) ієрархічні зв'язки.

Завдання 2. Написати реферат на одну з таких тем.

1. Історія та сучасні перспективи розвитку штучного інтелекту.
2. Застосування експертних систем.
3. Методи логічного виведення.
4. Методи пошуку у просторі станів та особливості їхнього застосування.
5. Байєсівські мережі виведення.
6. Методи пояснення рішень в експертних системах.
7. Принципи аналізу предметної області.

1.9 Література до розділу

Загальні питання штучного інтелекту висвітлено в [10, 13, 18, 22, 28, 36, 37, 43, 48, 58, 61, 77, 84, 88, 89, 92, 97, 99, 106, 112, 125, 126, 135–140, 142, 145, 147–149, 151, 184, 190, 191, 204], питання побудови та використання експертних систем розглянуто у [13, 17, 22, 30, 32, 46, 47, 48, 65–70, 79, 82, 90, 91, 104, 107, 111, 118, 133, 135–141, 142, 143, 145, 146, 148, 149, 156, 168, 183, 187, 193, 201, 202, 203], методи логічного виведення описано у [13, 17, 19, 22, 34, 38, 46, 47, 71, 72, 87, 94, 107, 118, 133, 141, 142, 148, 149, 165, 166, 168, 183, 187, 191, 200].

Навчально-методичні матеріали до розділу доступні на сайті автора в мережі Інтернет за адресою: `http://csit.narod.ru`.

2. МОДЕЛІ ПОДАННЯ ТА МЕТОДИ ОБРОБКИ ЧІТКИХ ЗНАНЬ

2.1 Знання та їхні властивості

2.1.1 Знання як спосіб подання інформації

Інформація (у широкому розумінні) – це будь-які відомості про певний об'єкт, процес або явище.

Факти – це інформація, що розглядається як надійна.

Ієрархію способів подання інформації (у порядку збільшення рівня ієрархії) виділяють таку.

1. *Шум* – відсутність видимих ознак інформації, складається з інформаційних елементів, що не становлять інтересу і можуть лише ускладнити сприйняття й подання інформації.

2. *Дані* – потенційне джерело інформації – елементи інформації, що у принципі можуть становити певний інтерес. Даними називають інформацію фактичного характеру, що є фіксованою певним способом та описує об'єкти, процеси і явища предметної області, а також їхні властивості. Дані являють собою ізольовані факти, відносини між якими з зовнішнім світом у них самих не зафіксовані.

3. *Інформація* (у вузькому розумінні) – потенційне джерело знань – оброблені дані, що явно становлять інтерес для користувачів.

4. *Знання* – це формалізована система суджень із принциповою і єдиною організацією, заснована на об'єктивній закономірності, що спостерігається у визначеній предметній області (принципи, зв'язки, закони), встановленій в результаті розумової діяльності людини, спрямованої на узагальнення досвіду, отриманого нею у результаті практичної діяльності, яка дозволяє ставити і вирішувати задачі в цій області. Знання визначають здатність використовувати інформацію і являють собою добре структуровані дані або *метадані* (дані про дані) – елементи інформації, зв'язані між собою і з зовнішнім світом.

5. *Метазнання* – спеціальним чином організовані знання про знання з метою реалізації процесу їхньої інтерпретації і планування виведення. Метазнання дозволяють інтелектуальній системі виправляти або доповнювати свої знання в міру навчання в процесі вирішення конкретних задач.

Онтологія в експертних системах являє собою метазнання, що описують усе, що відомо про розглянуту предметну область. В ідеальному випадку онтологія повинна бути подана у формальному виді задля того, щоб можна було легко виявляти несумісності і невідповідності.

Пояснення в експертних системах – вид метазнань, знання системи про саму себе, про спосіб використання своїх знань для одержання рішення. Крім того, за метазнаннями, отриманими пояснюючої підсистемою, можна згенерувати дерево

пояснень для формування підказки, повторного розширеного питання, навчального матеріалу або безпосереднього пояснення етапів рішення й у цілому результатів.

6. *Мудрість* – здатність використовувати знання щонайкраще – це метазнання, що дозволяють визначати найкращі цілі в житті і знаходити шляхи їхнього досягнення.

Питаннями збору, обробки, збереження даних та інформації займається *інформатика*. Мудрість є філософською категорією. Тому подальшим предметом розгляду є знання.

Епістемологія – наука про знання. У рамках цієї науки розглядаються характер, структура і походження знань.

2.1.2 Класифікація знань

Типи знань виділяють такі.

1. *Базові елементи знання* – інформація про властивості об'єктів реального світу. Пов'язані з безпосереднім сприйняттям, не вимагають обговорення і використовуються в тому виді, у якому отримані.

2. *Твердження і визначення* – засновані на базових елементах і заздалегідь розглядаються як достовірні.

3. *Концепції* – перегрупування або узагальнення базових елементів. Для побудови кожної концепції використовуються свої прийоми (приклади, контрприклади, окремі випадки, більш загальні випадки, аналогії).

4. *Відношення* – виражають як елементарні властивості базових елементів, так і відношення між концепціями. До властивостей відношень відносять їхні більші або менші правдоподібність і зв'язок з даною ситуацією.

5. *Теореми і правила перезапису* – окремий випадок продукційних правил (правил виду «якщо..., то..., інакше...») з цілком визначеними властивостями. Теореми не мають користі без експертних правил їхнього застосування.

6. *Алгоритми рішення* – необхідні для виконання визначених задач. В усіх випадках вони пов'язані зі знанням особливого типу, оскільки обумовлена ними послідовність дій виявляється оформленою в блок у строго визначеному порядку, на відміну від інших типів знань, де елементи знання можуть з'являтися і розташовуватися без зв'язку один з одним.

7. *Стратегії й евристика* – уроджені або придбані правила поведінки, що дозволяють у конкретній ситуації прийняти рішення про необхідні дії. Людина постійно користається цим типом знань при формуванні концепцій, вирішенні задач і формальних розсудах.

8. *Метазнання* – є присутнім на багатьох рівнях і подає знання того, що відомо, визначає значення коефіцієнта довіри до цього знання, важливість елементарної операції стосовно всієї множини знань. Сюди ж відносяться питання організації різного типу знань і вказівки, де, коли і як вони можуть бути використані.

Різновиди знань виділяють такі.

1. *Апріорні знання* (від лат. a priori – з попереднього) передують знанням, отриманим за допомогою органів почуттів, і не залежать від них. Апріорні знання розглядаються як універсально істинні, і ці знання неможливо спростувати, не впадаючи в протиріччя.

2. *Апостеріорні знання* (від лат. a posteriori – з наступного) – знання, отримані за допомогою органів почуттів – є протилежними стосовно апріорних знань. Істинність або хибність апостеріорних знань може бути перевірена на підставі чуттєвого досвіду. Але чуттєвий досвід не завжди може виявитися надійним, тому існує імовірність того, що апостеріорні знання будуть спростовані на основі нових знань.

Виділяють такі *види знань*.

1. *Процедурні знання* – знання, що задають послідовності дій, які мають бути виконані, і послідовності цілей, які мають бути досягнуті – відносяться до процедур обробки інформації і методів логічного виведення.

2. *Декларативні знання* – знання про те, чи є певне твердження істинним або помилковим – включають факти або аксіоми і правила, що відносяться до цих фактів. Термін *декларативний* застосовується до знань, виражених у формі декларативних тверджень. Для декларативних форм є особливістю організація бази знань, при якій у ній зберігаються тільки описи об'єктів і їхніх семантичних відношень і відсутня інформація про те, як можуть бути використані дані описи.

3. *Неявні знання* – підсвідомі знання, що не можуть бути виражені за допомогою мови. Якщо мова йде про комп'ютерні системи, то знання, подані в штучній нейронній системі, нагадують неявні знання, оскільки звичайно нейронна мережа нездатна безпосередньо пояснити суть знань, що містяться в ній, але могла б придбати таку здатність за наявності відповідної програми.

Логічною прозорістю системи називають здатність системи пояснювати методику прийняття рішення. Під цим розуміється, наскільки просто людині з'ясувати, що робить система і чому.

Експертні знання – спеціалізований різновид знань і навичок, яким володіють експерти. Експертні знання можуть відноситися до рівнів знань, метазнань і мудрості. Вони являють собою ті неявні знання і навички експерта, що повинні бути витягнуті і перетворені в явні задля того, щоб їх можна було подати в експертній системі. Причина, через яку знання є неявними, полягає в тому, що справжній експерт володіє цими знаннями настільки добре, що вони перетворилися в його другу натуру і не вимагають міркувань.

У залежності від часу існування виділяють:

– *статичні знання* – включають логічні правила рішення задач, правила реалізації процедур опитування експертів і користувачів, правила побудови функціональних та діагностичних моделей і аналізу результатів роботи, а також безпосередньо програмні модулі процедур опитування і математичних моделей, факти і дані про предметну область;

– *динамічні знання* – являють собою сукупність фактів і даних, одержуваних у ході рішення задачі, а також висновків (логічних висновків і строгих аналітичних рішень), вироблених у процесі рішення задачі.

У загальному вигляді знання подаються певною *семіотичною* (знаковою) *системою*. З поняттям «знак» безпосередньо зв'язані поняття денотат і концепт. *Денотат* – це об'єкт, що позначається даним знаком, а *концепт* – властивість денотата.

Інтенсіонал знака визначає зміст пов'язаного з ним поняття через його властивості, тобто значеннєвий зміст поняття. Інтенсіонал відокремлює знання від даних, що завжди задаються екстенсіонально.

Екстенсіонал знака визначає конкретний клас усіх його припустимих денотатів. Екстенсіонал поняття – набір конкретних фактів, що відповідають даному поняттю.

Інтенсіональні знання описують абстрактні об'єкти, події, відношення.

Екстенсіональні знання являють собою дані, що характеризують конкретні об'єкти, їхній стан, значення параметрів у визначені моменти часу.

Зазначені відмінності призвели до появи спеціальних формалізмів у вигляді *моделей подання знань*.

Аспекти семіотичної системи виділяють такі.

1. *Синтаксис* описує внутрішній пристрій знакової системи, тобто правила побудови і перетворення складних знакових виразів. Для природної мови, як відомо, синтаксис визначає правила побудови речень і зв'язаного тексту.

2. *Семантика* визначає відношення між знаками і їх концептами, тобто задає зміст або значення конкретних знаків.

3. *Прагматика* визначає знак з погляду конкретної сфери його застосування або суб'єкта, що використовує дану знакову систему.

Відповідно до перерахованих аспектів семіотичних систем можна виділити три *типи знань*:

– *синтаксичні знання* – характеризують синтаксичну структуру описуваного об'єкта або явища, що не залежить від смислу і змісту використовуваних при цьому понять;

– *семантичні знання* – містять інформацію, безпосередньо пов'язану зі значеннями і змістом описуваних явищ і об'єктів;

– *прагматичні знання* – описують об'єкти і явища з погляду розв'язуваної задачі, наприклад, з урахуванням діючих у даній задачі специфічних критеріїв.

Знання, якими володіє фахівець у якій-небудь області (дисципліні), можна розділити на формалізовані (точні) і неформалізовані (неточні).

Формалізовані знання формулюються в книгах і посібниках у виді загальних і строгих суджень (законів, формул, моделей, алгоритмів і т. п.), що відбивають універсальні знання.

Неформалізовані знання, як правило, не попадають до книг і посібників у зв'язку з їхньою конкретністю, суб'єктивністю і приблизністю. Знання цього роду є

результатом узагальнення багаторічного досвіду роботи й інтуїції фахівців. Вони звичайно являють собою різноманіття емпіричних (евристичних) прийомів і правил.

У залежності від того, які знання переважають у тій чи іншій області (дисципліні), її відносять до формалізованих (якщо переважають точні знання) або до неформалізованих (якщо переважають неточні знання) описових областей.

Задачі, розв'язувані на основі точних знань, називають *формалізованими*, а задачі, розв'язувані за допомогою неточних знань, — *неформалізованими*. Тут мова йде не про такі задачі, які не можна формалізувати, а про такі задачі, формалізація яких є невідомою.

До неформалізованих задач відносять ті, котрі мають одну чи декілька з таких особливостей: алгоритмічне рішення задачі є невідомим (хоча, можливо, й існує) або не може бути використане через обмеженість ресурсів ЕОМ (часу, пам'яті); задача не може бути визначена в числовій формі (потрібно символьне подання); цілі задачі не можуть бути виражені в термінах точно визначеної цільової функції. Як правило, неформалізовані задачі мають неповноту, помилковість, неоднозначність та (або) суперечливість знань (як даних, так і використовуваних правил перетворення).

Форми існування знань в інтелектуальних системах:

– вихідні знання (правила, виведені на основі практичного досвіду, математичні й емпіричні залежності, що відбивають взаємні зв'язки між фактами; закономірності і тенденції, що описують зміну фактів з часом; функції, діаграми, графи);

– опис вихідних знань засобами обраної моделі подання знань (множина логічних формул або продукційних правил, семантична мережа, ієрархії фреймів і т. п.);

– подання знань структурами даних, що призначені для збереження й обробки в ЕОМ;

– бази знань на машинних носіях інформації.

2.1.3 Особливості знань

1. *Внутрішня інтерпретованість* – властивість знань, що забезпечує можливість їхньої змістовної інтерпретації без використання відповідної програми, що відрізняє їх від даних, які у відриві від програми не несуть ніякої змістовної інформації і можуть змістовно інтерпретуватися лише відповідною програмою.

2. *Наявність класифікуючих відношень* (*структурованість*) – властивість знань, що визначає можливість довільного встановлення між окремими одиницями знань відношень типу «частина – ціле», «рід – вид», «елемент – клас», «клас – підклас», «тип – підтип», «ситуація – підситуація» для забезпечення рекурсивної вкладеності одних одиниць знань в інші. Кожна одиниця знань може бути включена до складу будь-якої іншої, і з кожної інформаційної одиниці можна виділити деякі складові її інформаційні одиниці. Це дозволяє записати і збе-

рігати окремо інформацію, однакову для всіх елементів множини. При необхідності цю інформацію можна автоматично передати опису будь-якого елемента множини. Такий процес передачі називається *спадкуванням інформації.*

3. *Наявність ситуативних зв'язків* – здатність знань відбивати закономірності щодо фактів, процесів, явищ і причинно-наслідкові відношення між ними. Ситуативні зв'язки допомагають будувати процедури аналізу знань на сумісність, суперечливість і інші, котрі важко реалізувати при збереженні традиційних масивів даних.

4. *Шкалування* використовується для фіксації співвідношень окремих інформаційних одиниць на основі різних шкал.

Метричні шкали дозволяють установлювати кількісні співвідношення і порядок тих чи інших сукупностей інформаційних одиниць, наприклад, шкали ваги, відстані і т. д. Метричні шкали характерні для опису даних.

Лінгвістичні шкали (розмиті або нечіткі шкали) замість кількісних одиниць використовують для опису знань лінгвістичні одиниці типу «багато», «мало», «рідко», «давно», «далеко», «близько» і т. д. Такі шкали будуються з використанням апарата теорії нечітких множин (див. розділ 3).

Семантична метрика. На множині інформаційних одиниць у деяких випадках корисно задавати відношення, що характеризують ситуаційну близькість інформаційних одиниць, тобто силу асоціативного зв'язку між інформаційними одиницями. Його можна назвати *відношенням релевантності* для інформаційних одиниць. Таке відношення дає можливість виділяти в інформаційній базі деякі типові ситуації. Відношення релевантності при роботі з інформаційними одиницями дозволяє знаходити знання, близькі до вже знайденого.

5. *Активність* – властивість знань впливати на інформаційні процеси і дії інтелектуальної системи, що відрізняє їх від даних, які є пасивними.

Історично склалося так, що в ЕОМ усі процеси, які протікають, ініціюються командами (активність команд), а дані пасивно зберігаються в пам'яті і використовуються командами лише в разі потреби.

Людині ж властива пізнавальна активність, тобто знання людини активні. Як і в людини, в інтелектуальній системі актуалізації тих чи інших дій сприяють знання, що є в системі. Таким чином, виконання програм в системі повинно ініціюватися поточним станом інформаційної бази. Джерелами активності знань є: поява фактів або описів подій, установлення зв'язків між ними, неповнота знань (виражається в необхідності їхнього поповнення), виявлення протиріч у знаннях (стає спонукальною причиною їхнього подолання і появи нових знань).

2.2 Інженерія знань

Інженерія знань – розділ теорії штучного інтелекту, що вивчає процеси і методи одержання, подання і формалізації знань для розробки систем, заснованих на знаннях, зокрема, експертних систем.

Набуття знань – це передача потенційного досвіду рішення проблеми від певного джерела знань і перетворення його у вигляд, що дозволяє використовувати ці знання.

Джерелами знань можуть бути книги, архівні документи, вміст інших баз знань і т. п., тобто деякі *об'єктивізовані знання*, переведені у форму, що робить їх доступними для споживача.

Іншим типом знань є *експертні знання*, що є у фахівців, але не зафіксовані у зовнішніх стосовно нього сховищах. Експертні знання є *суб'єктивними.*

Емпіричні знання, отримані шляхом спостереження за навколишнім середовищем, є ще одним видом суб'єктивних знань.

Введення в базу знань об'єктивізованих знань не є проблемою, виявлення і введення суб'єктивних експертних знань є досить важким. Для витягу і формалізації експертних знань розроблено багато стратегій інтерв'ювання експерта і багато моделей подання знань.

Види методів витягу знань виділяють такі.

1. *Комунікативні методи* – охоплюють методи і процедури контактів інженера зі знань з безпосереднім джерелом знань – експертом. Комунікативні методи поділяються на активні і пасивні.

1.1 *Пасивні методи* включають такі методи, де ведуча роль у процедурі витягу фактично передається експерту, а інженер зі знань тільки фіксує судження експерта під час роботи з прийняття рішень. До цієї групи відносяться: спостереження, аналіз протоколів «думок уголос» і лекції.

1.1.1 *Метод спостереження* полягає в тому, що інженер зі знань знаходиться безпосередньо поруч з експертом під час його професійної діяльності або імітації цієї діяльності. При підготовці до сеансу експерту необхідно пояснити мету спостережень і попросити його максимально коментувати свої дії; під час сеансу аналітик записує всі дії експерта і його пояснення. Рекомендується використовувати магнітофонний запис і відеозапис у реальному масштабі часу. Протоколи спостережень після сеансу ретельно розшифровуються, а потім обговорюються з експертом.

1.1.2 *Метод протоколювання «думок уголос»* полягає в тому, що експерта просять не тільки прокоментувати свої дії і рішення але й пояснити, як це рішення було знайдено. Іноді цей метод називають «вербальні звіти». Основними труднощами при протоколюванні «думок уголос» є принципова складність для людини пояснити, як вона думає, оскільки відомо, що люди не завжди здатні достовірно описувати розумові процеси.

1.1.3 *Метод витягу знань у формі лекцій* використовується при розробці бази знань як ефективний метод швидкого занурення інженера зі знань у предметну область. Курс лекцій звичайно дуже короткий і не перевищує 2–5 лекцій тривалістю до 1,5 годин кожна.

1.2 *Активні методи витягу знань* припускають, що ініціатива знаходиться цілком у руках інженера зі знань, який активно контактує з експертом різними способами. Активні методи підрозділяють на індивідуальні і групові.

1.2.1 *Індивідуальні активні методи* можна розділити на анкетування, інтерв'ю, вільний діалог та ігри з експертом.

1.2.1.1 *Анкетування* передбачає, що інженер зі знань заздалегідь складає запитальник або анкету, розмножує її і використовує для опитування декількох експертів. Експерт самостійно заповнює анкету після попереднього інструктування.

1.2.1.2. *Інтерв'ю* – специфічна форма спілкування інженера зі знань і експерта, у якій інженер зі знань задає експерту серію заздалегідь підготовлених питань з метою витягу знань про предметну область. На якість проведення інтерв'ю впливають три основних характеристики питання: мова питання (зрозумілість, лаконічність, термінологія), порядок питань (логічна послідовність і немонотонність), доречність питань (етика, увічливість).

1.2.1.3 *Вільний діалог* – метод витягу знань у формі бесіди інженера зі знань і експерта, у якій немає твердого регламентованого плану і запитальника.

1.2.2 *Групові методи витягу знань* забезпечують можливість одночасного використання знань декількох експертів, взаємодія яких забезпечує принципову новизну одержуваної інформації від накладення різних поглядів і позицій. До групових методів відносяться дискусії за «круглим столом», «мозкові штурми» і рольові ігри.

1.2.2.1 *Метод «круглого столу»* передбачає обговорення якої-небудь проблеми з обраної предметної області, у якому беруть участь з рівними правами кілька експертів. Задача дискусії – колективно, з різних точок зору, під різними кутами досліджувати спірні гіпотези предметної області. Спочатку учасники висловлюються у визначеному порядку, а потім переходять до вільного обговорення.

1.2.2.2 *«Мозковий штурм»* – один з найбільш розповсюджених методів розкріпачення й активізації творчого мислення. Основна ідея штурму – це відділення процедури генерування ідей у замкнутій групі фахівців від процесу аналізу й оцінювання висловлених ідей. Тривалість «штурму» до 40 хвилин. Учасникам (до 10 осіб) пропонується висловити будь-які ідеї на задану тему (критику заборонено). Регламент – до 2 хвилин на виступ. При аналізі відповідей лише 10–15 % ідей виявляються розумними, але серед них можуть бути дуже оригінальні. Оцінює результати група експертів, що не брала участь в генерації ідей.

1.2.2.3 *Гра* – такий вид людської діяльності, що відбиває (відтворює) інші її види. Експертні ігри включають *ділові ігри* (експеримент, де учасникам пропонується виробнича ситуація, а вони на основі знань і життєвого досвіду приймають рішення, що потім аналізуються), *діагностичні ігри* (вид ділових ігор, застосовуваний для діагностики методів прийняття рішень у медицині) і *комп'ютерні ігри* (ігри-дії, симулятори, стратегічні ігри, пригодницькі ігри, рольові ігри).

2. *Текстологічні методи* включають методи витягу знань з документів (методик, посібників, посібників і т. д.) і спеціальної літератури (статей, монографій, підручників). Серед методів витягу знань ця група є найменш розробленою.

3. *Витяг знань з даних* на основі використання методів інтелектуального аналізу даних (нейромережних і нечітких моделей, дерев рішень, асоціативних правил та ін.). Такий метод витягу знань може бути автоматизовано та включено до складу інтелектуальної системи, яка отримає здатність навчатися.

Набуття знань реалізується за допомогою двох функцій: одержання інформації ззовні та її систематизації. При цьому в залежності від здатності системи навчання до логічних висновків можливі різні форми набуття знань, а також різні форми одержуваної інформації. Форма подання знань для їхнього використання визначається усередині системи, тому форма інформації, яку вона може приймати, залежить від того, які здібності має система для формалізації інформації до рівня знань. Якщо система, що навчається, зовсім позбавлена такої здатності, то людина повинна заздалегідь підготувати все, аж до формалізації інформації, тобто чим більше здатності в машини до логічних висновків, тим менше навантаження на людину.

У залежності від здатності інтелектуальної системи до сприйняття знань у різних форматах, що якісно розрізняються між собою і здатністю до формалізації, можна виділити такі *методи набуття знань*.

– *Навчання без виведення* (механічне запам'ятовування) – це простий процес одержання інформації, при якому є необов'язковими функції виведення, а отримана інформація у вигляді програм або даних використовується для вирішення задач у незмінному виді. Іншими словами, це спосіб одержання інформації, характерний для існуючих комп'ютерів.

– *Навчання за прикладами* – це процес збору окремих фактів, їхнє узагальнення і систематизація.

– *Аналогія* – це метод виведення, при якому виявляється подоба між декількома заданими об'єктами; завдяки переносу фактів і знань, справедливих для одних об'єктів, на основі цієї подоби на зовсім інші об'єкти або визначається спосіб вирішення задач або передбачуються невідомі факти і знання.

– *Навчання на основі виведення за індукцією* – це навчання з використанням виведення високого рівня, як і при навчанні за аналогією. У процесі цього навчання шляхом узагальнення сукупності наявних даних виводяться загальні правила.

Індуктивне виведення – це виведення із заданих даних пояснюючого їх загального правила. Для точного визначення індуктивного виведення необхідно уточнити: множину правил-об'єктів виведення, метод подання правил, спосіб показу прикладів, метод виведення і критерій правильності виведення.

Методи витягу знань є підготовкою до структурування знань.

Узагальнений метод структурування знань має такий вигляд.

Крок 1. Визначення вхідних і вихідних даних. Цей крок визначає напрямок руху в полі знань – від вхідних даних до вихідних. Крім того структура вхідних і вихідних даних істотно впливає на форму і зміст поля знань.

Крок 2. Складання словника термінів і наборів ключових слів. На цьому кроці на основі протоколів сеансів витягу знань виписуються всі значимі слова, що позначають поняття, явища, процеси, предмети, дії, ознаки і т. п.

Крок 3. Виявлення об'єктів і понять. На основі обробки словника вибираються значимі для прийняття рішень поняття і їхні ознаки. Утворюється повний систематичний підбор термінів у даній предметній області.

Крок 4. Виявлення зв'язків між поняттями і визначення «сили зв'язності». Побудова мережі асоціацій.

Крок 5. Виявлення метапонять і деталізація понять. Аналіз зв'язків дозволяє структурувати поняття, виявляти метапоняття або деталізувати поняття.

Крок 6. Побудова піраміди знань, тобто ієрархічної градації понять, підйом по якій означає поглиблення понять і підвищення рівня абстракції (узагальненості) понять.

Крок 7. Визначення відношення як усередині кожного з рівнів піраміди, так і між рівнями. На цьому кроці даються імена тим зв'язкам, що визначені на 4 та 5 кроках. Позначаються причинно-наслідкові й інші види відношень.

Крок 8. Визначення стратегій прийняття рішень, тобто виявлення ланцюжків суджень, що пов'язують усі сформовані поняття і відношення в динамічну систему поля знань. Стратегії додають активність знанням і забезпечують перехід від вхідних даних до рішення.

2.3 Подання знань

Теорія подання знань — це окрема область досліджень, тісно пов'язана з філософією формалізму і когнітивною психологією. Предмет дослідження в цій області – методи асоціативного збереження інформації, подібні тим, що існують у мозку людини. При цьому основна увага, природно, приділяється логічній, а не біологічній стороні процесу, опускаючи подробиці фізичних перетворень.

Подання (representation) – множина синтаксичних і семантичних угод, що уможливлює опис у певній проблемній області (об'єкти в цій області, їхні властивості, відношення, що існують між об'єктами).

Опис (description) дозволяє використовувати угоди з подання для опису визначених предметів.

Схема подання знань повинна адекватно виражати всю необхідну інформацію, підтримувати ефективне виконання кінцевого коду, забезпечувати природний спосіб вираження необхідних знань. Задача будь-якої схеми подання полягає в тому, щоб зафіксувати специфіку області визначення задачі і зробити цю інформацію доступною для механізму вирішення проблеми.

Мова подання (representation language) – комп'ютерна мова, орієнтована на організацію описів об'єктів і ідей, у противагу статичним послідовностям інструкцій або збереженню простих елементів даних. Основними критеріями доступу до подання знань є логічна адекватність, евристична потужність і природність, органічність нотації.

Мова подання знань є засобом, що дозволяє вирішувати задачі. Власне кажучи, спосіб подання знання повинний забезпечити природну структуру вираження знання, що дозволяє вирішити проблему. Спосіб подання повинний зробити це знання доступним комп'ютеру і допомогти програмісту описати його структуру.

Мова подання знань повинна дозволяти обробляти знання, виражені в якісній формі, одержувати нові знання з набору фактів і правил, відображати загальні принципи і конкретні ситуації, передавати складні семантичні значення, забезпечувати міркування на метарівні.

Абстракція (abstraction) – подання тільки тієї інформації, що необхідна для досягнення заданої мети, є необхідним засобом керування складними процесами. Крім того, кінцеві програми повинні бути раціональними в обчислювальному відношенні.

Виразність і *ефективність* є взаємозалежними характеристиками оцінки мов подання знань. Часто досить виразні засоби подання в одних задачах є зовсім неефективними в інших класах задач. Іноді виразністю можна пожертвувати на користь ефективності. У той же час не можна обмежувати можливості такого відображення, що дозволяє фіксувати істотні знання, що приводить до ефективного вирішення конкретної задачі. Розумний компроміс між ефективністю і виразністю – нетривіальна задача для розроблювачів інтелектуальних систем.

Синтаксис подання специфікує набір правил, що регламентують об'єднання символів для формування виразів мовою подання. Можна говорити про те, що вираз є гарно або погано сформованим, тобто про те, наскільки він відповідає цим правилам. Зміст повинний мати тільки добре сформовані вирази.

Семантика подання специфікує, як повинний інтерпретуватися вираз, побудований відповідно до синтаксичних правил, тобто як з його форми можна витягти якийсь зміст. Специфікація звичайно виконується присвоєнням змісту окремим символам, а потім індукуванням присвоєння в більш складних виразах.

Для того щоб наділити інтелектуальні системи знаннями, їх необхідно подати у визначеній формі. Виділяють такі *способи наділення знаннями програмних систем.*

Перший – помістити знання в програму, написану звичайною мовою програмування. Така система буде являти собою єдиний програмний код, у якому знання не винесені в окрему категорію. Незважаючи на те, що основна задача буде вирішена, у цьому випадку важко оцінити роль знань і зрозуміти, яким чином вони використовуються в процесі вирішення задач. Нелегкою справою є модифікація і супровід подібних програм, а проблема поповнення знань може стати нерозв'язною.

Другий спосіб базується на концепції баз даних і полягає у винесенні знань в окрему категорію, тобто знання подаються у визначеному форматі і розміщуються у базі знань. База знань легко поповнюється й модифікується. Вона є автономною частиною інтелектуальної системи, хоча механізм логічного виведення, реалізований у логічному блоці, а також засіб ведення діалогу накладають визначені обмеження на структуру бази знань і операції з нею. У сучасних системах прийнято саме цей спосіб.

Варто помітити, що для того, щоб помістити знання в комп'ютер, їх необхідно подати визначеними структурами даних, що відповідають обраному середовищу розробки інтелектуальної системи. Отже, при розробці системи спочатку здійснюються накопичення і подання знань, причому на цьому етапі обов'язкова участь людини, а потім знання подаються визначеними структурами даних, зручними для збереження й обробки в ЕОМ.

2.3.1 Моделі подання знань

Однією з основних проблем, характерних для систем, заснованих на знаннях, є *проблема подання знань*. Це обумовлюється тим, що форма подання знань впливає на характеристики і властивості системи.

Для можливості оперування знаннями з реального світу за допомогою комп'ютера, необхідно здійснити їхнє моделювання. При цьому необхідно відрізняти знання, призначені для обробки комп'ютером від знань, використовуваних людиною.

При проектуванні моделі подання знань варто враховувати такі фактори, як *однорідність подання*, що приводить до спрощення механізму керування логічним виведенням і керуванням знаннями, і *простота розуміння*, що припускає доступність розуміння подання знань експертами і користувачами системи. Однак виконати ці вимоги в однаковій мірі, як для простих, так і складних задач досить важко.

2.3.2 Класифікація моделей подання знань

Моделі подання знань класифікують *у залежності від ступеня декларативності (процедурності) мовних засобів*, використовуваних для опису знань.

Декларативна модель подання знань ґрунтується на припущенні, що проблема подання деякої предметної області зважується незалежно від того, як ці знання потім будуть використовуватися. Тому модель складається зі статичних описових структур знань і механізму виведення, що оперує цими структурами і практично не залежить від їхнього змістовного наповнення.

Декларативні знання надходять у систему від експертів предметної області і включають факти або аксіоми і правила, що відносяться до цих фактів.

Для декларативних форм особливою є організація бази знань, при якій у ній зберігаються тільки описи об'єктів і їхніх семантичних відношень і відсутня інформація про те, як можуть бути використані дані описи. При цьому в якійсь мірі виявляються роздільними синтаксичні і семантичні аспекти знання, що є перевагою через можливість досягнення визначеної універсальності. Така універсальність забезпечує гнучкість і економічність декларативного подання, тому що дозволяє по-різному використовувати ті самі факти.

У декларативних моделях не містяться в явному виді описи виконуваних процедур. Ці моделі являють собою звичайно множини незалежних фактів або тверджень, що дозволяє здійснювати модифікацію знань і навчання простим додаванням або усуненням тверджень.

Предметна область подається у вигляді синтаксичного опису її стану (за можливістю повного). Виведення рішень ґрунтується, як правило, на процедурах пошуку в просторі станів.

У *процедуральній моделі* знання містяться в процедурах – невеликих програмах, що задають послідовності дій, які повинні бути виконані, і послідовності цілей, що повинні бути досягнуті. При цьому можна не описувати всі можливі стани об'єкта для реалізації виведення. Досить зберігати деякі початкові стани і процедури, що генерують необхідні описи ситуацій і дій.

При процедуральному поданні знань семантика безпосередньо закладена в опис елементів бази знань, за рахунок чого підвищується ефективність пошуку рішень. У порівнянні з процедуральною частиною статична база знань у них є малою. Вона містить не «незмінні аксіоми», а лише так звані «твердження», які є прийнятними в даний момент, але можуть бути змінені чи вилучені в будь-який час. Загальні правила виведення подані у вигляді спеціальних цілеспрямованих процедур, що активізуються в міру потреби.

Засобом підвищення ефективності генерації виведення в процедуральних моделях є додавання в систему знань про застосування, тобто знань про те, яким чином використовувати накопичені знання для вирішення конкретної задачі. Ці знання, як правило, теж подаються в процедуральній формі.

Головна перевага процедуральних моделей полягає в більшій ефективності механізмів виведення за рахунок уведення додаткових знань про застосування, що, однак, знижує їхню спільність.

Інша важлива перевага укладена у виразній силі. Процедуральні системи здатні змоделювати практично будь-яку модель подання знань. Крім того, у них реалізується розширена система виведення.

Більшість розширених форм виведень можуть бути охарактеризовані поняттям «припущення про відсутність» і зводяться до схеми: «Якщо *A* (попередня умова) є істинною і немає доказів проти *B*, то запропонувати *B*».

Подібні правила виведення виявляються корисними у випадках, якщо:

– у системі подання знань окремі факти не подані або невиведені (*неповнота знань*) – правила виведення дозволяють гіпотетично визнавати їх вірними за умови, що в системі немає чи в ній невиведені докази противного;

– через обмеженість ресурсів процеси виведення не можуть завершитися, а повинні бути залишені для одержання результатів *(виведення в умовах обмеженості ресурсів*) – у цьому випадку правила визначають подальші дії системи.

Системи подання, що містять подібні правила, виявляються *немонотонними*, тобто додавання нових тверджень може заборонити генерацію виведення, що спочатку могло бути отримане. Додавання нових фактів може привести до виникнення протиріч. У деяких системах крім самих тверджень містяться також записи причин, з яких були прийняті ці твердження. При додаванні нових фактів здійснюється перевірка того, чи зберігаються справедливість тверджень і відповідність причинам.

Узагальнюючи вищевикладене, розходження між декларативним і процедурним поданням можна виразити як розходження між «знати що» і «знати як». Кожне подання має свої переваги і недоліки. Прагнення найбільш повно використовувати переваги обох видів подань привело до розробки *змішаних подань*, тобто декларативних подань із приєднаними процедурами (наприклад, фреймові моделі і моделі, що використовують розширені семантичні мережі).

Моделі подання знань поділяють на логічні й евристичні.

Логічні моделі подання знань засновані на понятті формальної системи. Їхніми прикладами можуть бути числення предикатів і будь-яка конкретна система продукцій. Ці моделі є *дедуктивними* системами; у них використовується модель виведення з заданої системи посилок за допомогою фіксованої системи правил. Подальший розвиток предикатних систем пов'язаний з переходом до *індуктивних* систем, у яких правила виведення породжуються на основі обробки кінцевого числа навчаючих прикладів. Перспективний напрямок – розвиток псевдофізичних логік, що, як і індуктивні моделі, поки не одержали широкого практичного застосування.

Евристичні моделі подання знань на відміну від логічних мають різноманітний набір засобів, що передають специфічні особливості моделі, і в цьому змісті перевершують логічні як за можливістю адекватно подати предметну область, так і за ефективністю використовуваних правил виведення. До евристичних моделей варто віднести мережні, фреймові і продукційні моделі. Відзначимо, що продукційні моделі, використовувані для подання знань, відрізняються від формальних продукційних систем використанням більш багатих правил і змістом евристичної інформації про специфіку предметної області, що виражається у вигляді семантичних структур.

Основні *види моделей подання знань* виділяють такі: *мережні моделі* (включають семантичні мережі та фрейми) та *логічні моделі* (включають моделі на основі числення висловлювань та продукційні моделі).

Мережна модель подання знань – це граф, як правило, орієнтований, вершини якого відповідають певним поняттям, сутностям, а дуги – відношенням та зв'язкам між ними.

2.3.3 Порівняння моделей подання знань

Знання можливо переводити з однієї форми подання в іншу. Однак можливість опису деякого подання, отриманого одним зі способів, іншим способом означає, що ці подання є рівнозначними. Але ця попарна рівнозначність відношень зовсім не означає, що всі способи подання дозволяють одержати рівнозначні системи подань. Це пояснюється тим, що найчастіше подання однієї формули не можна виразити іншою формулою, а також тим, що різними є самі ідеї і теоретичні системи, що знаходяться в основі цих формул, і це має свій вплив на методи виведень.

Для порівняння моделей подання знань використовують такі *критерії оцінювання моделей і методів подання знань*.

– *Формалізованість*. Найнижчий рівень формалізованості мають семантичні мережі, більш високий рівень у продукційних моделей, ще більший рівень формалізованості є у фреймових моделей, найвищий рівень формалізованості має логіка предикатів 1-го порядку.

– *Можливість поповнення.* Найнижчий рівень має логіка предикатів, більш високі рівні займають, відповідно, семантичні мережі, фрейми, продукційні моделі.

– *Нотаційна адекватність* в основному виражається синтаксичною будовою моделей подання знань та має особливе значення для систем, що використовують експертні знання, проектування та підтримка котрих багато у чому залежить від людей, що не є спеціалістами у формалізації знань. Найнижчий рівень нотаційної адекватності є у логіки предикатів, інші моделі мають більш високий відносно логіки предикатів, але приблизно однаковий у порівнянні між собою.

– *Відображення семантики предметної області*. Найнижчий рівень відображення у логіки предикатів, продукційні моделі та семантичні мережі мають більш високий рівень ніж у логіки предикатів, але порівняний між собою, що свідчить про їхню приблизно однакову ефективність (продукціями легко відображуються каузальні залежності, а семантичними мережами – структурні відношення), найвищий рівень відображення мають фрейми.

– *Інференційні можливості*. У порядку зростання рівня цього критерію слідують: семантичні мережі, фрейми, продукції та логіка предикатів.

Жодна з моделей подання знань не є лідером за всіма порівнюваними характеристиками.

Більш прийнятні за виразною потужністю і нотаційною адекватністю мережні моделі. Причина, мабуть, у тому, що вони є менш формальними, більш гнучкими й адекватними конкретним застосуванням. Семантична мережа підхо-

дить для опису об'єктів динамічної природи, оскільки в ній може бути забезпечене включення нових понять і зв'язків, вона може навчатися і зростати.

Предикатні моделі дозволяють відображати твердження більш загального характеру і містять мало індивідних констант. Для знань предметної області, навпаки, характерно відносно велике число предметних констант і тверджень про ці константи. Використання предикатних моделей для опису об'єктів динамічної природи ускладнюється через їхню відносну закритість.

Речення логіки предикатів мають багато спільного з продукційними правилами. Власне кажучи, інтерпретуючи правило продукції як речення логіки предикатів, можна без яких-небудь утруднень замінити його на предикати. У загальному випадку кожну дугу семантичної мережі можна задати предикатами як відношення між сутностями, що описують вузли на кінцях дуги.

З загальних розумінь випливає, що теоретичний фундамент у продукційних правил і семантичних мереж є досить слабким у порівнянні з логікою предикатів. І від того, що він є слабким, немає необхідності завзято відстоювати прийняті в них формалізми і можна їх видозмінювати. Одним із прикладів цього є повноправне включення в опис тих понять, що визначають кількісну оцінку цього опису типу фактора впевненості, а у логіці предикатів зробити це не так просто і повинні бути створені відповідні теоретичні основи.

Проте описова потужність логіки предикатів як єдиної системи формалізмів подання знань вище, ніж в інших систем. Це пояснюється тим, що в логіці предикатів можна у формули вільно включати змінні, а в інших методах це зовсім неможливо або можливо у вкрай обмеженому ступені.

2.3.4 Семантичні мережі

При індуктивному виведенні на основі фактів і відношень між ними, що існують у навколишньому світі, люди використовують асоціативні зв'язки. Для подання таких знань доцільно використовувати мережну модель.

Семантика визначає смисл знаків та відношень між символами і об'єктами, які вони визначають.

Семантична мережа – графічна система позначень для подання знань в шаблонах пов'язаних вузлів і дуг. Більш формально: семантична мережа – це орієнтований граф, *вершини* якого – поняття, а *дуги* – відношення між ними. Мережні моделі формально можна задати у вигляді $H = <I, C_1, C_2, ..., C_n, G>$, де I – множина інформаційних одиниць; $C_1, C_2, ..., C_n$ – множина типів зв'язків між інформаційними одиницями; G – відображення, що задає між інформаційними одиницями, що входять у I, зв'язки з заданого набору типів зв'язків.

Для всіх семантичних мереж спільним є декларативне графічне подання, що може використовуватися для подання знань або створення автоматизованих систем прийняття рішень на основі знань.

Компоненти семантичних мереж

Сама семантична мережа є моделлю пам'яті людини і не розкриває повністю яким чином виконується подання знань. Тому в контексті подання знань семантична мережа повинна розглядатись як метод моделювання, в якому передбачені можливості структурування, механізм виведення та процедури використання знань.

Вершини можуть являти собою: поняття, події, властивості.

Мітки вершин мають посилальний характер і являють собою деякі імена. У ролі імен можуть виступати, наприклад, слова природної мови.

Мітки дуг позначають елементи множини відношень.

При використанні семантичної мережі для подання знань важлива класифікація типів об'єктів і виділення деяких фундаментальних видів зв'язків між об'єктами. Незалежно від особливостей середовища, що моделюється, можна припускати, що будь-яка більш-менш складна його модель відображає які-небудь узагальнені, конкретні й агрегатні об'єкти.

Узагальнений об'єкт – це деяке поняття, що є відомим і широко використовуваним у предметному середовищі, яке моделюється. Узагальнений об'єкт фактично подає певним чином клас об'єктів проблемного середовища.

Конкретний (*індивідний*) *об'єкт* – це певним чином виділена одинична (індивідна) сутність.

Агрегатний об'єкт – об'єкт проблемного середовища, що складений певним чином з інших об'єктів, які є його частинами. Агрегатним може бути як узагальнений, так і конкретний об'єкт.

У термінах описаної типізації об'єктів проблемного середовища визначаються і *зв'язки між об'єктами*.

Родовидовий зв'язок може існувати між двома узагальненими об'єктами (наприклад, між об'єктами «транспорт» – рід і «літак» – вид). Усі властивості родового поняття, як правило, властиві й видовому (*спадкування властивостей*). Використання спадкування забезпечує ефективний спосіб спрощення подання знань і скорочення обсягу інформації, що потрібно запам'ятовувати для кожного конкретного вузла. Це дає можливість значною мірою прискорити процес обробки знань, а також витягати інформацію за допомогою запитів загального характеру.

Зв'язок «є представником» може існувати між узагальненим і конкретним об'єктами. Він має місце в тому випадку, коли конкретний об'єкт належить класу, відображуваному відповідним узагальненим об'єктом (наприклад, конкретний об'єкт «Іванов» є представником узагальненого об'єкта «студент»). Властивості, що властиві узагальненому об'єкту, характеризують і будь-який конкретний об'єкт-представник. Таким чином, множина властивостей конкретного об'єкта містить у собі підмножину властивостей, якою

він наділяється як представник тих чи інших узагальнених об'єктів (або збігається з цією множиною).

Зв'язок «є частиною» може існувати між агрегатним об'єктом і якимнебудь іншим об'єктом проблемного середовища. При цьому, частиною конкретного агрегатного об'єкта не може бути узагальнений об'єкт.

У семантичних мережах використовують три основні типи об'єктів: поняття, події і властивості.

Поняття – постійні елементи предметної області (концепти) – визначають абстрактні або фізичні (конкретні) об'єкти. В природній мові – це частіше іменники.

Властивості (атрибути) описують характеристики понять і подій. Відносно понять описують їх особливості (колір, розмір, якість та ін.), а відносно подій – час, місце, тривалість.

Події відповідають діям, які відбуваються в предметній області і визначають тип дії, роль, яку грають об'єкти в цих діях. Подаються дієсловом. При поданні подій заздалегідь виділяються прості відношення, які характеризують основні компоненти події. Насамперед з події виділяється дія, яка звичайно описується дієсловом. Далі визначаються: об'єкти, які діють та об'єкти, над якими ці дії виконуються.

Дуги відображають різноманітні *семантичні відношення*, які умовно можливо розділити на чотири класи:

– лінгвістичні (час, стан, вид, вага, колір, агент, джерело, приймач та ін.);

– логічні (заперечення, кон'юнкція, диз'юнкція, імплікація);

– теоретико-множинні (частина – ціле, елемент – множина, клас – підклас, підмножина, близькість та ін.);

– квантифікаційні (квантори загальності, існування, нечіткі квантори: багато, кілька, часто та ін).

Лінгвістичні відношення відображають смисловий зв'язок між подіями, поняттями або властивостями.

Лінгвістичні відношення бувають: *дієслівні* (час, вигляд, рід, стан, спосіб), *атрибутивні* (мати властивість, мати значення, колір, розмір, форма) та *відмінкові* (всі зв'язки понять, подій і властивостей з дією – див. табл. 2.1).

Таблиця 2.1 – Основні відмінки

Відмінок	*Лінгвістичне (відмінкове) відношення, зв'язок дій з*
Агент	предметом, що є ініціатором дії
Об'єкт	предметом, що зазнає дії
Джерело	розміщенням предмета перед дією
Приймач	розміщенням предмета після дії
Час	моментом виконання дії
Місце	місцем проведення дії
Мета	дією іншої події

Серед відмінкових відношень окремо можна виділити *просторові зв'язки* (далеко, близько, від, за, під, над) та *часові зв'язки* (раніше, пізніше, протягом, учора).

Логічні відношення – це операції, що використовуються в численні висловлювань (диз'юнкція, кон'юнкція, інверсія, імплікація).

Теоретико-множинні відношення – це відношення частини і цілого, відношення підмножина, відношення елемент множини (приклади: зв'язки типу «це»: «AKO – A kind of», «is», IS-A; зв'язки «має частиною»: «has part», PART-OF).

Квантифікаційні відношення – це логічні квантори спільності та існування. Вони використовуються для подання таких знань як «будь-який двугун необхідно діагностувати», «існує технік *A*, що діагностує двугун *B*».

Інтенсіоналом відношення R_j називають набір пар виду: $\mathrm{INT}(R_j)=\{...,[A_i, \mathrm{DOM}(A_i)],...\}$, де R_j – ім'я відношення; $\mathrm{DOM}(A_i)$ – *домен* A_i, тобто множина значень атрибута A_i відповідного відношення R_j, множина атрибутів $A = \{A_i, i=1, 2, ..., n\}$ і множина відношень $R=\{R_j, j=1, 2, ..., m\}$ – кінцеві.

Базовою множиною або модельною множиною об'єктів, на яких задаються відношення R_j, називають об'єднання всіх доменів.

Екстенсіоналом відношення R_j називають множину $\mathrm{EXT}(R_j)=\{F_1, ..., F_p\}$, де F_k ($k = 1, 2, ..., p$) – факт відношення R_j, що задається сукупністю пар виду «атрибут – значення», називаних *атрибутивними парами*. Під *фактом* розуміють конкретизацію визначеного відношення між зазначеними об'єктами.

Компонентами семантичної мережі є не тільки поняття (об'єкти) і відношення предметної області, але і складені з них різні ситуації – *фрейми*, що відображаються на семантичній мережі фрагментами мережі.

Класифікація семантичних мереж.

За *кількістю типів відношень* можна виділити семантичні мережі:

– *однорідні* з єдиним типом відношень;

– *неоднорідні* з різними типами відношень.

За *типами відношень* можна виділити семантичні мережі:

– бінарні – в яких відношення зв'язують два об'єкти;

– *N*-арні – в яких є спеціальні відношення, зв'язуючі більш двох понять.

За *типом структури вершин* виділяють семантичні мережі:

– *простого типу*, у яких вершини не мають власної структури;

– *ієрархічні*, у яких вершини мають власну структуру у вигляді мережі.

Одна з основних відмінностей ієрархічних семантичних мереж від простих семантичних мереж полягає в можливості розділити мережу на підмережі (*простори*) і встановлювати стосунки не тільки між вершинами, але і між просторами.

Усі вершини і дуги є елементами, принаймні, одного простору. Поняття простору аналогічно поняттю дужок у математичній нотації. Різні простори, що існують у мережі, можуть бути упорядковані у виді *дерева просторів*, вершинам якого відповідають простори, а дугам – відношення «видимості».

Відношення «видимості» дозволяє згрупувати простори в упорядковані множини – *«перспективи»*. Перспектива звичайно використовується для обмеження мережних сутностей, «видимих» деякою процедурою, що працює з мережею. Звичайно в перспективу включають не будь-які, а ієрархічно згруповані простори. Властивість «невидимості» дозволяє підвищити ефективність операції пошуку у мережі.

За *рівнем подання знань* виділяють:

– *Інтенсіональні семантичні мережі* – містять інтенсіональні знання про об'єкти, що моделюються, й описують предметні області, що моделюються, на узагальненому, концептуальному рівні.

– *Екстенсіональні семантичні мережі* – описують екстенсіональні знання про об'єкти, що моделюються, будучи ніби «фотографією» їхнього поточного стану, тобто в екстенсіональній мережі здійснюється конкретизація та наповнення фактичними даними.

У залежності від *типів та характеру зв'язків* у моделі виділяють:

– *Класифікаційні мережі* – використовують відношення структуризації. Такі мережі дозволяють у базах знань уводити різні ієрархічні відношення між інформаційними одиницями.

– *Функціональні мережі* – характеризуються наявністю функціональних відношень. Їх часто називають *обчислювальними моделями*, тому що вони дозволяють описувати процедури обчислення одних інформаційних одиниць через інші. У таких мережах дуги відбивають той факт, що вершина, з якої йде дуга в деяку іншу вершину, грає стосовно цієї вершини роль аргументу. Опис, що відповідає деякій вершині, задає ту процедуру знаходження результату, що відповідає даній вершині-функції.

– *Сценарії* – використовують каузальні відношення, а також відношення «засіб – результат», «знаряддя – дія» і т. п.

Виділяють такі найбільш загальні *види семантичних мереж*: означальні, доказові, імплікаційні, виконувані, такі, що навчаються, і гібридні.

1. *Означальні мережі* (definitional networks) підкреслюють підтип (subtype) або відношення «є» (is-a) між типом концепту й підтипом, що визначається.

Більш загальний концепт називається *супертипом* (supertype або hypernym), а більш спеціалізований концепт називається *підтипом* (subtype або hyponym).

Результуюча мережа, також називана узагальненням або *категоризаційною ієрархією*, підтримує *правило спадкування* (inheritance rule) для копіювання властивостей, визначених для супертипу, усім його підтипам. Оскільки визначення є істинними за визначенням, часто передбачається, що інформація в цих мережах обов'язково повинна бути істинною.

2. *Доказові мережі* (assertional networks) призначені для доказу суджень. На відміну від означальних мереж, інформація в доказовій мережі вважається умовно істинною, якщо це явно не відзначено модальним опера-

тором. Деякі доказові мережі були використані як моделі концептуальних структур, що лежать в основі семантики природної мови.

Логічна семантична мережа (propositional semantic network) – доказова мережа з вузлами, що подають судження. Вона може також містити підмережі або вкладені мережі, що виражають подальшу інформацію щодо цих суджень.

Begriffsschrift або написання концептів – логічна семантична мережа, запропонована Г. Фреге (G. Frege) у 1879 р., яка використовувала подання у вигляді дерева для першої нотації, здатної виражати всі можливі судження логіки першого порядку.

Ч. Пеірс (C. Peirce) у 1880–1897 р. незалежно розвивав алгебраїчну нотацію, що зі зміною символів Пеано стала сучасною нотацією для числення предикатів. Пеірс запропонував овал, що міг би додавати і заперечувати довільно великий граф або підграф. Тоді комбінації овалів із з'єднанням і квантором існування могли б виражати всі логічні оператори, використовувані в алгебраїчній нотації.

Екзистенціальний граф (existential graph) – логічна семантична мережа, що комбінує реляційні графи з овальними вузлами, які можуть містити інший екзистенціальний або реляційний граф. Коли овали використовуються, щоб заперечувати вкладені графи, екзистенціальний граф може подавати будь-яке судження в логіці першого порядку, але овали можуть також використовуватися, щоб подати модальну і метарівневу інформацію щодо вкладених графів.

Структура подання бесіди (discourse representation structure) – логічна семантична мережа, винайдена Г. Кампом (H. Kamp) у 1981–1993 р. для подання семантики природної мови, у якій судження подаються прямокутниками, що можуть містити деяку інформацію, виражену в лінійній нотації, і іншу інформацію, виражену вкладеними прямокутниками. Вкладення суджень у структурі подання бесіди є ізоморфним до вкладення в концептуальні графи й екзистенціальні графи. Замість вкладених овалів Камп використовував блоки, зв'язані стрілками, а замість ліній ідентичності – змінні.

Концептуальний граф (conceptual graph) – логічна семантична мережа, запропонована Д. Совою (J. Sowa) у 1976–2000 р., вузли якої подають концепти і концептуальні відношення – мережа, у якій відношення вкладені усередину логічних вузлів. Мітки типів на вузлах концепцій роблять концептуальні графи типізованою або сортованою версією екзистенціальних графів, у яких овали екзистенціальних графів подані відповідно до концептів типу «Судження».

Різні версії логічних семантичних мереж мають різні синтаксичні механізми для з'єднання відносного змісту з логічними вузлами, але формальні правила перекладу можуть бути визначені для відображення однієї версії в іншу.

3. *Імплікаційні мережі* (implicational networks) – спеціальний різновид логічної семантичної мережі, у якій як первинне відношення для з'єднання вузлів використовується імплікація. Імплікаційні мережі можуть використовуватися для подання шаблонів довіри, причинних зв'язків або виведень. Інші

відношення можуть бути вкладені усередину логічних вузлів, але вони ігноруються процедурами виведення. У залежності від інтерпретації такі мережі можуть називатися мережами довіри, причинними мережами, байєсовськими мережами або системами доведення істинності.

Система доведення істинності (truth-maintenance system) – імплікаційна мережа, вузли якої подають деякі судження, значення істинності яких відомі, і судження, значення істинності яких мають бути визначені. Системи доведення істинності використовуються для поширення значень істинності з метою перевірки погодженості, пошуку протиріч або знаходження вузлів, що не містять очікуваних значень.

Причинна мережа (causal network) – імплікаційна мережа, у якій вузли подають типи випадків, а дуга від випадку e_1 до випадку e_2 указує, що e_1 – потенційна причина e_2. Розроблена Ч. Райегером (C. Rieger) у 1976 р. для аналізу описів проблем англійською мовою і перекладу їх у мережу, що могла б підтримувати судження на метарівні.

Мережа довіри (belief network) – імплікаційна мережа, у якій вузли подають довіри, а дуга від довіри b_1 до b_2, указує, що b_1 має на увазі b_2. Методи застосування статистики й імовірності в штучному інтелекті а також мережі довіри, що є причинними мережами, зв'язки яких позначені ймовірностями, розробив у 1988–2000 р. Ю. Перл (J. Pearl).

Різні методи виведення можуть застосовуватися до одного й того ж самого основного графа, іноді з подальшими анотаціями для вказівки значень істинності або імовірності.

Два основних підходи використовують для виведення.

– *Логіка*. Методи логічного виведення використовуються в системах доведення істинності. Система починає у вузлах, значення істинності яких є відомими і поширює їх по мережі. Комбінаціями прямих і зворотних виведень система поширює значення істинності до вузлів, значення істинності яких є відомими. Крім виведення нової інформації система може використовуватися, щоб перевірити послідовність, знайти протиріччя або знаходити місця розташування, де очікувані імплікації не містяться. Коли протиріччя знайдені, структура мережі може бути змінена шляхом додавання або видалення вузлів; результатом є різновид немонотонного виведення, називаний *переглядом довіри* (belief revision).

– *Імовірність*. Багато що з прямого і зворотного виведень, використовуваних із системою доведення істинності, може також бути пристосоване для імовірнісної інтерпретації, оскільки істинність може розглядатися як імовірність 1,0, а хибність як імовірність 0,0. Неперервний діапазон ймовірностей від 1,0 до 0,0, однак, підвищує потребу в більш тонких інтерпретаціях і збільшує складність обчислень. Найбільш детальне вивчення імовірнісних виведень у причинних або довірчих мережах було виконано Ю. Перлом, який аналізував

різні методи для застосування байєсівської статистики, щоб одержати причинну мережу за даним, що спостерігаються, і робити судження щодо них.

Хоча імплікаційні мережі підкреслюють імплікацію, вони здатні до вираження всіх булевих зв'язків, допускаючи кон'юнкцію входів до логічного вузла і диз'юнкцію виходів. Г. Гентзен (G. Gentzen) у 1935 р. показав, що набір імплікацій у цій формі міг би виражати всю логіку доказів. Загальна форма імплікації, написаної у формі тверджень Гентзена, має вигляд: $p_1, ..., p_n \rightarrow q_1, ..., q_m$, де $p_1, ..., p_n$ – антецеденти імплікації, $q_1, ..., q_m$ – консеквенти. *Узагальнене правило modus ponens* стверджує, що, коли кожний з антецедентів є істинним, принаймні один з консеквентів повинний бути істинним. У дійсності, коми в антецеденті мають ефект операторів АБО.

4. *Виконувані мережі* (executable networks) містять деякий механізм, на зразок передачі маркера або приєднаних процедур, що дозволяє здійснювати виведення, передавати повідомлення або шукати шаблони й асоціації.

Виконувані семантичні мережі містять механізми, що можуть здійснювати безпосередньо деякі зміни в мережі. Виконувані механізми відрізняють їх від мереж, що є статичними структурами даних і можуть тільки змінюватися через дію зовнішніх для мережі програм.

Такі види механізмів звичайно використовуються виконуваними семантичними мережами.

– Мережі, що передають повідомлення, можуть передавати дані від одного вузла до іншого. Для деяких мереж, дані можуть складатися з одного біта, називаного маркером, токеном (символом) або тригером; для інших, це може бути числова вага або довільно велике повідомлення.

– Приєднані процедури – програми, що містяться або зв'язані з вузлам, що виконують деяку дію або обчислення за даними в цьому вузлі або в деякому прилеглому вузлі.

– Перетворення графа – комбінують графи, змінюють їх, або розривають їх на менші графи. У типовому доказі теорем, такі перетворення виконуються програмою, зовнішньою до графа. Коли вони викликані графами безпосередньо, вони поводяться подібно хімічним реакціям, що комбінують молекули або ламають їх на частині.

Ці три механізми можуть бути об'єднані різними способами. Повідомлення, що передаються від вузла до вузла, можуть бути оброблені відповідно до процедур, приєднаних до цих вузлів, і перетворення графа можуть також бути викликані повідомленнями, що з'являються в деяких з вузлів.

Важливий клас виконуваних мереж був натхненний роботою фізіолога О. Селза (O. Selz), який запропонував схематичне сподівання (schematic anticipation) як спрямований метою метод зосередження розумового процесу на задачі заповнення порожніх слотів у шаблоні або схемі.

Графи потоків даних (dataflow graph) – найпростіші мережі з приєднаними процедурами, що містять пасивні вузли, які містять дані, і активні вуз-

ли, які подають функції, що беруть дані від вхідних вузлів і надсилають обчислені результати вихідним вузлам.

Мережа Петрі (Petri net) – різновид графів потоків даних, запропонована К. А. Петрі у 1962 р., містить пасивні вузли, називані *місцями* (places), що містять дані, називані *токенами* (tokens), і активні вузли, називані *переходами* (transitions), що видаляють токени з їхніх вхідних вузлів і поміщають токени в їхні вихідні вузли. Крім того, вони мають набір правил для *маркірування* місць точками (токенами, символами) і для *виконання* (firing) або виключення переходів. Правила для виконання переходів роблять мережі Петрі формально еквівалентними версії лінійної логіки. Кожне місце в мережі Петрі є попередньою умовою для переходів, що використовують його як вхід, і постумовою для переходів, що використовують його як вихід. Символ у місці стверджує, що відповідна умова (стан) є істинною. Видаляючи символ з кожного місця входу, виключення переходу скасовує твердження попередніх умов. Додаючи символ до кожного місця виходу, виключення стверджує, що кожна з постумов стала істинною.

Хоча графи потоків даних і мережі Петрі звичайно не називаються семантичними мережами подібні методи були втілені в *процедурних семантичних мережах* (procedural semantic networks). Дж. Милопоулос (J. Mylopoulos) та ін. створили ряд семантичних мереж із прикладеними процедурами. Їхня система містить означальні мережі для визначення класів, доказові мережі для установлення фактів, і процедури, подібні до методів об'єктно-орієнтованих мов програмування.

Крім маркерів і процедур, третій метод створення виконуваних мереж полягає в тому, щоб дозволити їм рости і змінюватися динамічно. Пеірс та Селз можуть розглядатися як ініціатори цього підходу.

5. *Мережі, що навчаються* (learning networks) будують або розширюють свої уявлення, здобуваючи знання за прикладами. Нове знання може змінювати стару мережу за допомогою додавання і видалення вузлів і дуг або зміни числових значень, називаних *вагами*, зіставлених вузлам і дугам.

Система, що навчається, природна або штучна, опановує нову інформацію, змінюючи внутрішні подання в напрямку, що дозволяє системі більш ефективно відповідати навколишньому середовищу.

Системи, що використовують мережні подання, можуть змінювати мережі такими способами.

– *Занесення в пам'ять* – найпростіша форма навчання – перетворення нової інформацій у мережу і додавання її до поточної мережі без будь-яких подальших змін.

– *Зміна ваг*. Деякі мережі мають числа, називані вагами, зіставлені вузлам і дугам. У імплікаційній мережі, наприклад, ваги можуть подавати імовірності і кожна поява мережі того самого типу збільшить оцінку імовірності її повторення.

– *Реструктурування* – найбільш складна форма навчання, що здійснює фундаментальні зміни в структурі мережі. Оскільки кількість і види структурних

змін необмежені, вивчення і класифікація методів реструктурування найбільш важкі, але потенційно найбільш корисні, якщо гарні методи будуть знайдені.

Занесення в пам'ять має величезне комерційне значення, оскільки світова економіка залежить від точного збереження записів. Для таких додатків інформація іноді зберігається в таблицях реляційних баз даних, але також використовуються і мережі. Для кращої ефективності і застосовності, більшість систем баз даних додають індекси, щоб прискорити пошук, і вони підтримують мови запитів, типу SQL, що виконують перетворення, щоб витягати і комбінувати інформацію, необхідну щоб відповісти на запит. Оскільки система, що навчається, повинна бути здатна відрізняти загальні особливості і виключення серед подібних прикладів, їй необхідна здатність вимірювати подобу і шукати в базі даних мережі, що є подібними, але не ідентичні будь-якому даному прикладу.

Системи, що навчаються шляхом запам'ятовування або зміни ваг можуть використовуватися самостійно, а системи, які навчаються за допомогою зміни структури мережі, звичайно використовують один або два інших методи як засоби реструктуризації.

Нейронні мережі – широко використовувана технологія для навчання за допомогою зміни ваг, зіставлених вузлам або дугам мережі. На рис. 2.1 зображено *багатошарову нейронну мережу прямого поширення сигналу*, на входи якої подається послідовність чисел, що відбивають відносну пропорцію деяких відібраних ознак, а на виході мережа формує іншу послідовність чисел, що вказують найбільш ймовірний концепт, який характеризується цією комбінацією ознак.

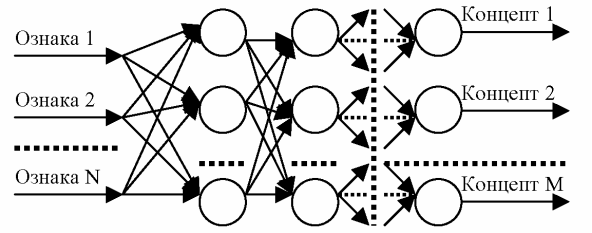


Рисунок 2.1 – Багатошарова нейрона мережа

У типової нейронної мережі структура вузлів і дуг є фіксованою і єдині зміни, що можуть відбуватися – зміни значень ваг дуг. Коли мережі пред'являється новий набір вхідних значень, він обробляється з урахуванням ваг дуг між вузлами мережі для визначення значень виходів. На стадії навчання система визначає, чи є отримані значення правильними, і методами, що засновані на зворотному поширенні помилки (error backpropagation) прагне відрегулювати ваги на дугах, що ведуть до результату.

Занесення в пам'ять найкраще підходить для додатків, що вимагають точного виправлення первісних даних, а методи зміни ваг краще підходять для розпізнавання образів. Для більш універсальних і творчих видів навчання необхідний деякий спосіб реструктурування мережі. Але кількість параметрів для реорганізації мережі є настільки значною, що повний діапазон можливостей у значній мірі є недослідженим.

6. *Гібридні мережі* (hybrid networks) – семантичні мережі, що підтримують дві і більше функції, такі як твердження, визначення, виведення, обчислення або навчання в одній мережі чи в декількох окремих, але близько взаємодіючих мережах.

Побудова та використання семантичних мереж

Процес побудови семантичних мереж вимагає ретельного *аналізу предметної області*. Для формалізації та фіксації окремих результатів процесу аналізу предметної області розробляють короткий опис предметної області та словник предметної області. Також у процесі аналізу предметної області виконують узагальнення понять, результати якого описують та враховують при побудові семантичної мережі.

Основним *принципом організації банків знань на основі семантичних мереж* є поділ екстенсіональних й інтенсіональних знань. При цьому екстенсіональна семантична мережа є основою бази даних, а інтенсіональна – бази знань (рис. 2.2).

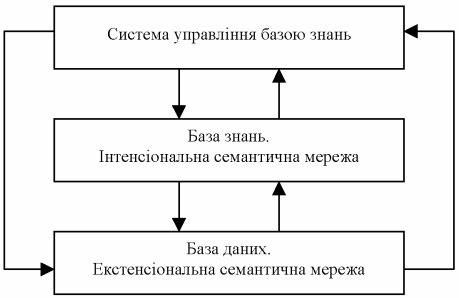


Рисунок 2.2 – Структура системи подання знань

У семантичній мережі можуть бути продані такі види об'єктів, як поняття, події, спеціалізовані методи рішення. Опис об'єктів предметної області виводиться на природно-мовний рівень. Збільшення номенклатури об'єктів знижує однорідність мережі і призводить до необхідності збільшення арсеналу методів виведення.

Багатомірність семантичних мереж дозволяє подавати в них числені семантичні відношення, що зв'язують окремі поняття, поняття і події в речення, а також речення в текстах; крім того, у семантичній мережі може бути відбита семантична ієрархія взаємної підпорядкованості спеціалізованих методів.

Усі знання, включаючи факти, що знову надходять, а також деякі спеціалізовані методи рішення накопичуються у відносно однорідній структурі пам'яті.

На мережах визначається ряд більш-менш уніфікованих семантичних відношень між об'єктами, яким відповідають уніфіковані методи виведення.

Методи виведення в сукупності з цілями (запитами) визначають області семантичного знання, що має відношення до поставленої задачі, формулюючи акт розуміння запиту і певний ланцюг виведень і неповних виведень, що відповідають рішенню задачі.

На кожній стадії формування рішення можна чітко розділити повне знання системи (повна семантична мережа) і поточне знання – збуджена область мережі, у якій здійснюються певні операції (процес розуміння, виведення і т. д.).

Десигнат є важливим поняттям у семантичних мережах. Це унікальне внутрисистемне ім'я, що ставиться у відповідність деякому об'єкту предметної області, якщо про нього в даний момент немає повної інформації. Десигнат відбиває найбільш значимий зміст об'єкта, зокрема, просто факт його існування. Порожні місця заповнюються в процесі нагромадження даних значеннями відповідного атрибута. Таким чином, у момент першого згадування про об'єкт у базі знань заводиться його десигнат, яким можна маніпулювати, не очікуючи повної інформації про об'єкт. Уведення десигнатів збільшує гнучкість семантичної мережі, однак їхнє використання вимагає додаткових обчислювальних ресурсів.

Операції модифікації бази знань на семантичних мережах зводяться до видалення і додавання нових вершин і ребер.

Проблема *пошуку рішення* в базі знань типу семантичної мережі зводиться до задачі пошуку фрагмента мережі, відповідного деякій підмережі, що відображає поставлений запит до бази знань.

Особливістю семантичної мережі як механізму подання знань є єдина база знань і механізм виведення.

Базові *операції пошуку інформації* в мережі забезпечують пошук вершини чи ребра за іменем, переходячи від однієї вершини до іншої по зв'язках і від одного зв'язку до іншої через суміжні вершини. Мета пошуку – одержання знань, поданих у мережі і необхідних для вирішення задачі.

Операція зіставлення зі зразком є одним з потужних засобів маніпуляції знаннями. Сутність її полягає в такому. Інформаційна потреба, що визначає зміст і ціль запиту до бази знань, описується автономною семантичною мережею – *мережею запиту*, яка побудована за тими ж правилам і відображує ті ж об'єкти і відношення, що подані в системі знань семантичною мережею.

Пошук відповіді на запит реалізується зіставленням мережі запиту з фрагментами семантичної мережі, що подає систему знань. Позитивний результат зіставлення дозволяє одержати одну з відповідей на запит. Усі відповіді можна одержати шляхом виявлення всіх порівнянних з мережею запиту фрагментів.

Метод перехресного пошуку використовується в семантичних мережах нарівні з методом зіставлення. При цьому здійснюється пошук відношень

між поняттями. Відповідь на запит формується шляхом знаходження вершини, у якій перетинаються дуги, що йдуть з двох вершин.

Метод поширення активності та техніки перетинань – особливий тип генерації висновку, використовуваний у семантичних мережах. Цей метод відіграє важливу роль в обробці контекстів. Процес здійснюється побудовою ланцюжків виведення на основі уведених висловлень у всіх напрямках доти, поки не виявиться перетинання де-небудь у мережі.

Перевагами семантичних мереж є:

– великі виразні можливості мережних моделей;

– зручність та логічна прозорість – опис подій і понять проводиться на рівні, дуже близькому до природної мови;

– забезпечується можливість зчеплення різних фрагментів мережі;

– відношення між поняттями і подіями утворюють досить невелику і добре формалізовану множину;

– можливість відображення структури, що властива знанням, оскільки відношення можуть бути явно специфіковані;

– для кожної операції над даними і знаннями можна виділити з повної мережі, що подає всю семантику, деяку частину мережі (фрагмент), яка охоплює необхідні в даному запиті смислові характеристики;

– розміщення в базі знань навколо відповідної вершини всієї точно відомої інформації тієї чи іншої концепції;

– ефективний інформаційний пошук, оскільки асоціації між об'єктами визначають шляхи доступу, що проходять по базі знань.

Недоліками семантичних мереж є:

– складність організації процедури пошуку у процесі виведення на семантичній мережі;

– відсутність достатньо розробленої денотаційної і, особливо операційної семантики. Остання, як правило, використовує прості розуміння, пов'язані з реалізацією методів пошуку по шляхах доступу. Числені формалізми, що підтримують мережні схеми, не мають необхідного рівеня повноти синтаксису і семантики;

– програш семантичних мереж у поданні чисто структурних відношень, які легко реалізуються численням предикатів (логічні зв'язування, квантори спільності й існування) або процедуральному поданні (рівнобіжні процеси, динамічні події);

– менша виразна сила семантичних мереж у порівнянні з логікою предикатів. Зокрема, має певну складність відображення квантифікаторів. Деякі недоліки можуть бути усунуті за допомогою реалізації механізму спадкування: субконцепти успадковують властивості суперконцептів, якщо тільки це явно не заборонено. Як засоби подолання недоліків пропонувалося також уведення функцій Сколема в семантичних мережах, поділ семантичних мереж на частини. Остання пропозиція виявляється корисною для відображення

в системі як самих планів, цілей і гіпотез, так і пропозицій системи про плани, цілях і гіпотези передбачуваного партнера.

2.3.5 Фреймові моделі

Певною незручністю при використанні семантичних мереж є довільність їхньої структури і наявність різних типів вершин і зв'язків. Ця розмаїтість вимагає розмаїтості процедур обробки інформації, яка міститься в семантичній мережі, що приводить до ускладнення програмного забезпечення і зниження швидкості обробки.

Одним зі шляхів вирішення цієї проблеми за рахунок додавання мережі властивості регулярності є використання апарата фреймів. Фреймовий підхід до подання знань є окремим випадком підходу, заснованого на семантичних мережах.

Термін «фрейм» (від англійського frame, що означає «каркас» або «рамка») був запропонований М. Мінським у 70-х рр. XX століття.

Фрейм – це структура даних, призначена для подання стереотипних (стандартних) ситуацій. Фрейм є системно-структурним описом предметної області (події, явища, ситуації, стану і т. п.), що містить на підставі її семантичних ознак порожні рольові позиції (слоти), які після заповнення конкретними даними перетворюють фрейм у носій конкретного знання про предметну область.

Фрейм можна розглядати як фрагмент семантичної мережі, змістовно виражений структурою даних із при'єднаними процедурами обробки цих даних, призначений для опису об'єкта (ситуації) проблемного середовища з усією сукупністю властивих йому властивостей.

Основна ідея фреймового підходу до подання знань – більш тверде, ніж при підході, заснованому на семантичній мережі, виділення об'єктів і ситуацій проблемного середовища й їхніх властивостей, тобто усе, що стосується об'єкта або ситуації і є важливим з позицій розв'язуваних задач, не «розмивається по мережі», а подається у фреймі.

Фрейм можна подати у вигляді таблиці, дерева, формули.

Табличне подання фрейму.

Ім'я фрейму					
Ім'я слота	Покажчик спадкування	Покажчик типу	Значення слота	Приєднана процедура	
				Слуга	Демон
слот 1					
слот 2					
. . . . .					
слот N					

Формульне подання фрейму: (f {<S>, <v_1, g_1, [p_1]>, <v_2, g_2, [p_2]>, ..., <v_k, g_k, [p_k]>}), де f – ім'я фрейму, S – слот з родовим або суперпоняттям, v_i – ім'я

слота, g_i – значення слота, p_i – процедура. Слоти $<v_i, g_i, [p_i]>$ описують властивості, що відрізняють даний об'єкт від родового поняття.

Імена фреймів використовуються як мнемонічні елементи для конструювання мережі фреймів.

Слот (валентність) – це складова частина фрейму, яка може бути заповнена елементом даних конкретного типу для фіксації знань про об'єкт, якому відведено даний фрейм. Іншими словами слот – це атрибут, зв'язаний з вузлом у системі, заснованій на фреймах.

Кожний фрейм складається з довільного числа слотів.

Найважливішою властивістю теорії фреймів є запозичення з теорії семантичних мереж – так зване *спадкування властивостей.* І у фреймах, і в семантичних мережах спадкування відбувається за IS-A (АКО)-зв'язками. Слот IS-A (АКО) указує на фрейм більш високого рівня ієрархії, звідкіля неявно успадковуються, тобто переносяться, значення аналогічних слотів. При зображенні фреймів у вигляді таблиці родові поняття знаходяться на верхньому рівні.

Слоти мають такі параметри: ім'я слота, покажчик спадкування, покажчик типу, значення слота, приєднана процедура.

Ім'я слота – унікальний ідентифікатор слота у фреймі, до якого він належить. Ім'я слота в деяких випадках може бути службовим та вказувати на зміст фрейму, наприклад, на успадкування, на користувача, на дату фрейму, містити коментар.

Покажчик спадкування показує, яку інформацію про атрибути слотів у фреймі верхнього рівня успадковують слоти з тими ж іменами у фреймі нижнього рівня. Покажчики спадкування стосуються тільки фреймових систем ієрархічного типу, заснованих на відношенні «абстрактне-конкретне».

Типові покажчики спадкування:

- S (same – той самий) – слот успадковується з тими ж значеннями даних;
- U (unique – унікальний) – слот успадковується, але дані в кожнім фреймі можуть приймати будь-які значення, тобто сам слот у фреймі-нащадку зберігається, але значення слота не успадковується;
- I (independent – незалежний) – слот не успадковується;
- R (range – діапазон) указує, що значення слотів вищих фреймів визначають межі фреймів-нащадків, тобто значення слота повинні знаходитися в межах інтервалу значень, зазначених в однойменному слоті батьківського фрейму;
- O (override – анулювати) – вказує, що слот не має значення.

Покажчик типу даних визначає тип даних значення слота. Типом даних, що включаються в слот, можуть бути:

- FRAME – покажчик імені фрейму верхнього рівня;
- ATOM – змінна;

- TEXT або STRING – текстова (строкова) інформація;
- INTEGER – ціле значення;
- REAL – дійсне значення;
- BOOL або BOOLEAN – логічне значення;
- LIST – список;
- LISP – приєднана процедура;
- TABLE – таблиця;
- EXPRESSION – вираз.

Значення слоту повинне збігатися з зазначеним типом даних цього слота, крім того, повинна виконуватися умова спадкування.

Існує кілька способів одержання слотом значень у фреймі-екземплярі:

- за замовчуванням від фрейму-зразка (Default-значення);
- через спадкування властивостей від фрейму, зазначеного в слоті IS-A (АКО);
- за формулою, зазначеній у слоті;
- через приєднану процедуру;
- явно з діалогу з користувачем;
- з бази даних.

Як значення слота може виступати ім'я іншого фрейму, так утворюються мережі фреймів. Для економії однакові значення однойменних слотів вказуються тільки у фреймах-нащадках.

Агрегат (*фасет*) – це деякий символьний або текстовий об'єкт, який подається як єдине ціле.

Приєднані до слоту процедури (*зв'язані процедури*) прийнято поділяти на два типи: процедури-демони і процедури-слуги.

Процедура-демон – це прихована або віртуальна приєднана процедура, яка автоматично виконується при наявності певних умов, при певних змінах бази знань, наприклад, коли не встановлене значення слоту, до якого відбувається звертання; коли значення слоту стирається і коли в слоті підставляється його значення. Процедури-демони активізуються при кожній спробі додавання чи видалення даних зі слоту (за замовчуванням). За допомогою процедур цього типу автоматично виконуються, зокрема, усі рутинні операції, пов'язані з веденням баз даних і знань.

З кожним слотом може бути пов'язана одна або декілька процедур, які виконуються, коли змінюються значення слотів. Частіше за все зі слотами зв'язуються процедури:

- якщо додано (IF-ADDED) – запускається, коли нова інформація заноситься до слоту (при спробі зміни значення слоту);
- якщо видалено (IF-REMOVED) – запускається при спробі видалення інформації зі слоту;

– якщо треба (IF-NEEDED) – запускається, якщо у момент звертання до слоту його значення не було установлене (слот є пустим).

Процедури-слуги активізуються тільки при виконанні умов, визначених користувачем при створенні фрейму.

У залежності від стану слотів можуть бути фрейми: прототипи і екземпляри.

Фрейми-зразки (*фрейми-прототипи, протофрейми, фрейм-інтенсіонали*) – це фрейми, в яких частина або всі значення слотів відсутні – інтенсіональний опис певної множини фреймів-прикладів.

Фрейм-екземпляр (*фрейм-приклад*) – це фрейм, в якому слоти заповнені конкретними значеннями, що являють собою описи, – екстенсіональний опис відповідного фрейму-прототипу. Фрейми-екземпляри створюються для відображення реальних фактичних ситуацій на основі даних, що надходять.

Під *підстановкою* (значенням) *фрейму* розуміється результат, що виходить за допомогою заміни кожної типової змінної фрейму константою того ж типу.

Фреймові системи поділяються на *статичні* і *динамічні*, останні допускають зміну фреймів у процесі рішення задачі.

У *залежності від функціонального призначення* фрейми поділяються на фрейми-описи та рольові фрейми.

Фрейм-опис моделює властивості або ситуації. У якості імен слотів використовують назви атрибутів (ознак), що описують ситуацію.

У *рольовому фреймі* для імен слотів використовують назви ролей, сукупність яких визначає зміст, що приписується всьому фрейму. Роллю можуть бути об'єкти, над якими виконуються дії. Як правило, у рольовому фреймі як імена слотів виступають питальні слова, відповіді на які є значеннями слотів.

Взаємопов'язані фрейми сполучаються в *мережу фреймів*, яка описує предметну область. У мережі фреймів органічно поєднуються декларативні та процедурні знання.

Мережне подання рольового фрейму складається з кореня, рольових дуг, вузлів і вершин аргументів конкретного типу (дуги з мітками ролі).

Фрейми мають властивість *вкладеності*, тому що значенням слоту може виступати система імен слотів нижнього рівня.

Оскільки концептуальному поданню властива *ієрархічність*, цілісний образ знань будується у виді однієї фреймової системи, що має ієрархічну структуру.

Властивість вкладеності, можливість значень слотів посилатися на інші слоти того ж фрейму забезпечують *структурованість, послідовність і зв'язність знань*. Значення, які зберігаються у фреймах, мають характер посилань і тому є *внутрішньо інтерпретованими* (це одна з властивостей знань). З іншого боку, фрейми не пристосовані до подання структурних знань.

Елементи фреймових структур можуть зв'язуватися двома *типами відношень* – відношенням «абстрактне – конкретне» і «ціле – частина».

Відношення *«абстрактне – конкретне»* характерні тим, що на верхніх рівнях розташовані абстрактні об'єкти (*концепти*), а на нижніх – конкретні об'єкти, причому об'єкти нижніх рівнів успадковують атрибути об'єктів верхніх рівнів. Ці відношення називаються також відношеннями типу IS-A чи KIND-OF.

Відношення *«ціле – частина»* (PART-OF) стосується структурованих об'єктів і показує, що об'єкт нижнього рівня є частиною об'єкта верхнього рівня. При такім відношенні спадкування властивостей не відбувається.

Пошук за зразком є основною операцією для фреймових моделей. *Зразок* являє собою фрейм, у якому заповнено не всі структурні одиниці, а тільки ті, за якими серед фреймів, що зберігаються в пам'яті ЕОМ, будуть відшукуватися потрібні фрейми. Зразок може, наприклад, містити ім'я фрейму, а також ім'я деякого слота у фреймі з указівкою значення слота. Такий зразок перевіряє наявність у пам'яті ЕОМ фрейму з даним ім'ям і даним значенням слота, зазначеним у зразку. Якщо в зразку зазначено ім'я деякого слота і його значення, то процедура пошуку за зразком забезпечує вибірку усіх фреймів, у яких міститься слот з таким ім'ям і таким значенням слота, як у зразка. Нарешті, може бути задана деяка логічна функція від імені фрейму, якихось імен слотів і значень слотів.

Процедури поповнення слотів даними, а також введення в систему нових фреймів-прототипів і нових зв'язків між ними є іншими процедурами, характерними для фреймових моделей.

Процес зіставлення – процес, у ході якого перевіряється правильність вибору фрейму. Звичайно цей процес здійснюється відповідно до поточної мети та інформації (значень), що міститься в даному фреймі. Іншими словами, фрейм містить умови, що обмежують значення слота, а мета використовується для визначення, яка з цих умов, маючи відношення до даної ситуації, є релевантною.

У результаті процес зіставлення фрейму здійснюється в такий спосіб.

Спочатку за допомогою припущення й інтуїції вибирається деякий базовий фрейм, і за допомогою знань, заснованих на виявлених особливостях, релевантності чи за допомогою підфреймів, передбачуваних як найбільш релевантні, даний фрейм підтверджує або не підтверджує свою релевантність. При цьому відповідно до поточної мети визначається, яке обмеження слота варто використовувати при зіставленні. При підтвердженні процес зіставлення завершується. У противному випадку, якщо в даному фреймі є слот, у якому виникла помилка, що стосується, наприклад, умови погодженості з інформацією, заданої за замовчуванням, то необхідна інформація, що забезпечує присвоювання належного значення даному слоту. Присвоювання необхідної інформації даному слоту повинне не суперечити обмеженням цілісності і сподіванням.

В інтелектуальних системах із фреймовим поданням знань використовуються три способи *керування логічним виведенням*: демони, приєднані процедури і механізм спадкування. Останній можна назвати єдиним основним механізмом виведення, яким оснащені фреймові (об'єктно-орієнтовані) системи.

Управлінські функції *механізму спадкування* полягають в автоматичному пошуку і визначенні значень слотів фреймів нижче розташованих рівнів за значеннями слотів фреймів верхніх рівнів, а також у запуску приєднаних процедур і демонів.

Приєднані процедури і демони дозволяють реалізувати будь-який механізм виведення в системах із фреймовим поданням знань. Однак ця реалізація має конкретний характер і вимагає значних витрат праці проектувальників і програмістів.

Механізм керування виведенням може бути організований у такий спосіб. Зв'язки між даним фреймом і іншими фреймами задаються за допомогою спеціального слота, значенням якого є приєднана процедура – специфічна процедура виведення в цьому фреймі. При здійсненні виведення спочатку запускається одна з приєднаних процедур деякого фрейму. Потім оцінюється значення, що повертається, і в залежності від нього послідовно запускаються приєднані процедури інших фреймів. У ході цього процесу відбувається генерація і знищення слотів, зміна значень слотів і т. п. Таким чином, відбувається поступове просування до одержання цільового значення.

У рамках фреймового підходу передбачається, що знання в системі подаються у вигляді окремих підструктур (кластерів), що містять відомості про стереотипи (стандартні ситуації).

Відповідно до даного припущення розуміння ситуації для системи означає пошук у переліку накопичених структур такої структури, що щонайкраще описувала б розглянуту ситуацію. При цьому слоти заповнюються деякою інформацією і заповнений фрейм перевіряється на адекватність даній ситуації. У випадку розбіжності шукається новий фрейм і процес продовжується.

Таким чином, можна виділити два основних процеси, що відбуваються у фреймових системах:

1. *Створення екземпляра фрейму*. Для створення екземпляра фрейму необхідно знайти придатний фрейм і заповнити його слоти інформацією, що описує специфіку розглянутої ситуації. Для заповнення слотів використовується спеціальна інформація про те, як знайти потенційні «заповнювачі» слотів. Ця інформація часто зберігається в процедуральній формі.

2. *Активація фреймів*. У випадку, якщо фрейм вважається придатним для опису даної ситуації, здійснюється його активація глобальним процесом. Якщо виявляється занадто багато відмінностей умісту фреймів від специфічних особливостей розглянутої ситуації або вони носять досить серйозний характер, організується пошук іншого, більш придатного фрейму.

При цьому «відкинутий» фрейм може містити вказівки на те, які саме фрейми потрібно досліджувати замість даного (наприклад, більш загальні чи, навпаки, більш спеціалізовані).

Основною *перевагою фреймів* як моделі подання знань є те, що вони відображають концептуальну основу організації пам'яті людини, а також її гнучкість і наочність.

Подання знань на основі фреймової моделі, особливо ефективно для структурного опису складних понять і вирішення задач, у яких відповідно до ситуації бажано застосовувати різні способи виведення.

У той же час ускладнюється керування завершеністю і сталістю цілісного образа, що є *недоліком* фреймових моделей. Зокрема, з цієї причини існує велика небезпека порушення приєднаної процедури. Слід зазначити, що фреймову систему без механізму приєднаних процедур (а отже, і механізму пересилання повідомлень) часто використовують як базу даних системи продукцій.

2.3.6 Сценарії

Сценарій (script) – це структуроване подання, що описує стереотипну послідовність подій в окремому контексті. Сценарії спочатку були запропоновані Р. Шенком (R. Schank) і Р. Абельсоном (R. Abelson) у 1977 р. як засіб для організації структур *концептуальної залежності* в описах типових ситуацій. Сценарії використовуються в системах розуміння природної мови для організації бази знань у термінах ситуацій, які система повинна розуміти.

Сценарій включає такі компоненти:

– *початкові умови*, що повинні бути істинними при виклику сценарію;

– *результати* або факти, що є істинними, коли сценарій завершується;

– *припущення*, що підтримують контекст сценарію. Множина припущень описує прийняті за замовчуванням умови реалізації сценарію;

– *ролі* є діями, що виконують окремі учасники;

– *сцени* подають часові аспекти сценарію.

Елементи сценарію – основні «частини» семантичного значення – подаються відношеннями концептуальної залежності. Зібрані разом у фреймоподібній структурі вони подають послідовність значень або подій.

2.3.7 Логічні моделі

Логічні моделі в основі мають *формальну систему*, що задається четвіркою виду $M=<T, P, A, B>$, де T – *множина базових елементів* різної природи, P – множина *синтаксичних правил*, за допомогою яких з елементів T утворюють *синтаксично правильні сукупності*, у множині яких виділяється деяка підмножина A, елементи якої називаються *аксіомами*, B – множина *правил виведення*, застосовуючи які до елементів A, можна одержувати нові синтаксично правильні сукупності, до яких знову можна застосовувати правила з B.

Для множини *T* існує деякий спосіб визначення приналежності чи неприналежності довільного елемента до цієї множини. Процедура такої перевірки *П*(*T*) може бути будь-якою, але за кінцеве число кроків вона повинна давати позитивну чи негативну відповідь на питання, чи є *x* елементом множини *T*.

За допомогою процедури *П*(*P*) за кінцеве число кроків можна одержати відповідь на питання, чи є сукупність *X* синтаксично правильною.

За допомогою процедури *П*(*A*) для будь-якої синтаксично правильної сукупності можна одержати відповідь на питання про приналежність її до множини *A*.

Якщо є процедура *П*(*B*), за допомогою якої можна визначити для будь-якої синтаксично правильної сукупності, чи є вона виведеною, то відповідна формальна система називається *розв'язною*. Це показує, що саме правила виведення є найбільш складною складовою формальної системи.

Для знань, що входять у базу знань, можна вважати, що множина *A* утворює всі інформаційні одиниці, що введені в базу знань ззовні, а за допомогою правил виведення з них виводяться нові *похідні знання*. Іншими словами, формальна система являє собою генератор породження нових знань, що утворять множину *виведених* у даній системі *знань*. Ця властивість логічних моделей робить їх притягальними для використання в базах знань. Вона дозволяє зберігати в базі лише ті знання, що утворять множину *A*, а всі інші знання одержувати з них за правилами виведення.

Основна ідея логічного підходу полягає в тому, щоб розглядати всю систему знань, необхідну для вирішення прикладних задач, як сукупність фактів (тверджень). *Факти* подаються як *формули* в деякій логіці (першого чи вищого порядку, багатозначній, нечіткій чи ін.) Система знань відображається сукупністю таких формул і, подана в ЕОМ, вона утворює базу знань. Формули неподільні і при модифікації бази знань можуть лише додаватися або видалятися.

Логічні методи забезпечують розвинутий апарат виведення нових фактів з тих, котрі явно подані в базі знань. Основним примітивом маніпуляції знаннями є *операція виведення*.

Логічними моделями називаються описи предметних областей, виконані в логічних мовах. Логічні моделі, як правило, засновані на численні предикатів першого порядку, яке, у свою чергу, засновано на численні висловлень.

Висловлення – речення, зміст якого можна виразити значеннями: істина (1) або хибність (0).

Елементарні (прості) висловлення – висловлення, які не можна розділити на частині. Прості висловлення розглядаються цілком, без урахування їхньої суб'єктно-предикатної структури.

Складні висловлення – висловлення, які можна розділити на прості, котрі зв'язані між собою за допомогою логічних зв'язувань.

Логічні зв'язування (пропозиціональні зв'язування) – *k*-арні логічні операції ($k \geq 0$), які перетворюють той чи інший набір з k висловлень у висловлення,

а набір із пропозиціональних (висловлювальних) форм від деяких змінних (тобто виразів, що містять змінні і перетворюються у висловлення при заміні змінних конкретними висловленнями) – у пропозиціональну форму від цих змінних.

У формалізованих логіко-математичних мовах логічні зв'язування виконують функції, аналогічні функціям союзів і союзних слів у природних мовах.

В алгебрі логіки логічні зв'язування розглядають як булеві функції, тобто операції на множині з двох значень {істина, хибність} або {1 та 0} (див. табл. 2.2).

Числення висловлень – логічне числення, яке визначає логічні закони, що характеризують логічні зв'язування. Воно характеризується двозначною інтерпретацією і в ньому доказові всі тотожньо-істинні формули даного числення і тільки вони. Числення висловлень входять як частина до числення предикатів.

Розроблено багато різновидів класичного числення висловлень, що відрізняються одне від одного вибором логічних зв'язувань, аксіом і правил виведення.

Числення висловлень дозволяє формалізувати лише малу частину множини суджень, оскільки цей апарат не дозволяє враховувати внутрішню структуру висловлення, що існує в природних мовах.

Предикат від *n* змінних (*n*-арний предикат) на множині *A* – *n*-місцева функція, визначена на множині *A* зі значеннями в множині {істина, хибність}. Під предикатом розуміють деякий зв'язок, що задано на наборі змінних.

Логіка предикатів першого порядку – розділ математичної логіки, що вивчає логічні закони, загальні для будь-якої непорожньої області об'єктів із заданими в цій області предикатами. Ці закони формулюються в спеціальній формальній логічній мові першого порядку, основними синтаксичними одиницями якої є константи, змінні, функції, предикати, квантори і логічні зв'язування.

Формальний синтаксис числення предикатів першого порядку зручно подати в нормальній формі Бекуса-Наура, яка традиційно застосовується для запису граматик мов програмування.

<константа> → <ідентифікатор1>

<змінна> → <ідентифікатор2>

<функція> → <ідентифікатор3>

<предикат> → <ідентифікатор4>

<терм> → <константа> | <змінна> | <функція> (<список термів>)

<список термів> → <терм> | <терм>, <список термів>

<атом> → <предикат> | <предикат> (<список термів>)

<літера> → <атом> | ¬ <атом>

<оператор> → ∧ |∨ |→ |↔

<список змінних> → <змінна> | <змінна>,<список змінних>

<квантор> → <(∃ <список змінних>) | < (∀ <список змінних>)

<формула>→ <літера> | <формула> | <квантор> (<формула>) | (<формула>) <оператор> (<формула>)

Таблиця 2.2 – Логічні зв'язування

Назви	Позначення	Таблиця істинності					Як читати
		x_1	0	0	1	1	
		x_2	0	1	0	1	
Заперечення, доповнення	$\neg x_1$, $\bar{x}_1$		1	1	0	0	не x_1
Кон'юнкція, логічний добуток, логічне «ТА», функція збігу	$x_1 \,\&\, x_2$, $x_1 \wedge x_2$, $x_1 \cdot x_2$, $x_1 x_2$		0	0	0	1	x_1 та x_2
Диз'юнкція, логічна сума, логічне «АБО», функція поділу	$x_1 + x_2$, $x_1 \vee x_2$		0	1	1	1	x_1 або x_2; або x_1, або x_2, або x_1 та x_2
Матеріальна імплікація	$x_1 \rightarrow x_2$, $x_1 \supset x_2$		1	1	0	1	якщо x_1, то x_2; x_1 тягне x_2; x_1 імплікує x_2
Еквівалентність, функція рівнозначності	$x_1 \equiv x_2$, $x_1 \sim x_2$, $x_1 \leftrightarrow x_2$		1	0	0	1	x_1 тоді і тільки тоді коли x_2; x_1 еквівалентно x_2
Сума за модулем 2, розділова диз'юнкція, заперечення еквівалентності, функція нерівнозначності	$x_1 \oplus x_2$, $x_1 \dot{\vee} x_2$		0	1	1	0	або x_1, або x_2; x_1 не еквівалентно x_2
Штрих Шефера, заперечення кон'юнкції, антикон'юнкція	x_1 / x_2		1	1	1	0	x_1 та x_2 несумісні; невірно, що x_1 та x_2
Стрілка Пірса, заперечення диз'юнкції, антидиз'юнкція, функція Вебба	$x_1 \downarrow x_2$		1	0	0	0	ні x_1, ні x_2
Заперечення матеріальної імплікації, матеріальна антиімплікація	$x_1 \not\rightarrow x_2$, $x_1 \not\supset x_2$		0	0	1	0	x_1, але не x_2; невірно, що x_1 тягне x_2
Зворотна імплікація	$x_1 \leftarrow x_2$, $x_1 \subset x_2$		1	0	1	0	x_1, якщо x_2; якщо x_2, то x_1; x_2 тягне x_1
Заперечення зворотної імплікації, зворотна антиімплікація	$x_1 \not\leftarrow x_2$, $x_1 \not\subset x_2$		0	1	0	0	не x_1, але x_2; невірно, що x_2 тягне x_1

У даному записі будь-яке ім'я в кутових дужках являє собою тип синтаксичного об'єкта. Визначення кожного типу починається з появи його імені в лівій частині кожного запису, тобто ліворуч від знака →. У правій частині кожного запису наводяться можливі способи організації синтаксично коректних об'єктів обумовленого типу. Альтернативні варіанти розділені знаком |,

який можна інтерпретувати як «або». Номера ідентифікаторів варто трактувати в тому змісті, що ідентифікатори, використовувані для позначення об'єктів різних типів, повинні бути помітними.

Функції, як і предикати, задають деякий зв'язок між змінними або константами. Але цей зв'язок або відношення не характеризуються істинносним значенням. Функції можуть бути аргументами предикатів (тобто *термами*), а предикати – ні.

Для визначення області дії змінних у логіці предикатів використовують квантори.

Квантори – логічні оператори, що переводять одну висловлювальну форму в іншу і дозволяють вказувати обсяг тих значень предметних змінних, для яких ця висловлювальна форма є істинною. У формалізованих логіко-математичних мовах для вираження зазначених кількісних характеристик найчастіше використовуються квантори, наведені у табл. 2.3.

Таблиця 2.3 – Логічні квантори

Назва	Позначення	Читається як
Квантор спільності	$\forall x$, (x), (Ax), $\cap_x$, $\wedge_x$, $\prod_x$	для всякого x ..., для всіх x ...
Квантор існування	$\exists x$	існує x
Квантор одиничності	$!x$	єдиний x

Зв'язані змінні – змінні в логічних формулах, що знаходяться в сфері дії кванторів.

Вільні змінні – змінні в логічних формулах, що знаходяться поза сферою дії кванторів.

Для того щоб можна було говорити про істинність якого-небудь твердження без підстановки значень у змінні, усі вхідні в нього змінні повинні бути зв'язані кванторами.

Якщо в логічну формулу входить кілька кванторів, необхідно враховувати їхнє взаємне розташування. Для усунення цієї невизначеності введемо дужки і порядок застосування кванторів – зліва-направо.

Операції і квантори в логіці предикатів мають неоднакові пріоритети. У порядку убування пріоритету: $\forall x, \exists x$, $\neg$, $\wedge, \vee$, $\rightarrow$, $\leftrightarrow$.

В одній логічній формулі не допускається застосування різних кванторів до однієї змінної.

У логіці предикатів першого порядку не дозволяється застосування кванторів до предикатів (більш високі порядки це дозволяють).

Речення – формула, у якій усі змінні зв'язані. Кожному реченню можна поставити у відповідність визначене значення – істина або хибність.

Складні формули в логіці предикатів отримують шляхом комбінування атомарних формул за допомогою логічних операцій.

Правильно побудовані логічні формули – формули, отримані з атомарних формул за допомогою логічних операцій.

Область інтерпретації формули – множина усіх можливих значень термів, що входять у формулу.

Інтерпретація правильно побудованих формул можлива тільки з урахуванням конкретної області інтерпретації. Для подання знань конкретної предметної області у вигляді правильно побудованої формули необхідно насамперед установити область інтерпретації, тобто вибрати константи, що визначають об'єкти в даній області, а також функції і предикати, що визначають залежності і відношення між об'єктами. Після цього можна побудувати логічні формули, що описують закономірності даної предметної області.

Розглянемо *логічну мову*, що називається *багатосортним численням предикатів першого ступеня* (багатосортною логікою першого порядку).

У логічних моделях, заснованих на численні предикатів, класам сутностей предметної області відповідають так називані *сорти* (типи). Точніше, у логічних моделях беруть участь імена сортів, що інтерпретуються як класи сутностей.

У загальному випадку *сигнатурою* називається множина виразів виду p: $A_1 \times A_2 \times ... \times A_n \rightarrow B$, де A_j, B – сорти, а p – функція або предикат.

Ім'я р є ім'ям предиката у виразі, якщо $B=T$, де T – особливий сорт у тому змісті, що він завжди інтерпретується як множина логічних значень {0,1} чи {«істина», «хибність»}. Сорти сигнатури інтерпретуються як множини. Таким чином, предикат розглядається як функція, задана на об'єктах (узагалі говорячи, різних сортів) і приймаюча значення «істина» або «хибність».

Предикат можна розглядати також як багатомісне відношення між об'єктами (різних) сортів: об'єкти e_1, e_2, ..., e_n знаходяться у відношенні p, що записується як $p(e_1, e_2, ..., e_n)$ тоді і тільки тоді, коли $p(e_1, e_2, ..., e_n) = 1$. Помітимо, що для двомісних предикатів часто використовується інфіксний запис: замість $p(e_1, e_2)$ пишуть $e_1 p e_2$.

Сигнатура задає структурні зв'язки між поняттями предметної області, поданими предикатами і функціями. Логічні зв'язки між цими поняттями задаються формулами, що записуються в сигнатурі. Структурні і логічні зв'язки виражають деяке знання про предметну область. Таким чином, сигнатура формально подає одну частину знання про предметну область, а формули, записані в цій сигнатурі, подають іншу частину знання.

Перевагами логічного підходу є:

– наявність логічних формалізмів – головним чином числення предикатів, що має ясну формальну семантику й операційну підтримку в тому змісті, що для нього розроблені механізми виведення. Тому числення предикатів

було першою логічною мовою, яку застосували для формального опису предметних областей, пов'язаних з вирішенням прикладних задач;

– забезпечення простої та ясної нотації для запису фактів, що має чітко визначену семантику (принаймні для методів, заснованих на традиційній логіці першого порядку). Кожний факт подається в базі знань тільки один раз, незалежно від того, як він буде використовуватися надалі. База знань, розроблена з застосуванням логічних методів, як правило, є досить простою для розуміння;

– високий рівень модульності знань і одночасно можливість одержання єдиної системи подання, у якій логічно роз'ясняються властивості знань як єдиного цілого. Отже, за допомогою логіки предикатів можна, визначаючи довільним чином знання, з'ясувати, чи є або відсутні протиріччя між новими і вже існуючими знаннями.

Недоліками логічного підходу є:

– неможливість використання природної мови в системах машинного подання знань через відсутність формальної семантики природної мови, що мала б досить ефективну операційну підтримку, а також властивих їй нерегулярностей, двозначностей і пресупозицій;

– відсутність чітких принципів організації фактів у базі знань. Без виділення і послідовного проведення таких принципів велика модель перетворюється в погано оглядний конгломерат незалежних фактів, що важко піддаються аналізу й обробці;

– надмірний рівень формалізації подання знань унаслідок збереження властивості цілісності, труднощі їхнього прочитання, не занадто гарна продуктивність обробки.

2.3.8 Продукційні моделі

Продукцією (продукційним правилом) називають вираз виду:

$$(i);\ Q;\ P;\ A_1, A_2, ..., A_n \rightarrow B_1, B_2, ..., B_k;\ N,$$

де *i – ім'я продукції*, у якості котрого може виступати деяка лексема, що відбиває суть даної продукції або її порядковий номер, Q – елемент, що характеризує сферу застосування продукції, $A_1, A_2, ..., A_n \rightarrow B_1, B_2, ..., B_k$ – *ядро продукції*, «→» – знак секвенції, A_i – і-та передумова (умова) правила, B_j – *j*-ий висновок (наслідок, дія) правила, P – умова застосовності ядра продукції, N – постумови продукції.

Продукційні правила читаються в такий спосіб: якщо передумови A_1 та A_2 та ... та A_n є вірними, то виконати дії B_1 та B_2 та ... та B_k.

Сфера застосування – визначає для яких випадків може бути застосована продукція, тобто задає множину елементів для якої продукція є застосовною. Поділ знань на окремі сфери дозволяє заощаджувати час на пошук потрібних знань.

Ядро продукції складається з двох частин:

– *антецедент* (передумова, умови правила) – являє собою комбінацію *умов правила* (припущень про наявність деяких властивостей, що приймають значення істина або хибність з визначеним ступенем вірогідності), з'єднаних логічними зв'язуваннями (ТА, АБО і т. д.), призначену для розпізнання ситуації, коли це правило повинне спрацювати: правило спрацьовує, якщо факти з робочої пам'яті задовольняють умовам передумови правила, після цього правило вважається відпрацьованим. Передумови звичайно бувають подані у формі вектора *об'єкт – атрибут – значення*. Впевненість у вірогідності передумови залежить від того, наскільки достовірною є оцінка умов;

– *консеквент* (висновок, наслідки правила) – містить опис дій, що повинні бути виконані над робочою пам'яттю у випадку виконання відповідних умов.

Іноді використовується й інша термінологія, відповідно до якої передумови називаються *лівою частиною* правила, а дії – *правою*.

Умова застосовності ядра продукції – визначає при якій умові продукція може бути виконана. Звичайна умова застосовності ядра являє собою логічний вираз. Коли вона приймає значення «істина», ядро продукції може бути активізовано. Якщо умова є хибною, то ядро продукції не може бути використано.

Постумови продукції – описують дії і процедури, які необхідно виконати після реалізації наслідків продукції й актуалізуються тільки в тому випадку, якщо ядро продукції реалізувалося.

Факти і правила не завжди бувають тільки істинні або тільки хибні. Іноді існує деякий ступінь непевності у вірогідності факту або точності правила, що явно виражається у вигляді коефіцієнта впевненості (див. п. 1.5.3).

Продукційна система (production system) складається з *продукційної пам'яті* (production memory) – бази знань у вигляді продукційних правил, *машини логічного виведення*, що послідовно визначає, які продукції можуть бути активовані в залежності від умов, у них що містяться, вибирає одне з застосовних у даній ситуації правил продукцій і виконує дії для обраного правила, а також *робочої пам'яті*, що містить дані (факти), опис мети і проміжні результати, що у сукупності визначають поточний стан проблеми.

Опис поточного стану проблеми, поданий фактами робочої пам'яті, є *зразком*, що зіставляється з умовною частиною продукцій з метою вибору відповідних дій при вирішенні задачі.

Керування системою продукцій (механізм виведення) здійснюється за допомогою машини логічного виведення, що виконує дві функції:

– перегляд існуючих фактів з робочої пам'яті і правил з бази знань і додавання (у міру можливості) у робочу пам'ять нових фактів;

– визначення порядку перегляду і застосування правил.

Керування пошуком у продукційній системі здійснюють:

– *за допомогою структури правил*, що у продукційній системі, включаючи розходження між умовою і дією, а також порядок перевірки умов, визначає метод дослідження простору рішень. Оскільки продукційна система перевіряє правила у визначеному порядку, програміст може керувати пошуком через структуру і порядок проходження правил у продукційному наборі;

– *на основі зразків* (pattern-directed search): *зразок* – опис задачі, що подає поточний стан світу, визначає конфліктну множину і, отже, конкретний шлях пошуку і рішення задачі.

Типи виконання систем продукцій виділяють:

– *прямий* (висхідний): пошук йде від лівих частин продукцій, тобто перевіряються умови й актуалізуються ті продукції, для яких умови виконуються;

– *зворотний* (спадний): пошук йде від початково заданих висновків, за якими визначаються необхідні для висновків значення умов, що, у свою чергу, ототожнюються з правими частинами ядер продукцій у системі.

Чиста продукційна модель не має ніякого механізму виходу з тупикових станів у процесі пошуку; вона просто продовжує працювати доти, поки не будуть вичерпані всі припустимі продукції. Багато практичних реалізацій продукційних систем містять механізми повернення в попередній стан робочої пам'яті.

Машина логічного виведення працює в режимі здійснення *циклів «розпізнавання – дія»* (цикл «вибрання – виконання», цикл «ситуація – відгук», цикл «ситуація – дія»): вона послідовно в циклі виконує деякі групи задач до виявлення визначених критеріїв, що викликають припинення виконання, при цьому в одному циклі може спрацювати тільки одне правило (див. рис. 2.3).

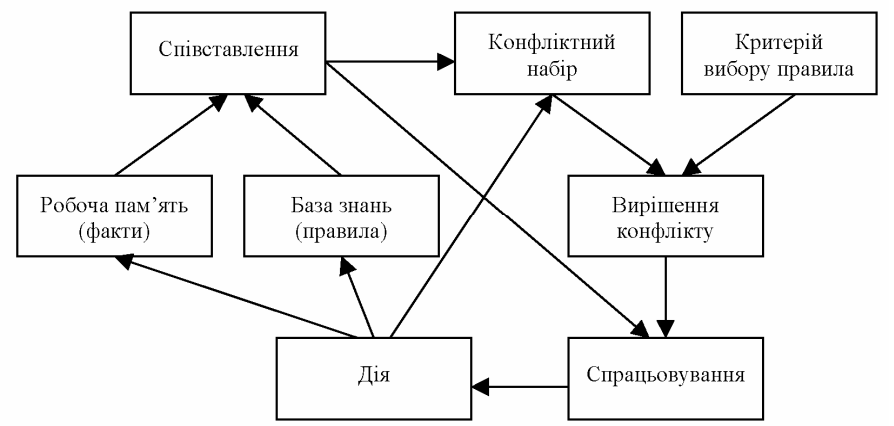


Рисунок 2.3 – Схема циклу роботи механізму виведення

Метод роботи машини логічного виведення містить такі кроки.

1. *Ініціалізація:* робоча пам'ять ініціалізується зразком – початковим описом задачі в у робочій пам'яті.

2. *Цикл «розпізнавання – дія»*.

2.1 *Ідентифікація* (зіставлення, узгодження, розпізнавання) – порівняння умов з лівих частин продукцій з фактами робочої пам'яті, що призводить до конкретизації й активізації правил.

Конкретизація правила – сукупність узагальненого формулювання правила і значень використовуваних у ньому змінних.

Активізоване (реалізоване) – правило, всі умови якого задоволено. Всі активізовані правила заносяться до робочого списку правил.

Конфлікти правил виникають у робочому списку правил, якщо одночасно активізується кілька правил.

Конфліктний набір правил (конфліктна множина, conflict set) – набір конкретизованих правил, які активізовані протягом одного циклу обчислень.

Припустимі продукції – продукції, що містяться в конфліктній множині.

2.2 *Вирішення конфліктів правил* (conflict resolution) – вибір зі сформованого списку заявок (конфліктної множини) єдиного правила, яке має бути виконане в поточній ситуації. Не маючи механізму вирішення конфліктів, продукційна система буде не в змозі ефективно справлятися з відсутністю детермінізму в наборі правил, обробкою виключень і переключенням уваги на визначений стиль розмірковувань. Іншими словами, подання буде страждати відсутністю евристичних здібностей, а керувати функціонуванням такої системи буде досить важко, навіть якщо знання подано цілком коректно.

2.3 *Дія* (запуск правил) – зміна вмісту робочої пам'яті за допомогою послідовного виконання дій, зазначених у правій частині активізованого правила, обраного з робочого списку правил. Після виконання дії, відповідне правило видаляється з робочого списку правил або виконується його релаксація.

Терміну запуск у нейрофізіології (науці про те, як працює нервова система) відповідає термін збудження. У результаті стимуляції окрема нервова клітина випускає електричний сигнал. Подальша стимуляція, наскільки сильною вона б ні була, не може змусити нейрон знову збудитися, поки не пройде якийсь короткий період часу. Цей феномен називається *релаксацією***.** Експертні системи, засновані на правилах, створюються з використанням релаксації з метою запобігання тривіальних циклів. Для забезпечення релаксації розроблено різні методи. Наприклад, кожному факту після його введення в робочу пам'ять присвоюється унікальний ідентифікатор, називаний *часовою оцінкою* (timetag). Слідом за тим, як деяке правило запускається під впливом деякого факту, машина логічного виведення не запускає його знову під впливом того ж самого факту, якщо з моменту застосування його часової оцінки пройшло занадто мало часу.

2.4 *Перевірка умов зупинення*. Якщо хоча б одна з умов зупинення задовольняється, тоді закінчити роботу і повернути рішення, якщо воно знайдено, у противному випадку – повторити кроки 2.1–2.4.

Умови зупинення виділяють такі:

– переривання роботи користувачем;

– вичерпання всіх правил із продукційної пам'яті;

– виконання деякої умови, якій задовольняє вміст робочої пам'яті (наприклад, поява в ній якогось факту);

– невідповідність вмісту робочої пам'яті ніяким умовам.

Протягом кожного циклу можуть бути активізовані і поміщені в робочий список правил багато правил. Крім того, у робочому списку правил залишаються результати активізації правил від попередніх циклів, якщо не відбувається *деактивізація* цих правил у зв'язку з тим, що їхні ліві частини більше не виконуються. У такий спосіб у ході виконання програми кількість активізованих правил у робочому списку правил змінюється. У залежності від програми раніше активізовані правила можуть завжди залишатися в робочому списку правил, але ніколи не вибиратися для запуску. Аналогічним чином деякі правила можуть ніколи не ставати активізованими. У подібних випадках варто повторно перевіряти призначення цих правил, оскільки або такі правила взагалі не потрібні, або їхні антецеденти неправильно спроектовано.

Використання різних стратегій перебору наявних знань, як правило, досить істотно впливає на характеристики ефективності програми. Ці стратегії визначають, яким чином програма відшукує рішення проблеми в деякому просторі альтернатив. Як правило, не буває так, щоб дані, які має програма роботи з базою знань, дозволяли точно визначити ту область у цьому просторі, де є сенс шукати відповідь. Оскільки стратегія вирішення конфліктів впливає на продуктивність системи в цілому, у більшості програмних систем передбачаються визначені опції для підстроювання цього механізму.

Стратегії вирішення конфліктів правил виділяють такі.

– *Розмаїтість* (рефракція – refraction): після активізації правила воно не може бути запущене знову, поки не зміняться елементи робочої пам'яті, що відповідають його умовам. Найпростіший варіант реалізації цього механізму – видаляти зі списку заявок застосоване раніше правило. Іноді використовується інший варіант: зі списку видаляється правило, активізоване в попередньому циклі, що запобігає зацикленню, але якщо бажано саме повторювати процедуру, то в розпорядження програміста надається функція відновлення, що дозволяє тимчасово придушити механізм, який діє за замовчуванням.

– *Новизна* (recency) – віддає перевагу правилам, умови яких відповідають фактам, доданим у робочу пам'ять останніми. Це дозволяє зосередити пошук на одній лінії суджень. Елементи в робочій пам'яті в таких системах позначаються спеціальним атрибутом часу породження. Це дозволяє системі ра-

нжирувати елементи в списку заявок відповідно до того, як давно введені в робочу пам'ять дані, що використовувалися при зіставленні, а потім пріоритет віддається правилам, що реагують на більш свіжі дані. Ідея полягає в тому, щоб випливати за поточною хвилею і менше уваги приділяти тим даним, що були давно сформовані. До них можна буде повернутися надалі, якщо поточний ланцюжок розсудів наштовхнеться на яку-небудь перешкоду.

– *Новизна правил*: найбільш нові правила, введені в систему в останню чергу, здобувають за замовчуванням найвищий пріоритет.

– *Старовина правил*: найбільш старі правила, введені в систему в першу чергу, здобувають за замовчуванням найвищий пріоритет.

– *Стратегія глибини* – утілення стратегії новизни даних стосовно правил, що мають однаковий клас опуклості. Правила, обрані в список заявок на підставі даних, що були включені в робочу пам'ять порівняно недавно, розташовуються в цьому списку раніше правил, при виборі яких використані більш старі дані. Таким чином, перевага віддається принципу пошуку в глибину в просторі станів проблеми, тобто правила, що є наслідком більш пізніх змін стану системи, мають визначений пріоритет.

– *Стратегія ширини* – протилежна стратегії глибини і призначається для реалізації пошуку в ширину в просторі станів проблеми. Правила, обрані в список заявок на підставі даних, що були включені в робочу пам'ять порівняно давно, розташовуються в цьому списку раніше правил, при виборі яких використані більш свіжі дані.

– *Стратегія складності* (специфічність – specifity, принцип найбільш довгої умови) – спирається на розуміння «здорового глузду», що частки правила, які відносяться до вузького класу ситуацій, є важливішими загальних правил, що відносяться до широкого класу ситуацій, оскільки частки правила враховують більше інформації про ситуацію, ніж загальні, і полягає у виборі з фронту готових продукцій тієї, у якої складність більше. *Складність правила* визначається кількістю операцій перевірки, які потрібно виконати при аналізі умов даного правила. Одне правило є більш специфічним (складним, конкретним) ніж інше, якщо воно містить більше умов, і відповідно складніше задовольняється, а виходить, відповідає меншій кількості правил у робочій пам'яті і має пріоритет перед більш загальними правилами. Цю стратегію можна ефективно використовувати при роботі з виключеннями з загальних правил, коли знання і самі продукції добре структуровані прив'язкою до типових ситуацій, на яких задане відношення типу «часткове – загальне». Труднощі використання даного принципу полягає в тому, що треба заздалегідь упорядкувати умови за входженням одна до одної за відношенням «часткове – загальне».

– *Стратегія простоти* – протилежна стратегії складності: більш верхнє положення (більший пріоритет) у списку заявок при реалізації цієї стратегії віддається тим правилам, складність яких нижче.

– *LEX-стратегія* – припускає спочатку видалення зі списку заявок усіх правил, що вже були раніше використані. Правила, що залишилися з рівним значенням опуклості, потім впорядковуються за новизною використовуваних даних. Якщо виявиться, що два правила використовують дані однакової новизни, то перевага віддається тому правилу, що втягує в аналіз передумов більше даних. Метод LEX практично ідентичний методу MEA за одним виключенням – у ньому відсутній крок 2, а на кроці 3 порівнюються всі елементи умов конкретизованих правил і зв'язаних з ними елементів робочої пам'яті (перші елементи умов конкретизованих правил, є, як правило, лексемами задач у робочій пам'яті).

– *MEA-стратегія* (Mean-Ends Analysis) багато в чому аналогічна LEX-стратегії, але при аналізі новизни беруться до уваги тільки перші умови в передумовах правил. Якщо виявиться, що в списку заявок виявилися два претенденти з рівними показниками, то для вибору між ними застосовується механізм LEX-стратегії. Метод MEA включає п'ять кроків.

Крок 1. Виключити з конфліктної множини правил ті правила, що вже були застосовані в попередньому циклі. Якщо після цього множина стала порожньою, припинити процес.

Крок 2. Порівняти новизну елементів у робочій пам'яті, що відповідають першим елементам умов у конкретизованих правилах, які залишилися в конфліктній множині. Пріоритет віддається тим правилам, що звертаються до найновіших елементів робочої пам'яті, – ці правила домінують над іншими. Якщо існує єдине таке правило, то воно вибирається для застосування, після чого процес повинний бути припинений. У противному випадку, якщо є кілька домінуючих правил з рівним пріоритетом, вони зберігаються в конфліктній множині, а інші з неї видаляються. Далі виконується перехід до кроку 3.

Крок 3. Упорядкувати конкретизовані правила за новизною інших елементів умов у правилах. Якщо домінує одне правило, воно застосовується і потім процес припиняється. Якщо ж домінують два чи кілька правил, то вони залишаються в конфліктуючій множині, а інші видаляються з неї. Потім виконується перехід до кроку 4.

Крок 4. Якщо за критерієм новизни елементів умов відібрано кілька правил з рівними показниками, то порівнюється інший показник правил – показник специфіки. Перевага віддається тому правилу, застосування якого вимагає перевірки найбільшої кількості умов у робочій пам'яті. Якщо претендентів виявиться кілька, інші видаляються з конфліктної множини і виконується перехід до кроку 5.

Крок 5. Застосовується правило, обране довільним чином з тих, що залишилися у конфліктній множині, і процес припиняється.

Таким чином, стратегія MEA поєднує в одному методі аналіз таких показників, як повторюваність, новизна і специфіка.

– *Принцип «стопки книг»* (частотна перевага) – заснована на ідеї, що найбільш часто використовувана продукція є найбільш корисною. Готові продукції як би утворять «стопку», у якій порядок визначається накопиченою частотою використання продукцій у минулому. На самому верху «стопки» знаходиться продукція, що використовувалася частіше усіх. При актуалізації деякого фронту готових продукцій для виконання вибирається та продукція (чи ті продукції при наявності рівнобіжних технічних пристроїв), у якої частота використання є максимальною. Подібний принцип є особливо гарним, коли частота виконання підраховується з урахуванням деякої ситуації, у якій раніше виповнювалася продукція, і це виконання мало позитивну оцінку. При такому зворотному зв'язку метод стопки книг може перетворитися в процедуру, що навчається та адаптується до тих задач, що виникають у зовнішньому середовищі. Принцип стопки книг доцільно застосовувати, якщо продукції є відносно незалежними одна від одної.

– *Принцип метапродукцій* – заснований на ідеї введення в систему продукцій спеціальних метапродукцій (метаправил), завданням яких є організація керування в системі продукцій при можливості неоднозначного вибору з фронту готових продукцій.

– *Керування за іменами* – засновано на задаванні для імен продукцій, що входять у деяку систему, певної формальної граматики або іншої процедури, що забезпечує звуження фронту готових продукцій і вибір з нього чергової продукції для виконання.

– *Упорядкування на основі цілей:* правила упорядковуються, принаймні частково, у залежності від того, наскільки швидко вони приводять до досягнення мети. Виконується пошук випадків, ціль яких відповідає поточній ситуації.

– *Принцип пріоритетного вибору* – пов'язаний із уведенням статичних або динамічних пріоритетів на продукції. Статичні пріоритети можуть формуватися апріорі на підставі відомостей про важливість продукційних правил у даній проблемній області. Ці відомості, як правило, являють собою інформацію, що витягається з експерта. Динамічні пріоритети виробляються в процесі функціонування системи продукцій і можуть відбивати, наприклад, такий параметр, як час перебування продукції у фронті готових продукцій. До даного типу керування відноситься задавання послідовності пріоритетів за допомогою спеціальної каузальної семантичної мережі. У цьому випадку задається певний каузальний сценарій, рух за яким визначається виникаючими ситуаціями, і в кожній вершині якого задана функція вибору чергової продукції з фронту готових продукцій.

– *Модель дошки оголошень* (модель класної дошки) – стратегія вирішення складних системних задач із залученням різнорідних джерел знань, взаємодіючих через загальне інформаційне поле – модель керування, що успішно застосовується для вирішення задач, які вимагають координації різних процесів або джерел знання, яка ґрунтується на ідеї спускових функцій (див. рис. 2.4).

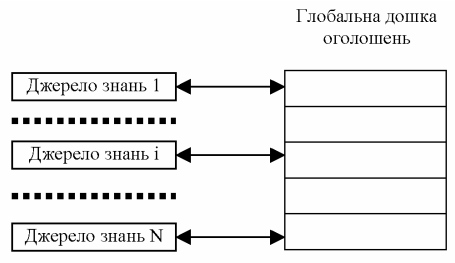


Рисунок 2.4 – Архітектура дошки оголошень

Метою розробки моделі дошки оголошень було створення такої системи, що компенсує обмеженість монотонних виведень і може застосовуватися для вирішення складних проблем за рахунок використання механізму поділу джерел знань та їхніх погоджених дій у загальній робочій пам'яті зі збереженням при цьому такої переваги продукційної системи, як модульність знань.

Дошка оголошень (класна дошка – blackboard) – центральна глобальна база даних, призначена для зв'язку незалежних асинхронних джерел знань.

Як правило, на дошці оголошень виділяють спеціальні поля для формування умов застосовності ядер продукцій, різні для різних сфер застосування продукцій, спеціальні поля для запису результатів спрацьовування продукцій і для запису постумов, якщо вони адресовані іншими продукціям.

Кожне *джерело знань* (knowledge source) є групою однотипних продукційних правил та одержує свої дані від дошки оголошень, обробляє їх і повертає результати дошці оголошень для їхнього подальшого використання іншими джерелами знань. Це ізольовані процеси, що діють відповідно до своїх власних специфікацій (опису). Тому при рівнобіжній обробці інформації або мультипроцесорній системі ці системи беруть участь у вирішенні єдиної задачі незалежно одна від одної. Це асинхронна система, оскільки кожне джерело знань, починає свою роботу, коли знаходить відповідні вхідні дані на дошці оголошень. Закінчивши обробку даних, воно повертає свої результати й очікує нових вхідних даних.

Якщо джерело знань на одному рівні не може обробити (осмислити) передані йому дані, то воно робить запит до відповідного джерела про нові дані. Потім, одержавши нові дані, робить ще одну спробу їхньої обробки або приймає іншу гіпотезу відносно спочатку отриманих даних. Усі процеси є асинхронними і керованими на основі даних. Вони запускаються з появою вхідних даних і продовжують роботу, поки не закінчать завдання, потім повертають свої результати і чекають наступного завдання.

Одне з джерел знань, називане *планувальником*, обробляє повідомлення про результати обробки інформації, передані між іншими джерелами знань. Планувальник ранжирує результати діяльності кожного джерела знань

і за допомогою пріоритетної черги забезпечує деякий напрямок вирішення задачі. Якщо жодне з джерел знань не активним, планувальник вирішує, що задача є завершеною, і припиняє свою роботу.

З принципом дошки оголошень може комбінуватися принцип керування за допомогою метапродукцій, оскільки він вимагає перевірки деяких умов, що фіксуються в робочому полі пам'яті, а також інші принципи керування.

Перевагами продукційних моделей є:

– модульність організації знань у вигляді продукційних правил;

– незалежність правил, що виражають самостійні фрагменти знань про предметну область. Важливий аспект у моделюванні продукційних систем – це відсутність синтаксичної взаємодії між продукційними правилами. Правила можуть тільки впливати на активізацію інших правил, змінюючи зразок у робочій пам'яті. Правила не можуть «викликати» інше правило безпосередньо. При цьому вони не можуть установлювати значення змінних в інших продукційних правилах. Область дії змінних цих правил обмежена окремим правилом. Ця синтаксична незалежність сприяє інкрементальній розробці експертних систем шляхом послідовного додавання, видалення або зміни знань (правил) системи;

– простота створення і розуміння окремих правил;

– легкість і природність модифікації знань;

– відділення керуючих знань, забезпечуваних циклом «розпізнання – дія» продукційної системи, від предметних знань, зосереджених безпосередньо в правилах, що дозволяє застосовувати різні керуючі стратегії і легко змінювати базу знань, а також не вимагає зміни коду програми керування і, навпаки, дозволяє змінювати код керуючої частини програми, не торкаючи набір правил виведення;

– природний паралелізм у системі продукцій, асинхронність їхньої реалізації, що роблять продукційні системи зручною моделлю обчислень для ЕОМ нової архітектури, у якій ідея паралельності й асинхронності є центральною;

– простота механізму логічного виведення;

– гнучкість у застосуванні знань;

– природна відповідність пошуку в просторі станів: компоненти продукційної системи природно відображаються в логічну структуру пошуку в просторі станів. Послідовні стани робочої пам'яті складають вершини графа простору станів. Правила виведення – набір можливих переходів між станами. Вирішення конфліктів забезпечує вибір переходу (гілки) у просторі станів. Ці правила спрощують виконання, відлагодження і документування методів пошуку.

Недоліками продукційних моделей є: неясність взаємних відношень правил; складність оцінки цілісного образа знань; украй низька ефективність обробки; відмінність від людської структури знань; складність перевірки несуперечності системи продукцій при великому числі продукцій, що змушує при додаванні нових продукцій витрачати багато часу на перевірку несупере-

чності нової системи; недетермінованість (неоднозначність вибору виконуваної продукції з фронту активізованих продукцій), що обумовлює принципові труднощі при перевірці коректності роботи системи (вважається, що якщо в інтелектуальній системі число продукцій досягає тисячі, то мало шансів, що система продукцій у всіх випадках буде правильно функціонувати).

2.3.9 Дерева рішень

Дерева рішень (дерева вирішальних правил) – один з методів автоматичного аналізу даних, що задає спосіб подання правил виду «Якщо – то» в ієрархічній послідовній структурі, де кожному об'єкту відповідає єдиний вузол, що дає рішення.

Основними поняттями теорії дерев рішень, є:

- *об'єкт* – приклад, шаблон, спостереження;
- *атрибут* – ознака, незалежна змінна, властивість;
- *мітка класу* – залежна (цільова) змінна, ознака, що визначає клас об'єкта;
- *вузол* – внутрішній вузол дерева, вузол перевірки;
- *лист* – кінцевий вузол дерева, вузол рішення;
- *перевірка* (test) – умова у вузлі.

Перші ідеї створення дерев рішень запропоновано у роботах П. Ховленда (P. Hoveland) та Е. Ханта (E. Hunt) наприкінці 50-х років XX століття.

Область застосування дерев рішень у даний час є дуже широкою, але всі задачі, розв'язувані цим апаратом, можуть бути об'єднані в такі три класи.

- Опис даних: дерева рішень дозволяють зберігати дані про об'єкти в компактній формі.
- Класифікація: дерева рішень відмінно справляються з задачами класифікації, тобто з віднесенням об'єктів до одного з заздалегідь відомих класів. Цільова змінна повинна мати дискретні значення.
- Регресія: якщо цільова змінна має неперервні значення, дерева рішень дозволяють установити залежність цільової змінної від незалежних (вхідних) змінних. Наприклад, до цього класу відносяться задачі чисельного прогнозування (передбачення значень цільової змінної).

Побудова дерева рішень

Нехай нам задана деяка навчаюча множина T, що містить об'єкти (приклади), кожний з яких характеризується m атрибутами, причому один з них указує на приналежність об'єкта до визначеного класу. Тоді дерево рішень може бути побудоване в результаті виконання таких дій.

Нехай через C_1, C_2, ..., C_k позначено класи (значення міток класів), тоді існують три ситуації:

- множина T містить один або більше прикладів, що відносяться до одного класу C_k. Тоді дерево рішень для T – це лист, що визначає клас C_k;

– множина T не містить жодного приклада, тобто порожня множина. Тоді це знову лист, і клас, асоційований з листом, вибирається з іншої множини відмінного від T, скажімо, з множини, асоційованої з батьком;

– множина T містить приклади, що відносяться до різних класів. У цьому випадку варто розбити множину T на деякі підмножини. Для цього вибирається одна з ознак, що має два і більше відмінних одне від одного значень O_1, O_2, ..., O_n. T розбивається на підмножини T_1, T_2, ..., T_n, де кожна підмножина T_i містить усі приклади, що мають значення O_i для обраної ознаки. Ця процедура буде рекурсивно продовжуватися доти, поки кінцева множина не буде складатися з прикладів, що відносяться до одного й того ж самого класу.

Вищеописана процедура лежить в основі багатьох сучасних методів побудови дерев рішень, цей метод відомий ще під назвою *поділ і захоплення* (divide and conquer). Очевидно, що при використанні цього методу, побудова дерева рішень буде відбувається зверху вниз.

Оскільки всі об'єкти були заздалегідь віднесені до відомих нам класів, такий процес побудови дерева рішень називається *навчанням із учителем* (supervised learning). Процес навчання також називають *індуктивним навчанням* або *індукцією дерев* (tree induction).

На сьогоднішній день існує значне число методів, що реалізують дерева рішень: *CART*, *C4.5*, *NewId*, *ITrule*, *CHAID*, *CN2* і т. д. Але найбільше поширення і популярність одержали:

– *C4.5* – метод побудови дерева рішень, де кількість нащадків у вузла є необмеженою. Не вміє працювати з неперервними цільовими змінними, тому вирішує тільки задачі класифікації;

– *CART* (Classification and Regression Tree) – метод побудови *бінарного дерева рішень* – дихотомічної класифікаційної моделі. Кожен вузол дерева при розбитті має тільки двох нащадків. Як видно з назви методу, він вирішує задачі класифікації і регресії.

Більшість з відомих методів є *«жадібними методами»*. Якщо один раз був обраний атрибут, і за ним було зроблено розбиття на підмножини, то метод не може повернутися назад і вибрати інший атрибут, що дав би краще розбиття. І тому на етапі побудови не можна сказати чи дасть обраний атрибут, в остаточному підсумку, оптимальне розбиття.

При побудові дерев рішень особлива увага приділяється вибору критерію атрибута, за яким піде розбиття, зупинки навчання і відсікання гілок.

Правило вибору ознаки для розбиття. Для побудови дерева на кожному внутрішньому вузлі необхідно знайти таку умову (перевірку), яка б розбивала множину, асоційовану з цим вузлом, на підмножини. У якості такої перевірки повинний бути обраний один з атрибутів. Загальне *правило для вибору атрибута* можна сформулювати в такий спосіб: обраний атрибут повинний розбити множину так, щоб одержувані в підсумку підмножини склада-

лися з об'єктів, що належать до одного класу, або були максимально наближені до цього, тобто кількість об'єктів з інших класів («домішок») у кожній з цих множин була якнайменшою. Відомі різні критерії.

– *Теоретико-інформаційний критерій* для вибору найбільш придатного атрибута запропонований Р. Куінленом (R. Quinlean)і використовується в методі C4.5 – удосконаленій версії методу ID3 (Iterative Dichotomizer):

$$\text{Gain}(X)=\text{Info}(T)–\text{Info}_X(T),$$

$$\text{Info}_X(T)=\sum_{i=1}^{n}\frac{|T_i|}{|T|}\text{Info}(T_i),$$

де Info(T) – ентропія множини T, а $|T|$ – *потужність множини* T – кількість прикладів у множині T.

Множини T_1, T_2, ..., T_n, отримані при розбитті вихідної множини T за перевіркою X. Вибирається атрибут, що дає максимальне значення за критерієм Gain(X).

– *Статистичний критерій – індекс Gini* (на честь італійського економіста C. Gini) оцінює «відстань» між розподілами класів і використовується в методі CART, запропонованому Л. Брейманом (L. Breiman):

$$\text{Gini}(c)=1-\sum_{j=1}^{k}p_j^2,$$

де c – поточний вузол, а p_j – імовірність класу j у вузлі c.

Правило зупинки визначає, чи розбивати далі вузол або позначити його як лист. На додаток до основного методу побудови дерев рішень були запропоновані такі правила.

– *Рання зупинка* (prepruning) – використання статистичних методів для оцінки доцільності подальшого розбиття. Рання зупинка процесу побудови приваблива в плані економії часу навчання, але цей підхід будує менш точні класифікаційні моделі і тому рання зупинка вкрай небажана. Рекомендується замість зупинки використовувати відсікання.

– *Обмеження глибини дерева*: подальшу побудову зупиняють, якщо розбиття веде до дерева з глибиною, яка перевищує задане значення.

– Розбиття повинне бути нетривіальним, тобто вузли, що вийшли в результаті, повинні містити не менш заданої кількості прикладів.

Цей список евристичних правил можна продовжити, але на сьогодні не існує такого правила, яке б мало велику практичну цінність. До цього питання варто підходити обережно, оскільки більшість цих правил є застосовними лише в якихось окремих випадках.

Правило відсікання визначає, яким чином гілки дерева повинні відтинатися.

Оскільки бажано мати дерево, що складається з малої кількості вузлів, яким би відповідала велика кількість об'єктів з навчаючої вибірки, виникає задача побудови всіх можливих дерев, що відповідають навчаючій множині, і вибору з них дерева з найменшою глибиною. На жаль, ця задача є NP-повною, що було показано Л. Хайфілем (L. Hyafill) та Р. Ривестом (R. Rivest), і, як відомо, цей клас задач не має ефективних методів рішення. Для вирішення даної проблеми часто застосовується *відсікання гілок* (pruning).

Під *точністю розпізнавання* для дерева рішень розуміється відношення правильно класифікованих об'єктів при навчанні до загальної кількості об'єктів з навчаючої множини, а під *помилкою* – кількість неправильно класифікованих. Припустимо, що нам відомий спосіб оцінки помилки для дерева, гілок і листів. Тоді, можливо використовувати таке просте правило: 1) побудувати дерево; 2) відітнути або замінити піддеревом ті гілки, що не приведуть до зростання помилки.

На відміну від процесу побудови, відтинання гілок відбувається знизу нагору, рухаючи з листів дерева, відзначаючи вузли як листи, або заміняючи їх піддеревом. Хоча відтинання не є панацеєю, але в більшості практичних задач дає гарні результати, що дозволяє говорити про правомірність використання подібної методики.

Іноді навіть усічені дерева можуть бути усе ще складними для сприйняття. У такому випадку можна удатися до *методики витягу правил з дерева з подальшим створенням наборів правил, що описують класи*. Для витягу правил необхідно досліджувати всі шляхи від кореня до кожного листа дерева. Кожен такий шлях дасть правило, де умовами будуть перевірки з вузлів, що зустрілися на шляху.

Метод C4.5 висуває такі вимоги до структури і значень даних. Дані, необхідні для роботи методу, повинні бути подані у вигляді плоскої таблиці. Вся інформація про об'єкти із предметної області повинна описуватися у виді кінцевого набору ознак. Кожний атрибут повинен мати дискретне або числове значення. Самі атрибути не повинні мінятися від приклада до приклада і кількість атрибутів повинна бути фіксованою для всіх прикладів. Кожен приклад повинний бути асоційований з конкретним класом, тобто один з атрибутів повинний бути обраний як мітка класу. Класи повинні бути дискретними, тобто мати кінцеве число значень. Кожен приклад повинний однозначно відноситися до конкретного класу. Випадки, коли приклади належать до класу з імовірнісними оцінками, виключаються. Кількість класів повинна бути значно меншою кількості прикладів.

Метод побудови дерева.

Нехай нам задана множина прикладів *T*, де кожний елемент цієї множини описується m атрибутами та мітка класу приймає значення C_1, C_2, ..., C_k.

Наша задача буде полягати в побудові ієрархічної класифікаційної моделі у виді дерева з множини прикладів *T*.

Процес побудови дерева буде відбуватися зверху вниз. Спочатку створюється корінь дерева, потім нащадки кореня і т. д. На першому кроці ми маємо порожнє дерево (є тільки корінь) і вихідну множину T (асоційовану з коренем). Потрібно розбити вихідну множину на підмножини. Це можна зробити, вибравши один з атрибутів як перевірку. Тоді в результаті розбиття виходять n (по числу значень атрибута) підмножин і, відповідно, створюються n нащадків кореня, кожному з яких співставлено у відповідність свою підмножину, отриману при розбитті множини T. Потім ця процедура рекурсивно застосовується до всіх підмножин (нащадків кореня) і т. д.

Розглянемо докладніше критерій вибору атрибута, за яким повинне піти розгалуження. Очевидно, що в нашому розпорядженні m (за числом атрибутів) можливих варіантів, з яких ми повинні вибрати самий придатний. Деякі методи виключають повторне використання атрибута при побудові дерева, але в нашому випадку ми таких обмежень накладати не будемо. Будь-який з атрибутів можна використовувати необмежену кількість разів при побудові дерева.

Нехай ми маємо перевірку X (у якості перевірки може бути обраний будь-який атрибут), що приймає n значень A_1, A_2, ..., A_n. Тоді розбиття T за перевіркою X дасть нам підмножини T_1, T_2, ..., T_n, при X рівному відповідно A_1, A_2, ..., A_n. Єдина доступна інформація – те, яким чином класи розподілені в множині T і її підмножинах, одержуваних при розбитті за X. Це і використовують при визначенні критерію.

Нехай $\text{freq}(C_j,S)$ – кількість прикладів з деякої множини S, що відносяться до того ж самого класу C_j. Тоді імовірність того, що випадково обраний приклад із множини S буде належати до класу C_j:

$$P = \text{freq}(C_j,S) / |S|.$$

Відповідно до теорії інформації, кількість інформації, що міститься в повідомленні, залежить від її імовірності $\log_2(P^{-1})$. Оскільки ми використовуємо логарифм із двійковою основою, то цей вираз дає кількісну оцінку в бітах.

Вираз

$$\text{Info}(T) = -\sum_{j=1}^{k} \frac{\text{freq}(C_j,T)}{|T|} \log_2\left(\frac{\text{freq}(C_j,T)}{|T|}\right)$$

дає оцінку середньої кількості інформації, необхідної для визначення класу приклада з множини T. У термінології теорії інформації цей вираз називається ентропією множини T.

Ту ж оцінку, але тільки вже після розбиття множини T за X, дає вираз:

$$\text{Info}_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \text{Info}(T_i) .$$

Тоді критерієм для вибору атрибута буде така формула:

$$\text{Gain}(X) = \text{Info}(T) - \text{Info}_X(T).$$

Даний критерій розраховується для всіх атрибутів. Вибирається атрибут, що максимізує даний вираз. Цей атрибут буде перевіркою в поточному вузлі дерева, а потім за цим атрибутом здійснюється подальша побудова дерева. Тобто у вузлі буде перевірятися значення за цим атрибутом і подальший рух по дереву буде здійснюватися в залежності від отриманої відповіді.

Таке ж міркування можна застосувати до отриманих підмножин T_1, T_2, ..., T_n і продовжити рекурсивно процес побудови дерева, доти, поки у вузлі не виявляться приклади з одного класу.

Якщо в процесі роботи методу отримано вузол, асоційований з порожньою множиною (тобто жоден приклад не потрапив в даний вузол), то він позначається як лист, і як рішення листа обирається клас, що найбільш часто зустрічається у безпосереднього предка даного листа.

З властивостей ентропії нам відомо, що максимально можливе значення ентропії досягається в тому випадку, коли всі його повідомлення є рівноімовірними. У нашому випадку, ентропія $\text{Info}_X(T)$ досягає свого максимуму, коли частота появи класів у прикладах множини T є рівноімовірною. Нам же необхідно вибрати такий атрибут, щоб при розбитті за ним один із класів мав найбільшу імовірність появи. Це можливо в тому випадку, коли ентропія $\text{Info}_X(T)$ буде мати мінімальне значення і, відповідно, критерій $\text{Gain}(X)$ досягне свого максимуму.

У випадку числових атрибутів варто обрати деякий поріг, з яким повинні порівнюватися всі значення атрибута. Нехай числовий атрибут має кінцеве число значень. Позначимо їх $\{v_1, v_2, ..., v_n\}$. Попередньо відсортуємо всі значення. Тоді будь-яке значення, що лежить між v_i і v_{i+1}, поділяє всі приклади на дві множини: ті, котрі лежать ліворуч від цього значення $\{v_1, v_2, ..., v_i\}$, і ті, що праворуч $\{v_{i+1}, v_{i+2}, ..., v_n\}$. Як поріг можна вибрати середнє між значеннями v_i і v_{i+1}: $\text{TH}_i = 0{,}5(v_i + v_{i+1})$.

Таким чином, ми істотно спростили задачу знаходження порога, і привели до розгляду усього $n-1$ потенційних граничних значень TH_1, TH_2, ..., TH_{n-1}. Послідовно для всіх потенційних граничних значень визначаються значення критерію $\text{Gain}(\text{TH}_i)$ і серед них вибирається те, що дає максимальне значення критерію. Далі це значення порівнюється зі значеннями критерію, підрахованими для інших атрибутів. Якщо з'ясується, що серед всіх атрибутів даний числовий атрибут має максимальне значення критерію, то як перевірка вибирається саме він.

Слід зазначити, що всі числові тести є бінарними, тобто поділяють вузол дерева на дві гілки.

Класифікація нових прикладів.

Нехай ми маємо дерево рішень і хочемо використовувати його для розпізнавання нового об'єкта. Обхід дерева рішень починається з кореня дерева. На кожному внутрішньому вузлі перевіряється значення об'єкта Y за

атрибутом, що відповідає перевірці в даному вузлі, і, у залежності від отриманої відповіді, знаходиться відповідне розгалуження, і по цій дузі рухаємося до вузла, що знаходиться на рівень нижче і т. д. Обхід дерева закінчується як тільки зустрінеться вузол рішення, що і дає назву класу об'єкта *Y*.

За допомогою розглянутого методу можна одержати дерево рішень, незначним недоліком якого буде гіллястість, якщо приклади подані дуже великою кількістю атрибутів. Цей недолік можна усунути, застосувавши відсікання (pruning) гілок.

Поліпшений критерій розбиття.

Критерій Gain(X) має один недолік – він надає переваги атрибутам, які мають багато значень. У випадку, якщо серед атрибутів зустрінеться такий, котрий для кожного приклада має унікальне значення, то при розбитті множини прикладів за цим атрибутом вийдуть підмножини, що містять тільки по одному прикладу. Оскільки всі ці множини «однозразкові», то і приклад відноситься, відповідно, до одного єдиного класу, тоді $\mathrm{Info}_X(T) = 0$.

Значить даний критерій приймає своє максимальне значення, і безсумнівно, що саме цей атрибут буде обраний методом. Однак, якщо розглянути проблему під іншим кутом – з погляду здібностей передбачення побудованої моделі, то стає очевидним уся марність такої моделі.

Проблема вирішується введенням деякої нормалізації. Нехай суть інформації повідомлення, що відноситься до прикладу, указує не на клас, до якого приклад належить, а на вихід. Тоді, за аналогією з визначенням Info(T), маємо

$$\mathrm{splitInfo}(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \log_2\left(\frac{|T_i|}{|T|}\right).$$

Цей вираз оцінює потенційну інформацію, одержувану при розбитті множини T на n підмножин.

Розглянемо вираз: gainratio(X) = Gain(X)/splitinfo(X).

Нехай цей вираз є критерієм вибору атрибута.

Очевидно, що атрибут, який має унікальне значення для кожного приклада, не буде високо оцінений критерієм даного виразу. Нехай є k класів, тоді чисельник даного виразу максимально буде дорівнювати $\log_2(k)$ і нехай n – кількість прикладів у навчаючій вибірці й одночасно кількість значень атрибута, тоді знаменник максимально дорівнює $\log_2(n)$. Якщо припустити, що кількість прикладів свідомо більше кількості класів, то знаменник росте швидше, ніж чисельник, і, відповідно, вираз буде мати невелике значення. Таким чином, ми можемо замінити критерій $\mathrm{Info}_X(T)$ на новий критерій gainratio(X), і знову ж вибрати той атрибут, що має максимальне значення за критерієм. Критерій $\mathrm{Info}_X(T)$ використовувався в методі ID3, критерій gainratio(X) введено у модифікованому методі C4.5.

Незважаючи на поліпшення критерію вибору атрибута для розбиття, метод може створювати вузли і листи, що містять незначну кількість прикладів. Щоб уникнути цього, варто скористатися ще одним евристичним правилом: при розбитті множини T, принаймні дві підмножини повинні мати не менше заданої мінімальної кількості прикладів k (k>1); звичайно вона дорівнює 2. У випадку невиконання цього правила, подальше розбиття цієї множини припиняється, і відповідний вузол відзначається як лист. Імовірно, що при такім обмеженні може виникнути ситуація коли приклади, асоційовані з вузлом, відносяться до різних класів. Як рішення для листа обирається клас, що найбільш часто зустрічається у вузлі, якщо ж прикладів рівна кількість з усіх класів, то рішення дає клас, що найбільш часто зустрічається у безпосереднього предка даного листа.

При виборі порога для числових атрибутів, можна ввести доповнення: якщо мінімальне число прикладів у вузлі – k, тоді є сенс розглядати тільки значення TH_1, TH_2, ..., TH_{n-1}, оскільки при розбитті за першим і останнім k–1 порогам у вузол попадає менше k прикладів.

Пропущені дані.

Метод побудови дерев рішень припускає, що для атрибута, обраного як перевірка, існують усі значення, хоча явно це ніде не стверджувалося. Тобто для будь-якого приклада з навчаючої вибірки існує значення за цим атрибутом. На практиці дані далекі від ідеальних, і часто зустрічаються пропущені, суперечні й аномальні дані.

Одне з можливих рішень – не враховувати приклади з пропущеними значеннями. Варто підкреслити, що вкрай небажано відкидати весь приклад тільки тому, що за одним з атрибутів пропущене значення, оскільки ми ризикуємо втратити багато корисної інформації.

Тоді нам необхідно використовувати процедуру роботи з пропущеними даними.

Нехай T – множина навчальних прикладів та X – перевірка за деяким атрибутом A. Позначимо через U кількість невизначених значень атрибута A. Будемо враховувати тільки ті приклади, у яких існують значення за атрибутом A. Тоді:

$$\mathrm{Info}(T) = -\sum_{j=1}^{k} \frac{\mathrm{freq}(C_j, T)}{|T| - U} \log_2\left(\frac{\mathrm{freq}(C_j, T)}{|T| - U}\right),$$

$$\mathrm{Info}_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T| - U} \mathrm{Info}(T_i).$$

У цьому випадку при підрахунку $\mathrm{freq}(C_j, T)$ враховуються тільки приклади з існуючими значеннями атрибута A.

Тоді критерій $\mathrm{Gain}(x)$ можна переписати:

$$\mathrm{Gain}(X) = (|T| - U)(\mathrm{Info}(T) - \mathrm{Info}_X(T)) / |T|.$$

Подібним чином змінюється і критерій gainratio. Якщо перевірка має n вихідних значень, то критерій gainratio розраховується як у випадку, коли вихідна множина розділена на n+1 підмножин.

Нехай тепер перевірка X з вихідними значеннями O_1, O_2, ..., O_n обрана на основі модифікованого критерію Gain.

Треба вирішити, що робити з пропущеними даними. Якщо приклад із множини T з відомим виходом O_i асоційований з підмножиною T_i, імовірність того, що приклад із множини T_i, дорівнює 1. Нехай тоді кожний приклад з підмножини T_i має вагу, яка вказує імовірність того, що приклад належить T_i. Якщо приклад має значення за атрибутом A, тоді вага дорівнює 1, у протилежному випадку приклад асоціюється з усіма множинами T_1, T_2, ..., T_n, з відповідними вагами $|T_i| / (|T| - U)$, сума яких дорівнює одиниці.

Коротко, цей підхід можна сформулювати в такий спосіб: передбачається, що пропущені значення за атрибутом імовірнісно розподілені пропорційно частоті появи існуючих значень.

Класифікація нових прикладів із пропущеними даними.

Така ж методика застосовується, коли дерево використовується для класифікації нових прикладів. Якщо на якомусь вузлі дерева при виконанні перевірки з'ясовується, що значення відповідного атрибута приклада, який класифікується, є пропущеним, то метод досліджує всі можливі шляхи вниз по дереву і визначає з якою імовірністю приклад відноситься до різних класів. У цьому випадку, «класифікація» – це скоріше розподіл класів. Як тільки розподіл класів установлено, то клас, що має найбільшу імовірність появи, вибирається як відповідь дерева рішень.

Слід зазначити, що крім підходу, використаного в даному методі, застосовуються й інші методики. В остаточному підсумку, успіх від використання того чи іншого методу роботи з пропущеними даними, прямо залежить як від предметної області, так і від самих даних.

CART (Classification And Regression Tree – дерево класифікації і регресії) – метод бінарного дерева рішень, призначений для вирішення задач класифікації і регресії. Вперше опубліковано у 1984 р. Л. Брейманом та ін.

Основними відмінностями методу CART від методів сімейства ID3 є: бінарне подання дерева рішень; функція оцінки якості розбиття; механізм відсікання дерева; метод обробки пропущених значень; побудова дерев регресії.

Бінарне подання дерева рішень – в методі CART кожен вузол дерева рішень має двох нащадків. На кожному кроці побудови дерева правило, формоване у вузлі, поділяє задану множину прикладів (навчаючу вибірку) на дві частини: частину, у якій виконується правило (правий нащадок – right) і частину, у якій правило не виконується (лівий нащадок – left). Для вибору оптимального правила використовується функція оцінювання якості розбиття.

Кожен вузол (структура або клас) повинний мати посилання на двох нащадків Left і Right – аналогічні структури. Також вузол повинний містити ідентифікатор правила, певним чином описувати праву частину правила, містити інформацію про кількість або відношення прикладів кожного класу навчаючої вибірки, що пройшла через вузол, і мати ознаку термінального вузла – листа.

Функція оцінювання якості розбиття – критерій, що дозволяє оцінити якість поточного розбиття.

Навчання дерева рішень відноситься до класу навчання з учителем, тобто навчаюча і тестова вибірки містять *класифікований* набір прикладів. Оцінна функція, використовувана методом CART, базується на інтуїтивній ідеї зменшення нечистоти (невизначеності) у вузлі.

В методі CART ідея нечистоти формалізована в індексі Gini(T), де T – набір даних, що містить дані n класів.

Якщо набір T розбивається на дві частини T_1 і T_2 з числом прикладів у кожній N_1 і N_2 відповідно, тоді показник якості розбиття буде дорівнювати:

$$\text{Gini}_{\text{split}}(T) = \frac{N_1}{N}\text{Gini}(T_1) + \frac{N_2}{N}\text{Gini}(T_2).$$

Найкращим вважається те розбиття, для якого $\text{Gini}_{\text{split}}(T)$ є мінімальним.

Позначимо N – число прикладів у вузлі – предку, L, R – число прикладів відповідно в лівому і правому нащадках, l_i та r_i – число екземплярів i-го класу в лівому та правому нащадках, відповідно. Тоді якість розбиття оцінюється за формулою:

$$\text{Gini}_{\text{split}} = \frac{L}{N}\left(1 - \sum_{i=1}^{n}\left(\frac{l_i}{L}\right)^2\right) + \frac{R}{N}\left(1 - \sum_{i=1}^{n}\left(\frac{r_i}{R}\right)^2\right) \to \min.$$

Щоб зменшити обсяг обчислень формулу можна перетворити:

$$G_{\text{split}} = \frac{1}{L}\sum_{i=1}^{n} l_i^2 + \frac{1}{R}\sum_{i=1}^{n} r_i^2 \to \max.$$

У результаті, кращим буде те розбиття, для якого величина G_{split} є максимальною.

Правила розбиття – визначають принцип прийняття рішень у вузлах дерева рішень.

Вектор предикторних змінних, подаваний на вхід дерева може містити як числові (порядкові) так і категоріальні змінні. У будь-якому випадку в кожному вузлі розбиття йде тільки за однією змінною. Якщо змінна числового типу, то у вузлі формується правило виду $x_i \le c$, де c – деякий поріг, що найчастіше вибирається як середнє арифметичне двох сусідніх упорядкованих значень змінної x_i навчаючої вибірки. Якщо змінна категоріального типу, то у вузлі формується правило $x_i \in V(x_i)$, де $V(x_i)$ – деяка непорожня підмножина множини значень змінної x_i у навчаючій вибірці. Отже, для n значень числового атрибута метод порів-

нює n–1 розбиттів, а для категоріального ($2^{n-1} - 1$). На кожному кроці побудови дерева метод послідовно порівнює всі можливі розбиття для всіх атрибутів і вибирає найкращий атрибут і найкраще розбиття для нього.

Нехай джерело даних, необхідних для роботи методу, подане плоскою таблицею. Кожен рядок таблиці описує один приклад навчаючої / тестової вибірки.

Кожен крок побудови дерева фактично складається із сукупності трьох трудомістких операцій.

1. Сортування джерела даних за стовпцем є необхідним для обчислення порога, коли розглянутий у поточний момент часу атрибут має числовий тип. На кожному кроці побудови дерева число сортувань буде як мінімум дорівнювати кількості атрибутів числового типу.

2. Поділ джерела даних. Після того, як знайдене найкраще розбиття, необхідно розділити джерело даних відповідно до правила формованого вузла і рекурсивно викликати процедуру побудови для двох половинок джерела даних.

Обидві ці операції пов'язані (якщо діяти прямо) з переміщенням значних обсягів пам'яті. Тут навмисно джерело даних не називається таблицею, тому що можна істотно знизити часові витрати на побудову дерева, якщо використовувати індексоване джерело даних. Звертання до даних у такому джерелі відбувається не прямо, а за допомогою логічних індексів рядків даних. Сортувати і розділяти таке джерело можна з мінімальною втратою продуктивності.

3. Обчислення індексів G_{split} для всіх можливих розбиттів – операція, що займає 60–80 % часу виконання програми. Якщо є n – числових атрибутів і m – прикладів у вибірці, то виходить таблиця $n \times (m-1)$ – індексів, що займає великий обсяг пам'яті. Цього можна уникнути, якщо використовувати один стовпець для поточного атрибута й один рядок для кращих (максимальних) індексів для всіх атрибутів. Можна і зовсім використовувати тільки кілька числових значень, одержавши швидкий код, але такий, що погано читається. Значно збільшити продуктивність можна, якщо використовувати, що $L = N - R$, $l_i = n_i - r_i$, а l_i та r_i змінюються завжди і тільки на одиницю при переході на наступний рядок для поточного атрибута. Тобто підрахунок числа класів, а це основна операція, буде виконуватися швидко, якщо знати число екземплярів кожного класу усього в таблиці і при переході на новий рядок таблиці змінювати на одиницю тільки число екземплярів одного класу – класу поточного приклада.

Усі можливі розбиття для категоріальних атрибутів зручно подавати за аналогією з двійковим поданням числа. Якщо атрибут має n – унікальних значень, то буде 2^n – розбиттів. Перше (де всі нулі) і останнє (всі одиниці) нас не цікавлять, одержуємо 2^n–2. І оскільки порядок множин тут теж неважливий, одержуємо $(2^n - 2)/2$ або $(2^{n-1} - 1)$ перших (з одиниці) двійкових подань. Якщо $\{A, B, C, D, E\}$ – усі можливі значення деякого атрибута X, то для поточного розбиття, що має подання, скажімо {0, 0, 1, 0, 1}, одержуємо правило X in $\{C, E\}$ для правої гілки та [not {0, 0, 1, 0, 1} = {1, 1, 0, 1, 0} = X in $\{A, B, D\}$] для лівої гілки.

Часто значення атрибута категоріального типу подані в базі як строкові значення. У такому випадку швидше і зручніше створити кеш усіх значень атрибута і працювати не зі значеннями, а з індексами в кеші.

Механізм відсікання дерева (minimal cost-complexity tree pruning) – найбільш серйозна відмінність методу CART від інших методів побудови дерева. CART розглядає відсікання як одержання компромісу між двома проблемами: одержанням дерева оптимального розміру й одержанням точної оцінки імовірності помилкової класифікації.

Основна проблема відсікання – велика кількість усіх можливих відсічених піддерев для одного дерева. Більш точно, якщо бінарне дерево має $|T|$ – листів, тоді існує ~$[1{,}5028369^{|T|}]$ відсічених піддерев. І, якщо дерево має хоча б 1000 листів, тоді число відсічених піддерев стає просто величезним.

Базова ідея методу – не розглядати всі можливі піддерева, обмежуючись тільки кращими представниками відповідно до приведеної нижче оцінки.

Позначимо $|T|$ – число листів дерева, $R(T)$ – імовірність помилки класифікації дерева, що дорівнює відношенню числа неправильно класифікованих прикладів до числа прикладів у навчаючій вибірці. Визначимо повну вартість (оцінку / показник витрата – складність) дерева T як: $C_\alpha(T) = R(T)+\alpha|T|$, де α – деякий параметр, що змінюється від 0 до $+\infty$. Повна вартість дерева складається з двох компонентів – помилки класифікації дерева і штрафу за його складність. Якщо помилка класифікації дерева є незмінною, тоді зі збільшенням α повна вартість дерева буде збільшуватися. Тоді в залежності від α менш гіллясте дерево, що дає велику помилку класифікації, може коштувати менше, ніж те, що дає меншу помилку, але є більш гіллястим.

Визначимо $T_{\max}$ – максимальне за розміром дерево, що має бути обрізане. Якщо ми зафіксуємо значення α, тоді існує найменше мінімізоване піддерево α, що задовольняє таким умовам.

1. $C_\alpha(T(\alpha)) = \min\limits_{T \le T_{\max}} C_\alpha(T)$.

2. Якщо $C_\alpha(T) = C_\alpha(T(\alpha))$, то $T(\alpha) \subseteq T$.

Перша умова говорить, що не існує такого піддерева дерева $T_{\max}$, що мало б меншу вартість, ніж $T(\alpha)$ при цьому значенні α. Друга умова говорить, що якщо існує більше одного піддерева, що має дану повну вартість, тоді ми вибираємо найменше дерево.

Можна показати, що для будь-якого значення α існує таке найменше мінімізоване піддерево. Але ця задача не є тривіальною. Вона говорить, що не може бути такого, коли два дерева досягають мінімуму повної вартості і вони є непорівнянними, тобто жодне з них не є піддеревом іншого.

Хоча α має нескінченне число значень, існує кінцеве число піддерев дерева $T_{\max}$. Можна побудувати послідовність зменшуваних піддерев дерева $T_{\max}$: $T_1 > T_2 > T_3 > ... > \{t_1\}$, (де t_1 – кореневий вузол дерева) таку, що T_k – найменше

мінімізоване піддерево для $\alpha \in [\alpha_k, \alpha_{k+1})$. Це важливий результат, тому що це означає, що ми можемо одержати наступне дерево в послідовності, застосувавши відсікання до поточного дерева. Це дозволяє розробити ефективний метод пошуку найменшого мінімізованого піддерева при різних значеннях α. Перше дерево в цій послідовності – найменше піддерево дерева T_{max}, що має таку ж помилку класифікації, як і T_{max}, тобто $T_1 = T(\alpha = 0)$. Якщо розбиття йде доти, поки в кожному вузлі залишиться тільки один клас, то $T_1 = T_{max}$, але, оскільки часто застосовуються методи ранньої зупинки (prepruning), тоді може існувати піддерево дерева T_{max}, що має таку ж помилку класифікації.

Метод обчислення T_1 з T_{max} є простим. Знайти будь-яку пару листів із загальним предком, що можуть бути об'єднані, тобто відсічені в батьківський вузол без збільшення помилки класифікації: $R(t) = R(l) + R(r)$, де r та l – листи вузла t. Продовжувати доти, поки таких пар більше не залишиться. Так ми одержимо дерево, що має таку ж вартість як T_{max} при $\alpha = 0$, але менш гіллясте, ніж T_{max}.

Позначимо як T_t гілку дерева T з кореневим вузлом t.

Якщо ми відітнемо у вузлі t, тоді його внесок у повну вартість дерева $T - T_t$ стане $C_\alpha(\{t\}) = R(t) + \alpha$, де $R(t) = r(t)p(t)$, $r(t)$ – це помилка класифікації вузла t і $p(t)$ – пропорція випадків, що пройшли через вузол t. Альтернативний варіант: $R(t) = m/n$, де m – число прикладів, класифікованих некоректно, а n – загальне число класифікованих прикладів для всього дерева.

Внесок T_t у повну вартість дерева T складе $C_\alpha(T_t) = R(T_t) + \alpha|T_t|$,

де

$$R(T_t) = \sum_{t' \in T_t} R(t').$$

Дерево $T - T_t$ буде кращим ніж T, коли $C_\alpha(\{t\}) = C_\alpha(T_t)$, оскільки при цій величині α вони мають однакову вартість, але $T - T_t$ найменше з двох. Коли $C_\alpha(\{t\}) = C_\alpha(T_t)$ ми одержуємо: $R(T_t) + \alpha|T_t| = R(t) + \alpha$, вирішуючи для α, одержуємо: $\alpha = (R(t) - R(T_t))/(|T_t| - 1)$.

Оскільки для будь-якого вузла t у T_1, якщо ми збільшуємо α, тоді коли $\alpha = (R(t) - R(T_{1,t}))/(|T_{1,t}| - 1)$, дерево, отримане відсіканням у вузлі t, буде кращим ніж T_1.

Обчислимо це значення α для кожного вузла в дереві T_1, і потім виберемо слабкі зв'язки (їх може бути більше одного), тобто вузли для яких величина $g(t) = (R(t) - R(T_{1,t}))/(|T_{1,t}| - 1)$ є найменшою. Ми відтинаємо T_1 у цих вузлах, щоб одержати T_2 – наступне дерево в послідовності. Потім ми продовжуємо цей процес для отриманого дерева і так поки ми не одержимо кореневий вузол (дерево в якого є тільки один вузол).

Метод обчислення послідовності дерев.

Крок 1. Установити: $T_1 = T(\alpha = 0)$, $\alpha_1 = 0$, $k = 1$.

Крок 2. Поки T_k більше ніж дерево, що складається тільки з одного вузла – кореня, виконувати кроки 2.1–2.4.

Крок 2.1 Для всіх нетермінальних вузлів у $t \in T_k$:

$$g_k(t) = (R(t) - R(T_{k,t}))/(|T_{k,t}| - 1).$$

Крок 2.2 Установити: $\alpha_{k+1} = \min_t g_k(t)$.

Крок 2.3 Обійти униз усі вузли й обрізати ті, де $g_k(t)=\alpha_{k+1}$, щоб одержати T_{k+1}.

Крок 2.4 Установити: $k = k + 1$. Перейти до кроку 2.

Вузли необхідно обходити вниз, щоб не відтинати вузли, що відсічуться самі собою, у результаті відсікання n-го предка.

Вибір фінального дерева.

Отже, ми маємо послідовність дерев і нам необхідно вибрати краще дерево з неї – те, що ми і будемо використовувати надалі. Найбільш очевидним є вибір фінального дерева через тестування на тестовій вибірці. Дерево, що дало мінімальну помилку класифікації, і буде кращим. Однак, це не єдиний можливий шлях.

Природно, якість тестування багато в чому залежить від обсягу тестової вибірки і рівномірності даних, що потрапили в навчаючу і тестову вибірки.

Часто можна спостерігати, що послідовність дерев дає помилки близькі одна до одної. Ця довга плоска послідовність є дуже чуттєвою до даних, що будуть обрані як тестова вибірка. Щоб зменшити цю нестабільність CART використовує *1–SE правило*: вибирається мінімальне за розміром дерево з R^{ts} у межах інтервалу [min R^{ts}, min R^{ts} + SE], де R^{ts} – помилка класифікації дерева, SE – стандартна помилка, що є оцінкою реальної помилки: $SE(R^{ts}) = (R^{ts}(1–R^{ts}) / n_{test})^{0,5}$, де n_{test} – число прикладів у тестовій вибірці.

Перехресна перевірка (V-fold cross-validation) – найоригінальніша і найскладніша частина методу CART. Цей шлях вибору фінального дерева використовується, коли набір даних для навчання малий або кожний запис у ньому по своєму унікальний так, що ми не можемо виділити вибірку для навчання і вибірку для тестування.

У такому випадку будуємо дерево на всіх даних, обчислюємо α_1, α_2, ..., α_k та $T_1 > T_2 > ... > T_N$. Позначимо T_k – найменше мінімізоване піддерево для $\alpha \in [\alpha_k; \alpha_{k+1})$.

Тепер ми хочемо вибрати дерево з послідовності, але уже використовували всі наявні дані. Хитрість у тому, що ми збираємося обчислити помилку дерева T_k із послідовності непрямим шляхом.

Крок 1. Установимо: $\beta_1=0$, $\beta_2 = \sqrt{\alpha_2\alpha_3}$, $\beta_3 = \sqrt{\alpha_3\alpha_4}$, ..., $\beta_{N-1} = \sqrt{\alpha_{N-1}\alpha_N}$, $\beta_N = \infty$. Вважається, що β_k буде типовим значенням для $[\alpha_k; \alpha_{k+1})$ і, отже, як значення відповідає T_k.

Крок 2. Розділимо весь набір даних на V груп однакового розміру G_1, G_2, ..., G_V. Брейман рекомендує брати $V = 10$. Потім для кожної групи G_i:

Крок 2.1 Обчислити послідовність дерев за допомогою описаного вище механізму відсікання на всіх даних, крім G_i, і визначити $T^{(i)}(\beta_1)$, $T^{(i)}(\beta_2)$, ..., $T^{(i)}(\beta_N)$ для цієї послідовності.

Крок 2.2 Обчислити помилку дерева $T^{(i)}(\beta_k)$ на G_i. Тут $T^{(i)}(\beta_k)$ означає найменше мінімізоване піддерево з послідовності, побудоване на всіх даних, крім G_i для $\alpha = \beta_k$.

Крок 3. Для кожного β_k підсумовувати помилку $T^{(i)}(\beta_k)$ за всіма G_i ($i = 1, ..., V$). Нехай β_h буде з найменшою загальною помилкою. Оскільки β_h відповідає дереву T_h, ми вибираємо T_h з послідовності, побудованої на всіх даних як фінальне дерево. Показник помилки, обчислений за допомогою перехресної перевірки, можна використовувати як оцінку помилки дерева.

Альтернативний шлях – щоб вибрати фінальне дерево з послідовності на останньому кроці можна знову використовувати 1–SE правило.

Метод обробки пропущених значень.

Більшість методів інтелектуального аналізу даних припускає відсутність пропущених значень. У практичному аналізі це припущення часто є невірним. До пропущених даних можуть привести такі причини:

1. Респондент не бажає відповідати на деякі з поставлених питань.

2. Помилки при введенні даних.

3. Об'єднання не зовсім еквівалентних наборів даних.

Найбільш загальне рішення – відкинути дані, що містять один чи кілька порожніх атрибутів. Однак це рішення має свої недоліки:

1. Зсув даних. Якщо викинуті дані лежать трохи осторонь від залишених, тоді аналіз може дати упереджені результати.

2. Зменшення потужності. Може виникнути ситуація, коли прийдеться викинути багато даних. У такому випадку точність прогнозу сильно зменшується.

Якщо необхідно будувати і використовувати дерево на неповних даних, тоді необхідно вирішити питання:

1. Як визначити якість розбиття?

2. У яку гілку необхідно послати спостереження, якщо пропущено змінну, на яку приходиться найкраще розбиття (побудова дерева і тренування)?

Помітимо, що спостереження з пропущеною міткою класу є непотрібним для побудови дерева і буде викинуто.

Щоб визначити якість розбиття CART просто ігнорує пропущені значення. Для вирішення, по якому шляху посилати спостереження з пропущеної змінної, утримуючої найкраще розбиття, CART обчислює так називане сурогатне розбиття. Воно створює найбільш близькі до кращої підмножини прикладів у поточному вузлі. Щоб визначити значення альтернативного розбиття як сурогатного ми створюємо таку крос-таблицю.

$p(l^*, l')$	$p(l^*, r')$
$p(r^*, l')$	$p(r^*, r')$

У цій таблиці $p(l^*,l')$ позначає пропорцію випадків, що будуть послані в ліву гілку при кращому s^* і альтернативній розбитті s' і аналогічно для $p(r^*,r')$, оскільки $p(l^*,l') + p(r^*,r')$ – пропорція випадків, що послані в ту саму гілку для обох розбиттів. Це міра подібності розбиттів або інакше це говорить, наскільки добре ми прогнозуємо шлях, по якому посланий випадок найкращим розбиттям, дивлячись на альтернативне розбиття. Якщо $p(l^*,l') + p(r^*,r') < 0{,}5$, тоді ми можемо одержати кращий сурогат, помінявши ліву і праву гілки для альтернативного розбиття. Крім того, необхідно помітити, що пропорції в таблиці обчислені, коли обидві змінні (сурогатна й альтернативна) є спостережними.

Альтернативні розбиття з $p(l^*,l') + p(r^*,r') > \max(p(l^*), p(r^*))$ відсортовані в спадному порядку подібності. Тепер, якщо пропущено змінну кращого розбиття, тоді використовуємо першу із сурогатних у списку, якщо пропущена вона, тоді наступну і т. д. Якщо пропущено всі сурогатні змінні, то використовуємо $\max(p(l^*), p(r^*))$.

Регресія.

Побудова дерева регресії багато в чому є схожою з деревом класифікації. Спочатку ми будуємо дерево максимального розміру, потім обрізаємо дерево до оптимального розміру.

Основна перевага дерев у порівнянні з іншими методами регресії – можливість працювати з багатомірними задачами і задачами, у яких є присутньою залежність вихідної змінної від змінних категоріального типу.

Основна ідея – розбиття всього простору на прямокутники, необов'язкового однакового розміру, у яких вихідна змінна вважається постійною. Помітимо, що існує сильна залежність між обсягом навчаючої вибірки і помилкою відповіді дерева.

Процес побудови дерева відбувається послідовно.

На першому кроці ми одержуємо регресійну оцінку просто як константу по всьому простору прикладів. Константу розраховуємо як середнє арифметичне вихідної змінної у навчаючій вибірці. Отже, якщо ми позначимо всі значення вихідної змінної як Y_1, Y_2, ..., Y_n, тоді регресійна оцінка виходить:

$$f(x_i) = \frac{I_R(x_i)}{n} \sum_{i=1}^{n} Y_i,$$

де R – простір навчальних прикладів, n – число прикладів, $I_R(x)$ – індикаторна функція простору – фактично, набір правил, що описують попадання змінної x_i у простір. Ми розглядаємо простір R як прямокутник. На другому кроці ми поділяємо простір на дві частини. Вибирається деяка змінна x_i і якщо змінна числового типу, тоді ми визначаємо: $R_1=\{x_i \in R: x_i \le a\}$, $R_2=\{x_i \in R: x_i > a\}$.

Якщо x_i категоріального типу з можливими значеннями A_1, A_2, ... , A_q, тоді вибирається деяка підмножина $I \subset \{A_1, \dots A_n\}$ і ми визначаємо: $R_1=\{x_i \in R: x_i \in I\}$, $R_2=\{x_i \in R: x_i \in \{A_1, A_2, \dots, A_q,\} \backslash I\}$.

Регресійна оцінка приймає вид:

$$f(x_i) = \frac{I_{R_1}(x_i)}{\left|I_1\right|}\sum_{I_1} Y_i + \frac{I_{R_2}(x_i)}{\left|I_2\right|}\sum_{I_2} Y_i,$$

де $I_1 = \{i, x_i \in R_1\}$, $|I_1|$ – число елементів у I_1, $I_2 = \{i, x_i \in R_2\}$, $|I_2|$ – число елементів у I_2.

Краще розбиття вибирається в такий спосіб. Як оцінка тут слугує сума квадратів різниць:

$$E = \sum_{i=1}^{n} (Y_i - f(x_i))^2.$$

Вибирається розбиття з мінімальною сумою квадратів різниць.

Ми продовжуємо розбиття доти, поки в кожному підпросторі не залишиться мале число прикладів або сума квадратів різниць не стане менше деякого порога.

Відсікання, вибір фінального дерева відбуваються аналогічно дереву класифікації. Єдина відмінність – визначення помилки відповіді дерева: $R(f) = E/n$, чи, інакше кажучи, середньоквадратичної помилки відповіді.

Вартість дерева дорівнює: $C_\alpha(f) = R(f) + \alpha|f|$.

Інші операції відбуваються аналогічно дереву класифікації.

Метод CART успішно поєднує у собі якість побудованих моделей і, при вдалій реалізації, високу швидкість їхньої побудови. Містить у собі унікальні методики обробки пропущених значень і побудови оптимального дерева сукупністю методів cost-complexity pruning і V-fold cross-validation.

Існує також кілька модифікованих *версій методу CART*.

Метод IndCART, є частиною пакета Ind і відрізняється від CART використанням іншого способу обробки пропущених значень, не здійснює регресійну частину методу CART і має інші параметри відсікання.

Метод DB-CART (distribution based CART) базується на такій ідеї: замість того щоб використовувати навчаючий набір даних для визначення розбиттів, використовуємо його для оцінки розподілу вхідних і вихідних значень і потім використовуємо цю оцінку, щоб визначити розбиття. Стверджується, що ця ідея дає значне зменшення помилки класифікації, у порівнянні зі стандартними методами побудови дерева.

Перевагами дерев рішень є: швидкий процес навчання; генерація правил в галузях, де експерту важко формалізувати свої знання; витяг правил природною мовою; інтуїтивно зрозуміла класифікаційна модель; висока точність прогнозу, порівнянний з іншими методами (статистика, нейронні мережі); побудова непараметричних моделей.

У силу цих і багатьох інших причин, методологія дерев рішень є важливим інструментом у роботі кожного фахівця, що займається аналізом даних, поза залежністю від того практик він або теоретик.

2.3.10 Асоціативні правила

Асоціативні правила дозволяють виявляти закономірності між пов'язаними подіями і є одним з інструментів видобування знань з масивів даних.

Нехай $I = \{i_1, i_2, ..., i_m\}$ – множина елементів, D – множина транзакцій, де кожна транзакція T – це набір елементів з I, $T \subseteq I$. Кожна транзакція – це множина елементів (подій), що відбулися одночасно, і являє собою бінарний вектор, де $t[k]=1$, якщо i_k елемент присутній у транзакції, інакше $t[k]=0$. Транзакція Т містить X, певний набір елементів з I, якщо $X \subseteq T$.

Розширеною транзакцією називається транзакція, розширена предками всіх елементів, що входять у цю транзакцію.

Асоціативним правилом (association rule) називається імплікація $X \rightarrow Y$, де $X \subset I$, $Y \subset I$ та $X \cap Y = \varnothing$.

Підтримкою (support) правила $X \rightarrow Y$ називається частка s, якщо s % транзакцій з D, містять $X \cup Y$, $\text{supp}(X \rightarrow Y) = \text{supp}(X \cup Y)$.

Вірогідність правила показує яка імовірність того, що з X випливає Y. Правило $X \rightarrow Y$ є справедливим з вірогідністю (confidence) с, якщо c % транзакцій з D, що містять X, також містять Y, $\text{conf}(X \rightarrow Y)=\text{supp}(X \rightarrow Y)/\text{supp}(X)$.

Іншими словами, метою аналізу є встановлення таких залежностей: якщо в транзакції зустрівся деякий набір елементів X, то на підставі цього можна зробити висновок про те, що інший набір елементів Y також повинний з'явитися в цій транзакції. Установлення таких залежностей дає нам можливість знаходити дуже прості й інтуїтивно зрозумілі правила.

Методи пошуку асоціативних правил призначені для знаходження всіх правил $X \rightarrow Y$, причому підтримка і вірогідність цих правил повинні бути вище деяких наперед визначених порогів, називаних відповідно *мінімальною підтримкою* (minsupport) і *мінімальною вірогідністю* (minconfidence).

Перший метод пошуку асоціативних правил, що називався *AIS* був розроблений у 1993 році співробітниками дослідницького центра IBM Almaden.

Задача знаходження асоціативних правил розбивається на дві підзадачі.

1. Знаходження всіх наборів елементів, що задовольняють порогові minsupport. Такі набори елементів називаються такими, що часто зустрічаються.

2. Генерація правил зі знайдених наборів елементів з вірогідністю, що задовольняє порогові minconfidence.

Один з перших методів, що ефективно вирішують подібний клас задач, – це метод *APriori*. Крім цього методу останнім часом був розроблений ряд інших методів: *DHP*, *Partition*, *DIC* та інші.

Значення для параметрів minsupport та minconfidence вибираються таким чином, щоб обмежити кількість знайдених правил. Якщо підтримка має велике значення, то методи будуть знаходити правила, добре відомі аналітикам або настільки очевидні, що немає ніякого сенсу проводити такий аналіз. З іншого боку, ни-

зьке значення підтримки веде до генерації величезної кількості правил, що, звичайно, вимагає істотних обчислювальних ресурсів. Проте, більшість цікавих правил знаходиться саме при низькому значенні порога підтримки. Хоча занадто низьке значення підтримки веде до генерації статистично необґрунтованих правил.

Пошук асоціативних правил – це зовсім не тривіальна задача, як може показатися на перший погляд. Одна з проблем – алгоритмічна складність при знаходженні наборів елементів, що часто зустрічаються, оскільки з ростом числа елементів I експоненційно росте число потенційних наборів елементів.

Узагальненим асоціативним правилом (generalized association rule) називається імплікація $X \rightarrow Y$, де $X \subset I$, $Y \subset I$ та $X \cap Y = \varnothing$ і де жоден з елементів, що входять у набір Y, не є предком жодного елемента, що входить у X. Підтримка і вірогідність підраховуються так само, як і у випадку асоціативних правил. Ми називаємо правило $X \rightarrow Y$ узагальненим, тому що елементи, які входять у нього можуть знаходитися на будь-якому рівні таксономії. Також будемо називати $\underline{x}$ предком x та x нащадком $\underline{x}$.

Основною відмінністю узагальнених асоціативних правил від асоціативних правил є те, що одержувані правила включають елементи, що є предками елементів, які входять у множину транзакцій.

Ієрархією (таксономією) елементів називається ліс спрямованих ациклічних дерев, листами яких є елементи транзакцій, а внутрішніми вузлами – групи елементів.

Уведення додаткової інформації про групування елементів у виді ієрархії допомагає установити асоціативні правила не тільки між окремими елементами, але і між різними рівнями ієрархії (групами).

Окремі елементи можуть мати недостатню підтримку, але в цілому група може задовольняти порогові minsupport. Це приводить до того, що раніше не виявлене потенційно цікаве правило, побудоване на елементах нижнього рівня ієрархії, може бути отримано, але його елементами будуть або елементи транзакції, або предки цих елементів.

Введення інформації про групування елементів може використовуватися для відсікання «нецікавих» правил.

Для знаходження таких правил можна використовувати кожний з вищезгаданих методів. Для цього кожну транзакцію потрібно доповнити всіма предками кожного елемента, що входить у транзакцію. Однак, пряме застосування цих методів неминуче приведе до таких проблем.

– Елементи на верхніх рівнях ієрархії прагнуть до значно більших значень підтримки в порівнянні з елементами на нижніх рівнях.

– З додаванням у транзакції груп збільшується кількість атрибутів і, відповідно, розмірність вхідного простору. Це ускладнює задачу, а також веде до генерації більшої кількості правил.

– Поява надлишкових правил, що суперечать визначенню узагальненого асоціативного правила. Отже, потрібні спеціальні оператори, що видаляють подібні надлишкові правила.

Виявлення узагальнених асоціативних правил.

Нехай I – ліс спрямованих дерев. Дуги в I – це залежності між елементами. Нехай елементи, що належать I, розташовані в деякій ієрархії. Якщо є дуга від a до b, то говорять, що a – предок b та b – нащадок a (a – це узагальнення b).

Необхідно знайти закономірності, що є узагальненими асоціативними правилами виду $X \to Y$, причому $\text{supp}(X \to Y) \geq \text{minsupport}$ та $\text{conf}(X \to Y) \geq \text{minconfidence}$.

Це визначення задачі має одну проблему. Справа в тому, що при такому визначенні задачі, будуть знайдені «зайві» узагальнені асоціативні правила. Для вирішення цієї проблеми розглянемо такий параметр правила як *рівень інтересу*.

Визначення «цікавих» правил.

Нехай $\underline{Z}$ – це предок Z, де Z та $\underline{Z}$ – множини елементів, що входять в ієрархію (Z, $\underline{Z} \subseteq I$). $\underline{Z}$ є предком Z, тільки в тому випадку, якщо $\underline{Z}$ можна одержати з Z шляхом підміни одного чи декількох елементів їхніми предками. Будемо називати правила $\underline{X} \to Y$, $X \to \underline{Y}$, $\underline{X} \to \underline{Y}$ предками правила $X \to Y$.

Правило $\underline{X} \to \underline{Y}$ є найближчим предком правила $X \to Y$, якщо не існує такого правила $X' \to Y'$, що $X' \to Y'$ – це предок $X \to Y$ та $\underline{X} \to \underline{Y}$ – це предок $X' \to Y'$.

Подібні визначення можна дати і для правил: $\underline{X} \to Y$, $X \to \underline{Y}$.

Нехай $\Pr(X)$ – це імовірність того, що всі елементи з X містяться в одній розширеній транзакції. Тоді $\text{supp}(X \cup Y) = \Pr(X \cup Y)$ та $\text{conf}(X \to Y) = \Pr(Y|X)$. Якщо підтримка $\{x,y\}$ більше значення мінімальної підтримки, то і підтримка $\{\underline{x},y\}$, і підтримка $\{x,\underline{y}\}$, і підтримка $\{\underline{x},\underline{y}\}$ будуть більше порога мінімальної підтримки. Однак якщо вірогідність правила $X \to Y$ більше мінімальної вірогідності, тільки правило $X \to \underline{Y}$ гарантовано буде мати вірогідність більшу ніж мінімальна. Підтримка елемента, узятого з внутрішнього рівня ієрархії, не дорівнює сумі підтримок елементів, що є безпосередніми нащадками цього елемента.

Розглянемо правило $X \to Y$. Нехай $Z = X \cup Y$. Помітимо, що $\text{supp}(X \to Y) = \text{supp}(Z)$. Назвемо $E_{\underline{Z}}[\Pr(Z)]$ очікуваним значенням $\Pr(Z)$ відносно $\underline{Z}$. Нехай $Z = \{z_1, ..., z_n\}$, $\underline{Z} = \{\underline{z}_1, ..., \underline{z}_j, \underline{z}_{j+1}, ..., \underline{z}_n\}$, $1 \leq j \leq n$. Тоді можна визначити:

$$E_{\underline{Z}}\left[\Pr(Z)\right] = \frac{\Pr(z_1)}{\Pr(\underline{z}_1)} \times ... \times \frac{\Pr(z_j)}{\Pr(\underline{z}_j)} \times \Pr(\underline{Z}) .$$

Аналогічно $E_{\underline{X} \to \underline{Y}}[\Pr(Y|X)]$ визначимо як очікуване значення вірогідності правила $X \to Y$ відносно $\underline{X} \to \underline{Y}$. Нехай $Y = \{y_1, ..., y_n\}$, $\underline{Y} = \{\underline{y}_1, ..., \underline{y}_j, \underline{y}_{j+1}, ..., \underline{y}_n\}$, $1 \leq j \leq n$. Тоді можна визначити:

$$E_{\underline{X} \to \underline{Y}}\left[\Pr(Y \mid X)\right] = \frac{\Pr(y_1)}{\Pr(\underline{y}_1)} \times ... \times \frac{\Pr(y_j)}{\Pr(\underline{y}_j)} \times \Pr(Y \mid X) .$$

Правило $X \to Y$ називається *R-цікавим* щодо правила-предка, якщо підтримка правила $X \to Y$ у R разів більше очікуваної підтримки правила $X \to Y$ щодо предка або якщо вірогідність правила $X \to Y$ у R разів більше очікуваної вірогідності правила $X \to Y$ щодо правила-предка.

Цікавим називається правило, якщо в нього немає предків або воно є R-цікавим щодо усіх своїх найближчих предків.

Частково цікавим називається правило, якщо в нього немає предків або воно є *R*-цікавим щодо будь-якого свого найближчого предка.

Тепер задачу виділення узагальнених асоціативних правил можна сформулювати по новому: необхідно знайти закономірності, що є узагальненими асоціативними правилами виду $X \to Y$, причому підтримка правила $X \to Y$ більше або дорівнює деякому наперед заданому значенню мінімальної підтримки і вірогідність більше або дорівнює значенню мінімальної вірогідності і правила $X \to Y$ є цікавими або частково цікавими.

Метод обчислення узагальнених асоціативних правил можна розбити на кілька етапів.

Етап 1. Пошук множин елементів, що часто зустрічаються, підтримка яких більше ніж заданий поріг підтримки (мінімальна підтримка).

Етап 2. Обчислення правил на основі знайдених на попередньому етапі множин елементів, що часто зустрічаються. Основна ідея обчислення правил на основі множин, що часто зустрічаються, полягає в такому: якщо $ABCD$ – це множина елементів, що часто зустрічається, то на основі цієї множини можна побудувати правила $X \to Y$ (наприклад, $AB \to CD$), причому $X \cup Y = ABCD$. Підтримка правила дорівнює підтримці множини, що часто зустрічається. Вірогідність правила обчислюється за формулою $\text{conf}(X \to Y) = \text{supp}(X \to Y)/\text{supp}(X)$. Правило додається до результуючого списку правил, якщо вірогідність цього правила більше порога minconf.

Етап 3. З результуючого списку правил видаляються всі «нецікаві» правила.

Базовий метод пошуку множин, що часто зустрічаються.

На першому кроці методу підраховуються 1-елементні набори, що часто зустрічаються. При цьому елементи можуть знаходитися на будь-якому рівні таксономії. Для цього необхідно переглянути весь набір даних і підрахувати для них підтримку, тобто скільки разів зустрічаються в базі.

Наступні кроки будуть складатися з двох частин: генерації потенційних наборів елементів, що часто зустрічаються, (їх називають кандидатами) і підрахунку підтримки для кандидатів.

Даний метод можна записати як послідовність кроків.

Крок 1. Виділити і занести в L_1 множини елементів і груп елементів, що часто зустрічаються. Установити: $k = 2$.

Крок 2. Якщо $L_{k-1} \neq \varnothing$, тоді перейти до кроку 3, у протилежному випадку – перейти до кроку 7.

Крок 3. Згенерувати C_k – множину кандидатів потужністю k на основі L_{k-1}.

Крок 4. Для всіх транзакцій $t \in D$ виконати кроки 4.1–4.3.

Крок 4.1 Розширити транзакцію t предками всіх елементів, що входять у транзакцію.

Крок 4.2 Видалити дублікати з транзакції t.

Крок 4.3 Для всіх кандидатів $c \in C_k$ виконати: якщо $c \subseteq t$, то установити: c.count = c.count+1.

Крок 5. Зробити відбір кандидатів: $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsupport}\}$.

Крок 6. Установити: $k = k+1$. Перейти до кроку 3.

Крок 7. Зупинення. Повернути як результат $\bigcup L_k$.

Функція генерації кандидатів. Для того щоб одержати k-елементні набори, скористаємося (k–1)-елементними наборами, які були визначені на попередньому кроці і є такими, що часто зустрічаються.

Метод генерації кандидатів буде складатися з двох кроків.

Крок 1. Об'єднання. Кожний кандидат C_k буде формуватися шляхом розширення набору, що часто зустрічається, розміром (k–1) додаванням елемента з іншого (k–1)-елементного набору: включити до C_k ті елементи a_1, a_2, ..., a_{k-1}, b_{k-1} з L_{k-1}: а, $b \in L_{k-1}$, для яких: $a_1 = b_1$, $a_2 = b_2$, ... , $a_{k-2} = b_{k-2}$, $a_{k-1} < b_{k-1}$.

Крок 2. Видалення надлишкових правил. На підставі властивості антимонотонності, варто видалити всі набори з C_k, якщо хоча б одна з його (k–1) підмножин не є такою, що часто зустрічається.

Геш-дерево можна використовувати для ефективного підрахунку підтримки кандидатів.

Геш-дерево будується щоразу, коли формуються кандидати. Первісне дерево складається тільки з кореня, що є листом, і не містить ніяких кандидатів-наборів. Щоразу, коли формується новий кандидат, він заноситься в корінь дерева і так доти, поки кількість кандидатів у корні-листі не перевищить деякий поріг. Як тільки кількість кандидатів стає більше порога, корінь перетвориться в геш-таблицю, тобто стає внутрішнім вузлом, і для нього створюються нащадки-листи. І всі приклади розподіляються по вузлах-нащадках відповідно до гешзначення елементів, що входять у набір, і т. д. Кожний новий кандидат гешується на внутрішніх вузлах, поки він не досягне першого вузла-листа, де він і буде зберігатися, поки кількість наборів знову ж не перевищить порога.

Поліпшений метод пошуку множин, що часто зустрічаються

До базового методу можна додати кілька оптимизацій, що збільшать швидкість роботи базового методу.

Доцільно один раз обчислити множини предків для кожного елемента ієрархії як для елемента нижнього рівня таксономії (лист дерева), так і для елемента внутрішнього рівня.

Необхідно видаляти кандидати, що містять елемент і його предка.

Підтримка множини X, що містить і елемент x і його предка $\underline{x}$, обчислюється за формулою: supp(X)=supp(x–$\underline{x}$). Беручи це до уваги, будемо видаляти кандидати, що містять і елемент і його предка зі списку кандидатів до початку процесу підрахунку підтримки.

Якщо L_k – це список множин, що часто зустрічаються, який не містить множини, що включають і елемент і його предка в одній множині, то C_{k+1} (список кандидатів, одержуваних з L_k) також не буде містити множини, що включають і елемент і його предка. З огляду на це, будемо видаляти кандидатів, що включають і елемент і його предка, зі списку кандидатів тільки на першій ітерації зовнішнього циклу.

Немає необхідності в додаванні всіх предків всіх елементів, що входять у транзакцію. Якщо який-небудь елемент, у якого є предок, не знаходиться в списку кандидатів, то в списку елементів із предками він позначається як вилучений. Отже, із транзакції видаляються елементи, позначені як вилучені, або здійснюється заміна цих елементів на їхніх предків. До транзакції додаються тільки невилучені предки.

Не будемо пропускати транзакцію через ґеш-дерево, якщо її потужність менше ніж потужність елементів, розташованих у ґеш-дереві.

Доцільно позначати транзакції як вилучені і не використовувати їх при підрахунку підтримки на наступних ітераціях, якщо на поточній ітерації в цю транзакцію не ввійшов жоден кандидат.

З огляду на написане вище, одержуємо такий метод.

Крок 1. Обчислити I^* – множини предків елементів для кожного елемента. Виділити і занести в L_1 множини елементів і груп елементів, що часто зустрічаються. Установити: $k = 2$.

Крок 2. Якщо $L_{k-1} \neq \varnothing$, тоді перейти до кроку 3, у протилежному випадку – перейти до кроку 9.

Крок 3. Згенерувати C_k – множину кандидатів потужністю k на основі L_{k-1}.

Крок 4. Якщо $k = 2$, то видалити ті кандидати з C_k, що містять елемент і його предка.

Крок 5. Позначити як вилучені множини предків елемента, що не міститься в списку кандидатів.

Крок 6. Для всіх транзакцій $t \in D$ виконати кроки 6.1–6.3.

Крок 6.1 Для кожного елемента $x \in t$: додати всіх предків x з I^* до t.

Крок 6.2 Видалити дублікати з транзакції t.

Крок 6.3 Якщо транзакція t не позначена як вилучена та $|t| \geq k$, то виконати кроки 6.3.1–6.3.2.

Крок 6.3.1 Для всіх кандидатів $c \in C_k$: якщо $c \in C_k$, то установити c.count = c.count+1.

Крок 6.3.2 Якщо в транзакцію t не ввійшов жоден кандидат, то позначити цю транзакцію як вилучену.

Крок 7. Виконати відбір кандидатів: $L_k = \{c \geq C_k \mid c.\text{count} \geq \text{minsupport}\}$.

Крок 8. Установити: $k = k + 1$. Перейти до кроку 2.

Крок 9. Зупинення. Повернути як результат $\bigcup L_k$.

Масштабувальний метод пошуку асоціативних правил Apriori

Для того, щоб було можливо застосувати метод Apriori, необхідно провести передобробку даних: по-перше, привести всі дані до бінарного виду; по-друге, змінити структуру даних – подати їх у виді таблиці. Кількість стовпців у таблиці дорівнює кількості елементів, що є присутніми у множині транзакцій *D*. Кожний запис відповідає транзакції, де у відповідному стовпці стоїть 1, якщо елемент присутній у транзакції, і 0 – у протилежному випадку. Всі елементи повинні бути упорядковані за абеткою (якщо це числа, вони повинні бути упорядковані в числовому порядку).

На першому кроці методу необхідно знайти набори елементів, що часто зустрічаються, а потім, на другому кроці, витягти з них правила. Кількість елементів у наборі будемо називати розміром набору, а набір, що складається з *k* елементів, – *k*-елементним набором.

Виявлення наборів елементів, що часто зустрічаються, – операція, що вимагає багато обчислювальних ресурсів і, відповідно, часу. Примітивний підхід до вирішення даної задачі – простий перебір усіх можливих наборів елементів. Це вимагає $O(2^{|I|})$ операцій, де $|I|$ – кількість елементів.

Властивість антимонотонності використовується в методі Apriori і слугує для зниження розмірності простору пошуку: підтримка будь-якого набору елементів не може перевищувати мінімальної підтримки кожної з його підмножин. Властивості антимонотонності можна дати й інше формулювання: з ростом розміру набору елементів підтримка зменшується, або залишається такою ж. З усього вищесказаного випливає, що будь-який *k*-елементний набір буде таким, що часто зустрічається, тоді і тільки тоді, коли всі його $(k-1)$-елементні підмножини будуть такими, що часто зустрічаються.

Усі можливі набори елементів з *I* можна уявити у вигляді ґрат, що починаються з порожньої множини, потім на 1-му рівні – 1-елементні набори, на 2-му – 2-елементні і т. д. На *k*-му рівні подані *k*-елементні набори, пов'язані з усіма своїми $(k-1)$-елементними підмножинами.

На першому кроці методу підраховуються одноелементні набори, що часто зустрічаються. Для цього необхідно пройтися по всьому набору даних і підрахувати для них підтримку, тобто скільки разів зустрічаються в базі.

Наступні кроки будуть складатися з двох частин: генерації потенційних наборів елементів, що часто зустрічаються, (їх називають кандидатами) і підрахунку підтримки для кандидатів.

Даний метод можна записати як таку послідовность кроків.

Крок 1. Задати F_1 – множину одноелементних наборів, що часто зустрічаються.

Крок 2. Установити: $k = 2$.

Крок 3. Якщо $F_{k-1} \neq \varnothing$, тоді перейти до кроку 4, у протилежному випадку – до кроку 8.

Крок 4. Виконати генерацію кандидатів C_k на основі F_{k-1}.

Крок 5. Для всіх транзакцій $t \in D$ виконати кроки 5.1–5.2.

Крок 5.1 Виконати видалення надлишкових правил з C_k, одержавши множини $C_t \subseteq C_k$.

Крок 5.2. Для всіх кандидатів $c \in C_t$: установити c.count = c.count +1.

Крок 6. Виконати відбір кандидатів: $F_k = \{c \in C_k \mid c.\text{count} \geq \text{minsupport}\}$.

Крок 7. Установити: $k = k+1$. Перейти до кроку 3.

Крок 8. Зупинення. Повернути як результат $\bigcup F_k$.

Функція генерації кандидатів. У даному випадку немає ніякої необхідності знову звертатися до бази даних. Для того, щоб одержати k-елементні набори, скористаємося (k–1)-елементними наборами, що були визначені на попередньому кроці і є такими, що часто зустрічаються.

Відзначимо, що вихідний набір зберігається в упорядкованому виді. Генерація кандидатів також буде складатися з двох кроків.

Крок 1. Об'єднання. Кожен кандидат C_k буде формуватися шляхом розширення набору, що часто зустрічається, розміром (k–1) додаванням елемента з іншого (k–1)-елементного набору: включити в C_k ті елементи $p_1, p_2, ..., p_{k-1}, q_{k-1}$ з F_{k-1}: $p, q \in F_{k-1}$, для яких $p_1 = q_1, p_2 = q_2, ..., p_{k-2} = q_{k-2}, p_{k-1} < q_{k-1}$.

Крок 2. Видалення надлишкових правил. На підставі властивості антимонотонності, варто видалити всі набори $c \in C_k$, якщо хоча б одна з його (k–1) підмножин не є такою, що часто зустрічається.

Після генерації кандидатів наступною задачею є підрахунок підтримки для кожного кандидата. Очевидно, що кількість кандидатів може бути дуже великою і потрібний ефективний спосіб підрахунку. Самий тривіальний спосіб – порівняти кожну транзакцію з кожним кандидатом. Але це далеко не краще рішення. Набагато швидше й ефективніше використовувати підхід, заснований на збереженні кандидатів у ґеш-дереві. Внутрішні вузли дерева містять ґеш-таблиці з покажчиками на нащадків, а листи – на кандидатів.

Підтримку для кожного кандидата легко підрахувати використовуючи ґеш-дерево. Для цього потрібно пропустити кожну транзакцію через дерево і збільшити лічильники для тих кандидатів, чиї елементи також містяться й у транзакції, тобто $C_k \cap T_i = C_k$. На кореневому рівні ґеш-функция застосовується до кожного елемента з транзакції. Далі, на другому рівні, ґеш-функция застосовується до других елементів і т. д. На k-рівні ґешується k-й елемент. І так доти, поки не досягнемо листа. Якщо кандидат, що зберігається в листі, є підмножиною розглянутої транзакції, тоді збільшуємо лічильник підтримки цього кандидата на одиницю.

Після того, як кожна транзакція з вихідного набору даних пропущена через дерево, можна перевірити чи задовольняють значення підтримки кандидатів мінімальному порогові. Кандидати, для яких ця умова виконується, переносяться в розряд таких, що часто зустрічаються. Крім того, варто запам'ятати і підтримку набору, вона буде корисною при витягу правил. Ці ж дії застосовуються для знаходження (k+1)-елементних наборів і т. д.

Витяг правил виконують після того, як знайдені всі набори елементів, що часто зустрічаються. Це менш трудомістка задача. По-перше, для підрахунку вірогідності правила досить знати підтримку самого набору і множини, що лежить в умові правила.

Наприклад, є набір {*A*, *B*, *C*}, що часто зустрічається, і потрібно підрахувати вірогідність для правила $AB \rightarrow C$. Підтримка самого набору відома, але і його множина {*A*, *B*}, що лежить в умові правила, також є такою, що часто зустрічається, в силу властивості антимонотонності, і значить його підтримка є відомою. Тоді легко можна підрахувати вірогідність. Це рятує нас від небажаного перегляду бази транзакцій, що був би потрібний у тому випадку, якби ця підтримка була невідомою.

Щоб витягти правило з набору *F*, що часто зустрічається, варто знайти всі його непорожні підмножини. І для кожної підмножини *s* ми зможемо сформулювати правило $s \rightarrow (F - s)$, якщо вірогідність правила $\text{conf}(s \rightarrow (F - s)) = \text{supp}(F)/\text{supp}(s)$ не менше порога minconf.

Зазначимо, що чисельник залишається постійним. Тоді вірогідність має мінімальне значення, якщо знаменник має максимальне значення, а це відбувається в тому випадку, коли в умові правила є набір, що складається з одного елемента. Усі супермножини даної множини мають меншу або рівну підтримку і, відповідно, більше значення вірогідності. Ця властивість може бути використана при витягу правил. Якщо ми почнемо витягати правила, розглядаючи спочатку тільки один елемент в умові правила, і це правило має необхідну підтримку, тоді всі правила, де в умові стоять супермножини цього елемента, також мають значення вірогідності вище заданого порога.

Наприклад, якщо правило $A \rightarrow BCDE$ задовольняє мінімальному порогові вірогідності minconf, тоді $AB \rightarrow CDE$ також задовольняє.

Для того, щоб витягти всі правила використовується рекурсивна процедура. Важливе зауваження: будь-яке правило, складене з набору, що часто зустрічається, повинне містити всі елементи набору. Наприклад, якщо набір складається з елементів {*A*, *B*, *C*}, то правило $A \rightarrow B$ не повинне розглядатися.

2.4 Програмні засоби для подання й обробки знань

Бібліотека SNToolbox (Semantic Network Toolbox) розроблена автором і містить функції для роботи із семантичними мережами у пакеті MATLAB.

Для роботи із бібліотекою на ЕОМ треба встановити пакет MATLAB (можна встановити тільки ядро пакету) та в окрему директорію на диску (бажано назвати її SNToolbox) записати файли бібліотеки SNToolbox. У середовищі пакету MATLAB треба вказати шлях до директорії з функціями бібліотеки SNToolbox (це можна зробити програмно за допомогою команди addpath 'шлях', або за допомогою опції Set path підменю File головного меню пакету MATLAB).

Процес створення та використання семантичних мереж у пакеті MATLAB при використанні бібліотеки SNToolbox полягає у послідовному виконанні таких етапів:

Етап 1. Побудова семантичної мережі.

1.1. Створення структури семантичної мережі та занесення її до змінної середовища пакету MATLAB.

1.2. Додавання до структури семантичної мережі вузлів.

1.3. Додавання до структури семантичної мережі відношень.

1.4. (Не обов'язково) Збереження семантичної мережі зі змінної середовища MATLAB у файл на диску.

Етап 2. Візуалізація семантичної мережі.

2.1. Завантаження семантичної мережі з диску у змінну середовища MATLAB, якщо вона відсутня у ньому.

2.2. Виклик функції візуалізації семантичної мережі.

2.3. (Не обов'язково) Збереження у файлі на диску (або виведення до друку на принтері) побудованої схеми семантичної мережі. Цей етап може бути виконаний як програмно – за допомогою відповідних функцій пакету MATLAB, так і за допомогою графічного інтерфейсу користувача із використанням команд меню фігури, на якій зображено схему семантичної мережі.

Етап 3. Пошук у семантичній мережі.

3.1. Для семантичної мережі – бази знань сформувати семантичну мережу-запит у окремій змінній середовища MATLAB. Для цього потрібно для мережі-запиту виконати підетапи 1.1–1.4 або скопіювати мережу бази знань до змінної мережі-запиту та видалити з неї зайві вузли і зв'язки та додати вузол мети із відповідними зв'язками.

3.2. Викликати функцію пошуку, яка поверне результати пошуку.

Розглянемо основні *функції бібліотеки*.

Функція `SN=Snnew` створює нову структуру для семантичної мережі у змінній SN. До *структури семантичної мережі* змінної SN будуть входити такі поля: SN.node – клітковий (cell) одновимірний масив, що містить назви вузлів мережі, які є рядками; SN.relation – клітковий (cell) двовимірний масив, що містить назви відношень між вузлами мережі, які є рядками (рядки масиву відповідають вузлам від яких, а стовпці – вузлам до яких направлені відношення); SN.nodetype – звичайний одновимірний масив, що містить коди типів вузлів мережі (0 – AND-вузол, 1 – OR-вузол);

Вузли семантичної мережі у бібліотеці поділяються на два типи: AND-вузли та OR-вузли.

AND-вузли (ТА-вузли), як правило, повинні зіставлятися поняттям, об'єктам, назвам атрибутів. У процесі виведення AND-вузли вважаються істинними (спрацювавшими, конкретизованими) у одному з таких випадків: 1) якщо вони є як у мережі бази знань, так і у мережі-запиті; 2) якщо всі вузли, що є їхніми батьками у ієрархії (тобто мають зв'язки направлені до них) є істинними.

OR-вузли (АБО-вузли), як правило, повинні зіставлятися значенням атрибутів. У процесі виведення OR-вузли вважаються істинними (спрацювавшими, конкретизованими) у одному з таких випадків: 1) якщо вони є як у мережі бази знань, так і у мережі-запиті; 2) якщо існує більше одного батьківського вузла та хоча б один з них є істинним. Якщо OR-вузол має лише один батьківський вузол, то він не буде наслідувати його істинність.

Функція `Res=SNaddANDnode(SN,'node1','node2',...,'nodeN')` додає вузли типу «AND» 'node1', 'node2', ..., 'nodeN' до семантичної мережі змінної SN (змінна SN при цьому не модифікується). Результат повертає у змінній Res.

Функція `Res=SNaddORnode(SN,'node1','node2',...,'nodeN')` додає вузли типу «OR» 'node1', 'node2', ..., 'nodeN' до семантичної мережі змінної SN (змінна SN при цьому не модифікується). Результат повертає у змінній Res.

Функція `Res=SNdelnode(SN,'node1','node2', ...,'nodeN')` видаляє вузли 'node1', 'node2', ..., 'nodeN' із семантичної мережі змінної SN (змінна SN при цьому не модифікується). Результат повертає у змінній Res.

Функція `Res=SNaddrelation(SN,'node1','relation','node2')` додає відношення 'relation' від вузла 'node1' до вузла 'node2' семантичної мережі змінної SN (змінна SN при цьому не модифікується). Результат повертає у змінній Res.

Функція `Res=SNdelrelation(SN,'node1','relation','node2')` видаляє відношення 'relation' від вузла 'node1' до вузла 'node2' семантичної мережі змінної SN (змінна SN при цьому не модифікується). Результат повертає у змінній Res.

Функція `SNplot(SN,type,NodeColour,RelationColour)` будує графічне зображення структури семантичної мережі змінної SN. Параметр type має бути рядком та задає тип розташування вузлів семантичної мережі на графіку: 'circle' – вузли розташовуються по кругу, 'random' – випадкове розташування вузлів, 'hierarchy' – ієрархічне розташування вузлів. Параметри NodeColour та RelationColour є необов'язковими. NodeColour має бути рядком та визначає колір вузлів мережі. RelationColour має бути рядком та визначає колір зв'язків мережі. Кольори мають бути задані символами подібно до стандартної функції plot: 'b' – blue (синій), 'g' – green (зелений), r – red (червоний), 'c' – cyan (світло-голубий), 'm' – magenta (бузковий), 'y' – yellow (жовтий), 'k' – black (чорний).

Функції Sncircleplot, Snhierarchyplot та Snrandomplot є внутрішніми і використовуються функцією Snplot для побудови графіків із круговим, ієрархічним та випадковим розташуванням вузлів, відповідно.

Функція `Res=SNfind(SN,SN1)` шукає цільовий вузол семантичної мережі змінної SN1 у семантичній мережі змінної SN. Мережа змінної SN1 має бути підмережею мережі змінної SN. Цільовий вузол у змінній SN1 має мати назву '?'. Функція повертає результат у змінній Res. Результатом є значення [], якщо цільовий вузол не був знайдений, або ім'я цільового вузла, якщо він був знайдений.

Функція `h=SNhierarchy(SN)` повертає у змінній *h* масив з номерами рівнів ієрархії вузлів семантичної мережі змінної SN. Номери рівнів ієрархії вузлів є цілими числами у діапазоні [1, *K*], де *K* – кількість встановлених рівнів ієрархії. Чим вище знаходиться вузол в ієрархії тим менший рівень ієрархії він має. Вузли одного рівня мають однакові номери рівня ієрархії.

Функція `k=isnodepresent(S,'node')` повертає у змінну *k* номер вузла з ім'ям 'node' у одновимірному клітковому масиві рядків *S*. Якщо вузол 'node' у одновимірному клітковому масиві рядків *S* відсутній, тоді функція повертає 0.

Функції бібліотеки висувають певні вимоги до семантичної мережі: кожен вузол повинний мати унікальне ім'я; мережа повинна мати вузли із вказаним типом (AND або OR); може бути тільки один цільовий вузол для пошуку; забезпечується тільки пряме виведення – від передумов до висновків; функції Snfind, SNhierarchy, а також функція Snplot у режимі 'hierarchy' працюють виключно з ієрархічними (шаруватими) мережами (тобто для коректного використання цих функцій не можна задавати повнозв'язні мережі, мережі де є латеральні зв'язки (зв'язки між вузлами одного шару) та мережі з елементами, які пов'язані самі з собою).

Поряд із функціями для роботи з семантичними мережами бібліотека містить графічний засіб для взаємодії з користувачем, що викликається командою `SNTool` (рис. 2.5).

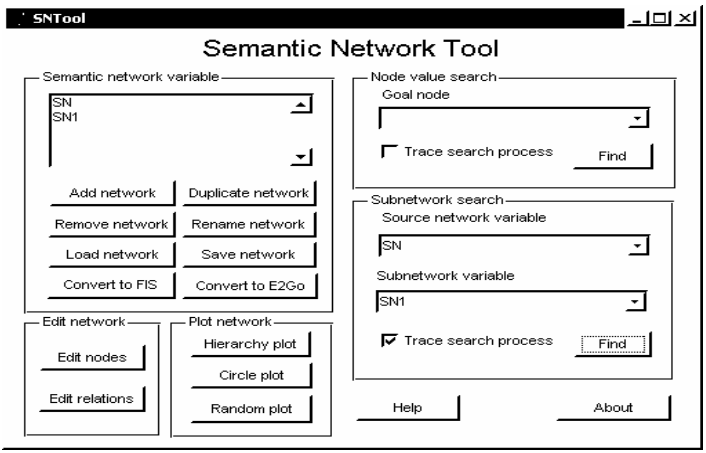


Рисунок 2.5 – Діалогова форма SNTool

Графічна панель SNTool містить такі компоненти: Sementic network variable – область вибору змінної семантичної мережі, Help – кнопка виклику допомоги, About – кнопка виклику інформації про автора, Add network – кнопка створення нової семантичної мережі, Remove network – кнопка видалення семантичної мережі, Load network – кнопка завантаження семантичної мережі з диску, Convert to FIS – кнопка перетворення семантичної мережі в програму для системи нечіткого виведення модуля Fuzzy Logic Toolbox пакета Matlab, Convert to E2Go – кнопка кнопка перетворення семантичної мережі в базу знань в форматі E2glite, Save network – кнопка збереження семантичної мережі у файлі на диску, Rename network – кнопка перейменування змінної семантичної мережі, Duplicate network – кнопка копіювання змінної семантичної мережі, Edit nodes – кнопка редагування вузлів мережі, Edit relations – кнопка редагування відношень мережі, кнопки Circle plot, Random plot та Hierarchy plot викликають відповідні режими функції SNplot для поточної змынної мережі, Subnetwork search – область пошуку підмережі (запиту) у семантичній мережі (базі знань), Source network variable – список вибору змінної бази знань, Subnetwork variable – список вибору змінної мережі-запиту, Trace search process – прапор трасування процесу пошуку, Find – кнопка запуску пошуку, Node value search – область пошуку значення цільового вузла у поточній мережі (базі знань), Goal node – список вибору цільового вузла.

E2gLite – це Java-аплет, що реалізує оболонку експертної системи, яка включається у веб-сторінку та завантажується із веб-сервера браузером користувача. Аплет завантажує базу знань із сервера, після чого запускається у браузері. Програмне забезпечення e2gLite постачається в архіві e2glite.jar.

Аплет e2gLite підключають до html-сторінки за допомогою конструкції:

```
<APPLET CODE=«e2glite.e2g.class»
ARCHIVE=«e2glite.jar» WIDTH=450 HEIGHT=300 MAYSCRIPT>
<PARAM NAME=«KBURL» VALUE=«knowledgebasefile.kb»>
(опційні <PARAM> теги)
.........
</APPLET>
```

У цій конструкції APPLET CODE містить ім’я класу, а ARCHIVE – назву файлу з архівом аплету. Назви класу та файлу є чутливими до регістру та мають бути наведені у нижньому регістрі. Якщо файл e2glite.jar розміщено у іншій директорії ніж html-документ, то параметр ARCHIVE повинен містити шлях до файлу. Параметр MAYSCRIPT включається тільки, коли e2gLite буде викликати функцію Java-script (Див. JSFUNCTION нижче).

Параметри аплету e2gLite:

KBURL – ім’я бази знань із шляхом до директорії, де знаходиться файл бази знань (або URL бази знань). Приклади використання: `<PARAM NAME=«KBURL» VALUE=«auto.kb»>` – база знань у тій же директорії, що й html-документ; `<PARAM NAME=«KBURL» VALUE=«http://www.site.com/auto.kb»>` – URL адреса файлу бази знань;

APPTITLE – будь-який рядок символів – перший рядок заголовку на початковому екрані. Значення за замовчанням: Blank. Приклад використання: <PARAM NAME=«APPTITLE» VALUE=«Auto Diagnosis»>;

APPSUBTITLE – будь-який рядок символів – другий рядок заголовку на початковому екрані. Значення за замовчанням: Blank. Приклад використання: <PARAM NAME=«APPSUBTITLE» VALUE=«by Ace Auto Repair»>;

BGCOLOR – Колір фону аплету. Задається як шістнадцятирічний колір RGB, що використовується у html-документах. Значення за замовчанням: #FFEE33 (блідо-жовтий). Приклад використання: <PARAM NAME=«BGCOLOR» VALUE=«#00FF00»> – зелений;

TITLECOLOR – колір тексту APPTITLE та APPSUBTITLE, визначений як шістнадцятирічний RGB колір. Значення за замовчанням: #000000 (чорний). Приклад використання: <PARAM NAME=«TITLECOLOR» VALUE=«#FFFFFF»> – білий;

PROMPTCOLOR – колір тексту діалогів, визначений як шістнадцятирічний RGB колір. Значення за замовчанням: #000000 (black). Приклад використання: <PARAM NAME = «PROMPTCOLOR» VALUE = «#FFFFFF»> – білий;

DEBUG – якщо має значення TRUE, то вікно відлагоджувача буде створюватися коли база знань стартує. Значення за замовчанням: FALSE. Приклад використання: <PARAM NAME=«DEBUG» VALUE=«TRUE»>;

STARTBUTTON – текст на кнопці, що починає консультацію. Значення за замовчанням: «Start the consultation». Приклад використання: <PARAM NAME=«STARTBUTTON» VALUE=«Help me!»>;

LOADMSG – повідомлення, яке відображується на екрані, що відкривається, коли завантажується база знань. Значення за замовчанням: «Loading knowledge base...». Приклад використання: <PARAM NAME = «LOADMSG» VALUE=«Chargeant la base de connaissance...»>;

PROMPTFIRST – якщо має значення TRUE, тоді діалог використовується для атрибута перед зворотнім виведенням. Значення за замовчанням: TRUE. Приклад використання: <PARAM NAME = «PROMPTFIRST» VALUE=«false»>;

JSFUNCTION – ім’я функції Java-скрипта, що викликається кожний раз, коли кнопка e2gLite натискається. Значення за замовчанням: null (no function is called). Приклад використання: <PARAM NAME=«JSFUNCTION» VALUE=«buttonPush»>. Коли використовується параметр JSFUNCTION, MAYSCRIPT повинен включатися у тег <applet>.

Імена параметрів аплету є чутливими до регістру та завжди мають бути визначені у верхньому регістрі в e2gLite. Значення параметрів аплету e2gLite є нечутливими до регістру. Єдиний параметр аплету, що обов’язково має бути визначений – KBURL. Якщо значення KBURL не починається

з http://, то значення буде додано до директорії, у якій знаходиться html-документ, що завантажує аплет e2glite.

База знань має бути завантажена з того ж веб-сайту, що й архів аплету e2glite.jar. Це вимога безпеки Java-аплетів.

Коди кнопок e2gLite. Натиснутій кнопці «Start the consultation» відповідає значення – 1 (ця кнопка може бути перейменована через параметр STARTBUTTON; інші кнопки можуть бути перейменовані командами бази знань TRANSLATE), кнопці «Submit your response» – 2, кнопці «Why Ask?» – 3, кнопці «Return (from Why Ask?)» – 4, кнопці «Explain» – 5, кнопці «Restart» – 6.

Методи e2gLite, що доступні з Java-скрипту

Ім'я методу	Значення, що повертається, та його тип	Аргументи та їхні типи
getAttrCF(attrName, valIx)	(Float) Коефіцієнт упевненості, асоційований з певним прикладом поіменованого атрибута (0.0, якщо не існує або ще не визначено)	(String) attrName: ім'я атрибута (Integer) valIx: приклад (між 1 та максимальним значенням) поіменованого атрибута для дозволення багатозначних атрибутів
getAttrCount( )	(Integer) Кількість атрибутів, що визначені у базі знань	
getAttrIx(attrName,valIx)	(Integer) Індекс атрибута, асоційованого з певним прикладом поіменованого атрибута (0, якщо не існує)	(String) attrName: ім'я атрибута (Integer) valIx: приклад (між 1 та максимальним значенням) поіменованого атрибута для дозволення багатозначних атрибутів
getAttrName(attrIx)	(String) Ім'я атрибута (якщо немає – рядок нульової довжини)	(Integer) attrIx: індекс атрибута
getAttrValue(attrIx)	(String) Строкове подання поточного значення (рядок нульової довжини, якщо не існує або ще не визначено)	(Integer) attrIx: індекс атрибута
getAttrValue(attrName, valIx)	(String) Строкове подання поточного значення (рядок нульової довжини, якщо не існує або ще не визначено)	(String) attrName: ім'я атрибута (Integer) valIx: приклад (між 1 та максимальним значенням) поіменованого атрибута для дозволення багатозначних атрибутів
getMaxValues(attrName)	(Integer) Кількість значень (прикладів) дозволених для поіменованого атрибута (0, якщо не існує)	(String) attrName: ім'я атрибута
getGoalAttr( )	(Integer) індекс атрибута поточної цілі або підцілі (0, якщо виведення завершено)	

Атрибути e2gLite позначаються ім'ям та номером індексу, що починається з 1. Кожне значення атрибута має унікальний індекс, але усі значення

багатозначних атрибутів зберігаються під одним і тим же ім'ям. Значення атрибута не буде повернено доти, поки значення не стане відомим з коефіцієнтом упевненості що найменше рівним мінімальному CF.

База знань e2gLite може містити такі команди.

REM – рядок коментарю. Установки коментарів REM – це однорядкові коментарі, що ігноруються, коли обробляється база знань. Команда REM повинна бути на початку рядка, а всі інші символи в тому ж рядку ігноруються.

Порожні рядки є необов'язковими й ігноруються. Вони можуть використовуватися для поліпшення зручності читання бази знань.

RULE – починає правило, яке слідує за описом правила у квадратних дужках. If – починає передумову правила: за ним слідує перша передумова правила (ім'я атрибута в квадратних дужках, реляційний оператор і порівнюване значення), необов'язковий логічний оператор and/or (якщо існує інша передумова правила). Then – починає наслідок правила, за ним слідує перше речення наслідку (назва атрибута в квадратних дужках, знак «=», значення коефіцієнта упевненості (необов'язково): @xxx, де xxx подає відсоток і за замовчуванням дорівнює 100), необов'язковий AND (потрібно, якщо мається інше речення наслідку). Всі правила RULE мають бути визначені перед використанням будь-якої команди, за виключенням REM.

PROMPT – визначає формат запиту користувача про значення атрибута, коли воно необхідне машині виведення. Типи діалогів: YesNo (булеве введення), MultChoice (багатозначний вибір текстових рядків), Choice (вибір текстових рядків у списку, що випадає в низ), AllChoice (багатозначний вибір текстових рядків), Numeric (введення із перевіркою діапазону). Необов'язковий параметр CF указує на необхідність запиту коефіцієнта упевненості).

Питання, що задається користувачу, знаходиться на наступному рядку в одинарних чи подвійних лапках. Варіанти відповідей (якщо потрібно діалогу) знаходяться на послідовних рядках у лапках. Для числового діалогу мінімальні і максимальні припустимі значення знаходяться на послідовних рядках у лапках.

GOAL – знаходження значення (чи значень) для цільових змінних – ціль консультації експертної системи. Коли всі цілі вирішено (або визначено, що вони не можуть бути вирішені) консультація завершується. Потрібною є принаймні одна ціль. Кожна ціль задається в окремому рядку, що починається з GOAL, супроводжуваного назвою цільового атрибута у квадратних дужках. Цілі можуть бути визначені перед або після діалогів, але вони повинні слідувати за визначеннями правил. У базі знань повинна міститися принаймні одна установка цілі: цілі – це атрибути, для яких машина виведення шукає значення. Процес виведення закінчується, коли всі цілі були задоволені або є нерозв'язними.

MINCF – мінімальне значення коефіцієнта впевненості для прийняття значення як факту. За замовчуванням: 80.0.

MAXVALS – (необов'язково) максимальна кількість різних значень, які може мати одна ознака. За замовчуванням – 1. Атрибути з діалогами типу AllChoice повинні мати MAXVALS більше ніж 1.

DEFAULT – (необов'язково) установлює значення атрибута за замовчуванням, якщо машина виведення не може одержати його значення з діалогів або від правил.

TRANSLATE – (необов'язково) налагоджує виведення тексту. Також підтримує переклад бази знань на мови, відмінні від англійської мови.

У тексті бази знань ім'я атрибутів мають бути взяті у квадратні дужки [], текстові рядки, що подають значення атрибутів мають бути взяті у одинарні або подвійні лапки. Команди, ім'я атрибутів та текстові значення є нечутливими до регістру. Проміжки всередині квадратних дужок або лапок є значущими. Проміжки між іншими елементами є незначущими. Речення з багатьма передумовами правил можуть бути пов'язані за допомогою And або Or, але не одночасно в одному й тому ж правилі.

Установки правил RULE. Можливі елементи кожного правила включають:

– перший рядок починається з RULE, що супроводжується коротким ідентифікуючим описом правила, взятим у квадратні дужки;

– наступний рядок починає передумову правила з IF і супроводжується логічним виразом, що складається з: назви атрибута, взятої в квадратні дужки; реляційного оператора: = (дорівнює), < (менше ніж), > (більше ніж), ! (не дорівнює), : (дорівнює кожному з); порівнюваного значення: число, булеві значення TRUE чи FALSE або рядок, взятий в одинарні чи подвійні лапки;

– (необов'язково) один з логічних операторів AND або OR, якщо в передумові правила є кілька логічних виразів. Якщо є більш ніж один логічний вираз передумови, вони повинні усі зв'язуватися тим самим логічним оператором. Кожен пункт передумови, що складається з назви атрибута, реляційного оператора і порівнюваного значення, вводиться в окремому рядку Заключний пункт передумови не повинний закінчуватися AND чи OR;

– наступний рядок після логічного виразу передумови, що не закінчується AND чи OR, починає наслідок правила з THEN, супроводжуваний присвоюванням, що складається з: назви атрибута, взятої у квадратні дужки; оператора присвоювання (=); значення, що буде присвоєне ознаці (число, булеві значення TRUE чи FALSE, або рядок, укладений в одинарні чи подвійні лапки);

– (необов'язково) якщо значення, зіставлене атрибута з менш ніж 100% вірогідністю, воно супроводжується @ і числовим значенням коефіцієнта упевненості від 1 до 100%;

– (необов'язково) якщо є ще одна установка значення в наслідку правила, завершуйте рядок за допомогою AND. Рядок у наслідку правила, що не закінчується за допомогою AND, завершує правило.

Елементи правила є не чуттєвими до регістра символів. Кожен елемент правила повинний бути на окремому рядку.

Атрибути в e2gLite мають один із трьох типів: строковий (текстовий), числовий і булевий. Тип атрибута встановлюється в перший раз, коли він з'являється в передумові правила або в наслідку правила при установці значення. Будь-яке наступне використання атрибута з іншим типом приводить до помилки. Використання атрибута, що попередньо не був визначений у правилі, у будь-якій іншій установці бази знань типу PROMPT чи GOAL також є помилкою.

Коли правила подаються користувачу експертної системи як частина механізму пояснення, вони форматуються для зручності читання. Квадратні дужки і лапки опускаються, а реляційні оператори перетворюються в текст. Наприклад, «=« переводиться в «is», а передумова правила: «If [precipitation] = «expected»« переводиться в: «If precipitation is expected». Ретельний вибір назв і значень ознак дозволяє створювати правила, зрозумілі для користувача.

Оператори відношень e2gLite

Оператор	Інтерпретація	Форматований переклад правила	Приклад неформатованого / форматованого запису передумови речення
=	дорівнює	є	[the result of trying the starter] = «nothing happens» the result of trying the starter is nothing happens
<	строго менше ніж	менше ніж	[the age of the battery in months] < 24 the age of the battery in months is less than 24
>	строго більше ніж	більше ніж	[the age of the battery in months] > 12 the age of the battery in months is greater than 12
!	не дорівнює	не є	[the gas tank] ! «empty» the gas tank is not empty
:	дорівнює будь-якому з (використовується тільки з текстовими даними)	є *значення1*, *значення2*,... або *значенняN*	[the battery brand] : «Delco» «Mopar» «Interstate» the battery brand is Delco, Mopar or Interstate

По обидва боки оператор відношення може бути оточений одним і більш необов'язковими пробілами.

Перекладене значення оператору відношення має братися до уваги, коли створюються ім'я та значення атрибутів, щоб покращити читабельність правил.

За винятком «:» (дорівнює кожному з), цінність багатозначного атрибута з найвищим CF використовується для обчислення виразу. Вираз із оператором «дорівнює кожному з» розглядається як істинний, якщо будь-яке значення атрибута з коефіцієнтом упевненості дорівнює або більше ніж MINCF відповідає будь-якому значенню в списку кількостей порівняння.

Визначення діалогів: діалоги (PROMPT) повинні бути включені в базу знань після того, як визначені всі правила. Можливі елементи кожного діалогу включають:

– перший рядок починається з PROMPT, супроводжуваного назвою атрибута, укладеною у квадратні дужки, і типом діалогу. Типи діалогу: MultChoice (багаторазовий вибір), Choice (список вибору, що випадає вниз), AllChoice (дозволяє вибирати кілька відповідей одночасно), YesNo (булеве уведення) і Numeric (уведення числового значення);

– (необов'язково) закінчення першого рядка PROMPT з CF дозволяє користувачу експертної системи вводити коефіцієнт упевненості, коли відображається діалог. Якщо CF не визначений, уведення користувача буде прийнято з вірогідністю 100 %;

– другий рядок містить питання діалогу, взяті в одинарні або подвійні лапки. Це питання буде задано користувачу експертної системи, якщо діалог буде викликаний машиною виведення;

– для діалогів типів MultChoice, Choice та AllChoice, альтернативні значення, що будуть пред'явлені користувачу експертної системи, записуються по одному в послідовних рядках, укладені в одинарні або подвійні лапки. Для діалогів числового типу на двох наступних рядках записуються мінімальні і максимальні прийнятні значення, взяті в одинарні або подвійні лапки.

Команда `TRANSLATE об'єкт=«текст»` дозволяє розробнику бази знань змінювати текст кнопок та повідомлень. Наведемо перелік параметрів команди TRANSLATE та їх значення за замовчанням.

Тексти кнопок: B_SUBMIT=«Submit your response», B_EXPLAIN= «Explain», B_WHYASK=«Why ask?», B_RESTART=«Restart», B_RETURN= «Return».

Тексти повідомлень: TR_KB = «Knowledge base:», TR_NORESP = «I don't know/would rather not answer», TR_HOWCONF = «How confident do you feel about your response?», TR_LOWCONF = «Very uncertain (50%)», TR_HICONF = »Very certain (100%)», TR_YES = «yes», TR_NO = «no», TR_FALSE = «false», TR_RESULTS = «FINAL RESULTS:», TR_MINCF = »Minimum confidence factor for accepting a value as a fact:», TR_NOTDETERMINED = «could not be determined», TR_ISRESULT = «is:», TR_WITH = «with», TR_CONF = «% confidence», TR_ALLGOALS = «all conclusions», TR_VALUE = «Value», TR_OF = »of», TR_THISRULE = «Rule below fired at CF=«, TR_RULEASGN = »and assigned the value», TR_TOFIND = »To find», TR_AVALUE = «a value for», TR_ISNEEDED = «is needed to try this rule:», TR_RULE = «RULE:», TR_IF = «IF:», TR_THEN = »THEN:», TR_AND = «and», TR_OR = «or», TR_EQUAL = «is», TR_LESSTHAN = «is less than», TR_GREATER = «is greater than», TR_NOTEQUAL = «is not», TR_VALUEFOR = «A value for:», TR_FOUND = »has been determined», TR_NOTFOUND = «has not yet been determined», TR_WASINPUT = «was input with «, TR_DETERMINED = »Determined», TR_IS = «is», TR_FROM = «from:», TR_DEFAULTED = «was set by default with».

Коди помилок e2gLite: 150 – помилка ініціалізації, 205 – зациклення процесу виведення, 210 – забагато установок GOAL, 220 – немає установки GOAL, 230 – забагато атрибутів, 240 – забагато невідомих атрибутів, 250 – забагато правил, 260 – забагато допустимих речень RULE, 270 – забагато послідовних речень RULE, 280 – немає місця для даних правил, 290 – забагато діалогів, 300 – немає місця для даних діалогів, 400 – очікується розмежоване ім'я або значення атрибута на цьому місці, 410 – немає кінця бази знань, 420 – очікується установка IF на цьому місці, 430 – очікується ім'я атрибута на цьому місці, 440 – очікується значення на цьому місці, 450 – очікується установка THEN на цьому місці, 460 – очікується реляційний оператор (= < > ! :) на цьому місці, 470 – очікується логічний оператор (and / or) на цьому місці, 480 – очікується оператор (=) на цьому місці, 490 – очікується AND на цьому місці, 500 – очікується коефіцієнт упевненості між 1 та 100 на цьому місці, 510 – очікується числове значення MAXVALS на цьому місці, 520 – очікується тип діалогу (YesNo, MultChoice, ...) на цьому місці, 530 – перекладне значення з цим ім'ям не існує, 540 – атрибут з цим ім'ям не існує, 600 – спробуйте змінити тип атрибута, 610 – недопустиме числове значення, 620 – погане числове значення в установці MAXVALS, 630 – логічний оператор не розпізнаний, 700 – тип діалогу не розпізнаний, 750 – команда бази знань не розпізнана, 760 – нерозпізнаний тип даних, 800 – відсутня або порожня база знань.

Більшість нумерованих помилок знаходиться, коли база знань завантажена. Завантаження припиняється, коли перша помилка знаходиться та рядок, який містить помилку відображується у вікні відлагоджувача (DEBUG), якщо запуск здійснено у режимі відлагоджувача.

Виведення результатів для бази знань містить такі установки: Starting new consultation with the following goals – список усіх ознак, названих в установках GOAL; Minimum confidence factor for accepting a value as a fact – мінімальний коефіцієнт упевненості для прийняття значення як факт (коли починається нова консультація наводиться після списку цілей); Trying rule – опис для правила, яке машина виведення збирається розглядати; >Add to goal stack – містить назву атрибута, доданого до стеку цілі (значення цього атрибута є необхідним для обчислення правила, яке машина виведення розглядала останнім рухаючи до статусу Unknown); >Rule status after evaluation is – статус правила, яке машина виведення тільки що розглянула (можливі значення: Unknown (невідоме), True (істина)/fired (попадання), False (хибність) і Failed (невдача)); Prompt for – діалог буде викликаний, щоб одержати значення для названого атрибута; *Prompt assigned – значення, отримане від останнього викликаного діалогу, супроводжуване впевненістю, з якою значення було введено; *Rule assigned – правило знайдено та у наслідку названа ознака отримала певне значення з рівнем упевненості, показаним у круглих дужках; *Default assigned – значення, зазначене в команді бази знань DEFAULT, було призначено названій ознаці з рівнем упевненості, показаним в круглих дужках після того, як машина виведення не зуміла

визначити його значення; Forward chain – значення атрибута було отримано з діалогу, оскільки знайдене правило, або з установки за замовчуванням з достатньою вірогідністю, що дозволяє розглядати його як факт (воно буде тепер використовуватися для обчислення придатних правил у базі знань); >Forward chain – trying rule – містить назву правила, для якого буде пробуватися процес прямого виведення, після >Rule status after evaluation is: (статус правила після обчислення) буде випливати виведення трасування; No eligible rules for – значення названого атрибута було знайдене, але відсутні прийнятні правила для прямого пошуку; Remove from goal stack – причина для видалення атрибута зі стека мети, супроводжувана назвою атрибута (причини: (Resolved–Вирішено) зміст значення був визначений або (Failed–Невдача) значення атрибута не могло бути визначено); Replace on goal stack – коли значення знайдене для багатозначного атрибута (MAXVALS > 1), він видаляється зі стека мети, якщо існують які-небудь збережені невирішені значення, ознака потрапляє назад у стек мети; Goal stack empty: end consultation – стек мети порожній, так що консультація закінчена: кожна змінна мети була вирішена або машина виведення вирішила, що вона не може бути вирішена з доступними фактами.

Трасування бази знань доступне, коли запуск у режимі відлагодження здійснений установкою параметра DEBUG у TRUE в параметрах <APPLET>. Кнопка «Trace is ON/OFF» вмикає / вимикає виведення результатів трасування. Ця кнопка може використовуватися в будь-який час протягом консультації.

Аналіз бази знань містить такі компоненти.

ATTRIBUTE USAGE: Усі назви ознак, знайдені в базі знань заносяться в список за абеткою (ігнорується регістр символів). Діалоги і правила, що можуть визначати значення атрибута, показуються. Потім у списку в окремих рядках показуються правила, що використовують ознаку. Ознака, що використовується, але ніколи не визначається, представляє логічну помилку та позначається знаком питання перед назвою атрибута. Це, як правило, відбувається коли назва атрибута була написана з орфографічною помилкою.

VALUE USAGE: У секції звіту VALUE USAGE, усі текстові значення, знайдені в базі знань, перелічуються за абеткою (ігноруючи регістр символів). Кожне значення супроводжується його довжиною, укладеною в круглі дужки. Потім його місце розташування в базі знань (у передумові правила (IF), реченні правила (THEN) чи діалозі) ідентифікується поряд зі зв'язаною ознакою. Це виведення допомагає ідентифікувати значення із записом, що слабко відрізняється, інтервалом або іншою пунктуацією, яка робить два значення, що повинні бути відповідними, невідповідними для машини виведення. Довжина кожного атрибута корисна для ідентифікації типографських помилок типу двох пробілів між словами, коли повинний був бути тільки один пробіл.

Аналізи бази знань доступні, коли запуск у режимі відлагоджувача отриманий встановленням параметру DEBUG у TRUE в параметрах <APPLET>. Для отримання цього виводу натисніть кнопку «Analyze KB» у вікні «KNOWLEDGE

BASE DEVELOPER'S OUTPUT». Аналіз представляє статичну інформацію про базу знань та може бути викликаний у будь-який час впродовж консультації.

Дамп бази знань містить такі компоненти: Minimum CF – мінімальний коефіцієнт упевненості, що повинний бути досягнутий перед тим, як значення атрибута буде прийняте як відоме; ATTRIBUTES – поточне значення атрибута, супроводжуване його типом (*S* для строкового, *N* для числового і *B* для булева) і вірогідністю в поточний момент (якщо база знань включає установку DEFAULT для цього атрибута, приймається значення за замовчуванням, тип і вірогідність, з якою воно зв'язано); RULES – перший рядок для кожного правила містить опис правила, поточний статус (*U* для невідомого, *T* для доведеного істинного, *F* для доведеного хибного та *X* для відмовлення (не може бути визначено)), супроводжуваний поточним коефіцієнтом упевненості, позв'язаним із правилом, кожна наступна передумова (IF:): і наслідок (THEN:) речення подані поряд з типом значення, послідовні речення також містять коефіцієнт упевненості, асоційований з установкою значення в круглих дужках; GOAL STACK – стек цілі містить список атрибутів, що машина виведення пробує вирішувати (остання (з найвищим номером) ціль – це поточна мета виведення; ціль видаляється, коли значення визначене або не існує способу визначити значення; коли стек цілі порожній, консультація завершується).

Дамп бази знань є доступним, коли запуск у режимі відлагоджувача отриманий встановленням параметра DEBUG у TRUE в параметрах <APPLET>. Для одержання дампа натисніть кнопку «Display KB dump» у вікні «KNOWLEDGE BASE DEVELOPER OUTPUT». Дампи можуть бути викликані в будь-який час впродовж консультації.

Якщо атрибуту дозволено мати множинні значення (існує установка MAXVALS для даного атрибута зі значенням більшим за 1), він може з'являтися багато разів у секції ATTRIBUTE дампа.

Бібліотека функцій ARMADA дозволяє створювати асоціативні правила за заданими користувачем даними та виконувати операції з асоціативними правилами у пакеті Matlab.

Для використання бібліотеки ARMADA необхідно додати до списку робочих шляхів пакету Matlab шлях до папки, де знаходяться функції бібліотеки, після чого ввести у командному рядку Matlab команду: `armada`

В результаті з'явиться головна інтерфейсна форма, в якій необхідно обрати файл з вхідними даними та параметри для аналізу даних (рис. 2.6).

Для виконання аналізу даних за допомогою асоціативних правил в полі FILE DETAILS необхідно ввести шлях до файлу з даними для аналізу. Файл з даними для аналізу можна також обрати у віконному режимі, натиснувши кнопку Browse. В полі зі списком Delimiting Character необхідно вказати символ, що відокремлює дані одне від одного. Компоненти головної форми

Minimum Confidence та Minimum Support призначені для введення значень мінімальної довіри та мінімальної підтримки, відповідно.

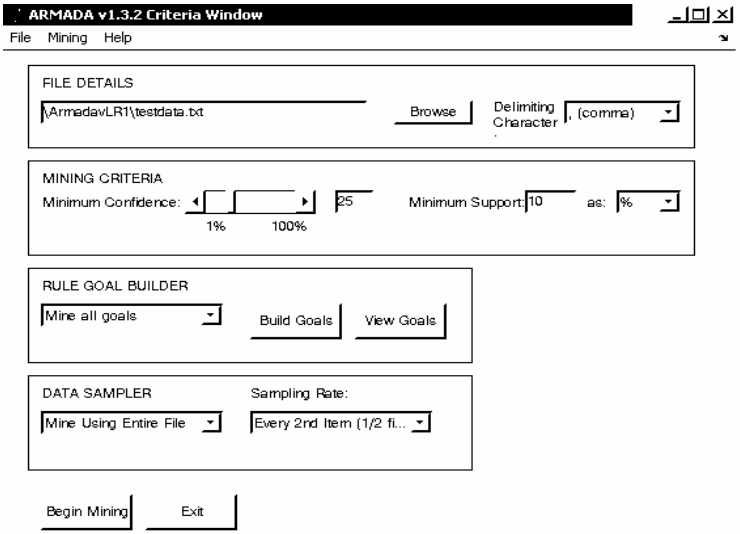


Рисунок 2.6 – Головна інтерфейсна форма ARMADA

Для виконання аналізу даних та отримання вихідної інформації у вигляді асоціативних правил необхідно натиснути кнопку Begin Mining, після чого відбудеться процес аналізу даних та з'явиться форма з результатами роботи програми (рис. 2.7). Вихід з програми виконується за допомогою кнопки Exit.

В області RULES наведено отримані асоціативні правила, в області File Items – елементи файлу для аналізу та кількість разів, з якою вони зустрічаються у файлі.

Компоненти в області MINING CRITERIA відображають параметри аналізу: значення параметрів minsupport та minconfidence, а також файл із даними для аналізу.

В області MINING REPORT відображається інформація про хід виконання аналізу: час, витрачений на виконання аналізу (Mining Time), розмір файлу (File Size), час, витрачений на один запис (Time per Entry), кількість отриманих асоціативних правил (No. Rules).

За допомогою головного меню отримані дані можна зберегти на диск (File –> Save) та отримати графіки, що відображають процес аналізу даних (Graphics).

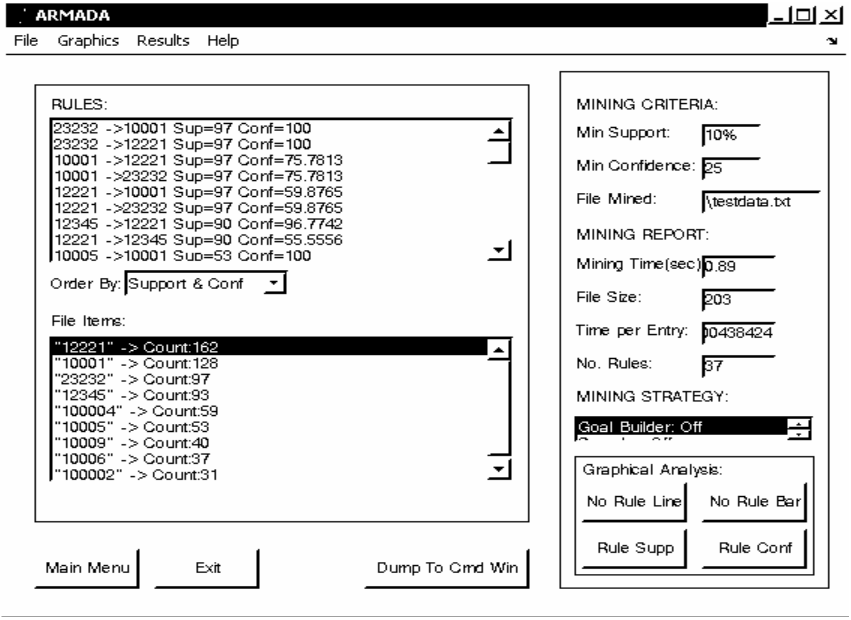


Рисунок 2.7 – Результати аналізу даних

Бібліотека Statistics Toolbox дозволяє будувати дерева вирішальних правил у пакеті Matlab. Основними функціями для роботи з деревами є: treefit, treeprune, treedisp, treetest, treeval.

Функція `t = treefit(X,y)` створює дерево вирішальних правил *t* на основі заданих значень незалежних змінних (матриця *X*) та значень вихідного параметру (вектор *y*).

Функція `t1 = treeprune(t,'level',n)` створює дерево вирішальних правил *t*1 на основі заданого дерева *t*, скороченого до *n*-го рівня.

Функція `t1 = treeprune(t,'nodes',nod)` створює дерево вирішальних правил *t*1 на основі заданого дерева *t*, видаляючи при цьому вузли, вказані в змінній nod.

Функція `treedisp(t)` відображає дерево *t* у графічному вигляді.

Функція `c = treetest(t,'test',X,y)` виконує тестування дерева *t* за допомогою тестової вибірки (*X*, *y*).

Функція `Ycalc = treeval(t,X)` за допомогою дерева *t* для масиву незалежних змінних *X* розраховує значення вихідного параметру Ycalc.

2.5 Приклади та ілюстрації

Приклад 1. Побудувати схему семантичної мережі для предметної області «Діагностика внутрішніх хвороб» (див. рис. 2.6).

Приклад 2. Побудувати семантичну мережу у пакеті MATLAB за допомогою бібліотеки SNToolbox. Зобразити побудовану мережу графічно. Створити мережу-запит. Виконати пошук у семантичній мережі на основі мережі-запиту.

Текст програми, що реалізує семантичну мережу із використанням функцій бібліотеки SNToolbox пакету MATLAB (Зауваження: в тексті програми коментарі наведено українською мовою для спрощення сприйняття. Використання кириличних літер у пакеті MATLAB може призводити до збоїв, тому коментарі бажано писати латинськими літерами).

```
SN=SNnew; % створюємо структуру для семантичної мережі
% додаємо вузли типу «АБО» - значення атрибутів
SN=SNaddORnode(SN,'high','normal','acute', 'weak','in stomach','in spine');
% додаємо вузли типу «ТА» - об'єкти і назви атрибутів
SN=SNaddANDnode(SN, 'patient', 'symptoms', 'temperature', 'pain type',
'pain', 'diagnosis', 'apendicitis', 'pielonefrit', 'acute apendicitis',
'acute pielonefrit', 'chronic apendicitis', 'chronic pielonefrit');
% додаємо відношення між вузлами
SN=SNaddrelation(SN, 'patient', 'has', 'symptoms');
SN=SNaddrelation(SN, 'patient', 'has', 'diagnosis');
SN=SNaddrelation(SN, 'symptoms', 'type', 'temperature');
SN=SNaddrelation(SN, 'symptoms', 'type', 'pain');
SN=SNaddrelation(SN, 'diagnosis', 'type', 'acute apendicitis');
SN=SNaddrelation(SN, 'diagnosis', 'type', 'acute pielonefrit');
SN=SNaddrelation(SN, 'diagnosis', 'type', 'chronic apendicitis');
SN=SNaddrelation(SN, 'diagnosis', 'type', 'chronic pielonefrit');
SN=SNaddrelation(SN, 'temperature', 'type', 'high');
SN=SNaddrelation(SN, 'temperature', 'type', 'normal');
SN=SNaddrelation(SN, 'pain type', 'type', 'acute');
SN=SNaddrelation(SN, 'pain type', 'type', 'weak');
SN=SNaddrelation(SN, 'pain', 'type', 'in stomach');
SN=SNaddrelation(SN, 'pain', 'type', 'in spine');
SN=SNaddrelation(SN, 'in stomach','include', 'apendicitis');
SN=SNaddrelation(SN, 'in spine', 'include', 'pielonefrit');
SN=SNaddrelation(SN, 'apendicitis', 'include', 'acute apendicitis');
SN=SNaddrelation(SN, 'apendicitis', 'include', 'chronic apendicitis');
SN=SNaddrelation(SN, 'symptoms', 'type', 'pain type');
SN=SNaddrelation(SN, 'pielonefrit', 'include', 'acute pielonefrit');
SN=SNaddrelation(SN, 'pielonefrit', 'include', 'chronic pielonefrit');
SN=SNaddrelation(SN, 'high', 'include', 'acute apendicitis');
SN=SNaddrelation(SN, 'normal', 'include', 'chronic apendicitis');
```

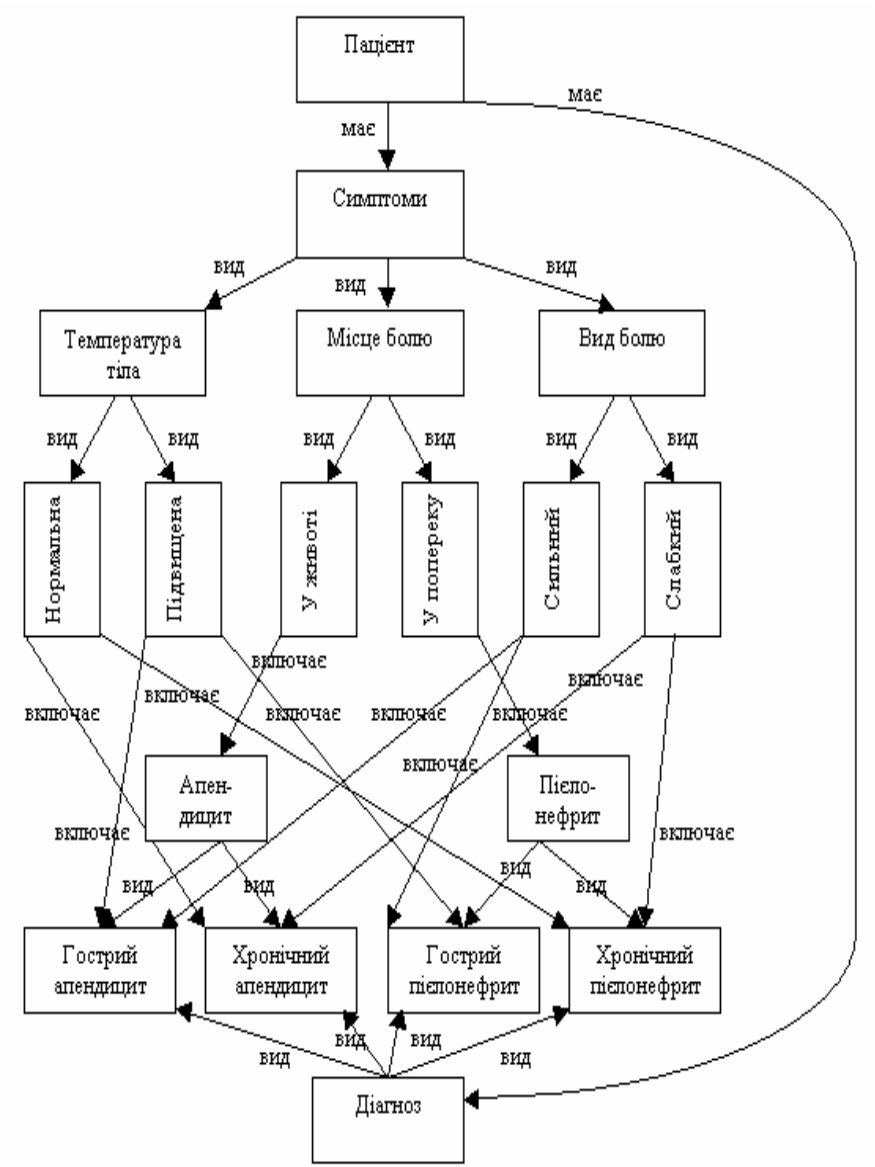


Рисунок 2.6 – Схема семантичної мережі

```
SN=SNaddrelation(SN, 'high' , 'include', 'acute pielonefrit');
SN=SNaddrelation(SN, 'normal', 'include', 'chronic pielonefrit');
SN=SNaddrelation(SN, 'acute', 'include', 'acute apendicitis');
SN=SNaddrelation(SN, 'weak', 'include', 'chronic apendicitis');
SN=SNaddrelation(SN, 'acute','include', 'acute pielonefrit');
SN=SNaddrelation(SN, 'weak', 'include', 'chronic pielonefrit');
% будуємо графічні зображення схеми семантичної мережі
SNplot(SN, 'hierarchy'); % ієрархічне розташування вузлів
figure; % створюємо нове вікно для іншої схеми
SNplot(SN, 'circle'); % кругове розташування вузлів
SN1=SN; % створюємо мережу-запит на основі мережі бази знань
SN1=SNdelnode(SN1,'normal','in    spine','weak','apendicitis','pielonefrit',
'acute apendicitis', 'acute pielonefrit', 'chronic apendicitis', 'chronic
pielonefrit'); % видаляємо з мережі-запиту зайві вузли
SN1=SNaddANDnode(SN1, '?'); % додаємо до мережі-запиту цільовий вузол
% додаємо до мережі-запиту відношення для цільового вузла
SN1=SNaddrelation(SN1, 'diagnosis', 'type', '?');
figure; % створюємо нове вікно для схеми
% будуємо графічне зображення схеми мережі-запиту
SNplot(SN1, 'random'); % випадкове розташування вузлів
% виконуємо запит до семантичної мережі, результати якого видаємо на екран
Res=SNfind(SN, SN1)
```

У результаті роботи програми отримаємо повідомлення у командному вікні пакету MATLAB:

```
RESULT: The number of founded solutions is 1.
Res = 'acute apendicitis'
```

Крім того отримаємо у вигляді окремих фігур зображення схеми семантичної мережі з ієрархічним розташуванням вузлів (рис. 2.7) та з круговим розташуванням вузлів (рис. 2.8), а також схему мережі запиту (рис. 2.9).

Приклад 4. Побудувати фреймову мережу для предметної області «Діагностика внутрішніх хвороб» (див. рис. 2.10).

Приклад 5. Побудувати логічну модель для діагностики хвороб.

Нехай симптоми позначаються як: A_1 – підвищена температура тіла, A_2 – нормальна температура тіла, B_1 – біль у животі, B_2 – біль у попереку, C_1 – слабкий біль, C_2 – сильний біль. Діагнози позначимо як: D_1 – гострий апендицит, D_2 – хронічний апендицит, D_3 – гострий пієлонефрит, D_4 – хронічний пієлонефрит.

Тоді можна визначити формули для встановлення кожного діагнозу:

$$D_1 = A_1 \wedge \neg A_2 \wedge B_1 \wedge \neg B_2 \wedge C_1 \wedge \neg C_2,$$
$$D_2 = \neg A_1 \wedge A_2 \wedge B_1 \wedge \neg B_2 \wedge \neg C_1 \wedge C_2,$$
$$D_3 = A_1 \wedge \neg A_2 \wedge \neg B_1 \wedge B_2 \wedge C_1 \wedge \neg C_2,$$
$$D_4 = \neg A_1 \wedge A_2 \wedge \neg B_1 \wedge B_2 \wedge \neg C_1 \wedge C_2.$$

Враховуючи, що для даної задачі: $A_1 = \neg A_2$, $B_1 = \neg B_2$, $C_1 = \neg C_2$, ці формули можна спростити:

$$D_1 = A_1 \wedge B_1 \wedge C_1,\ D_2 = A_2 \wedge B_1 \wedge C_2,$$

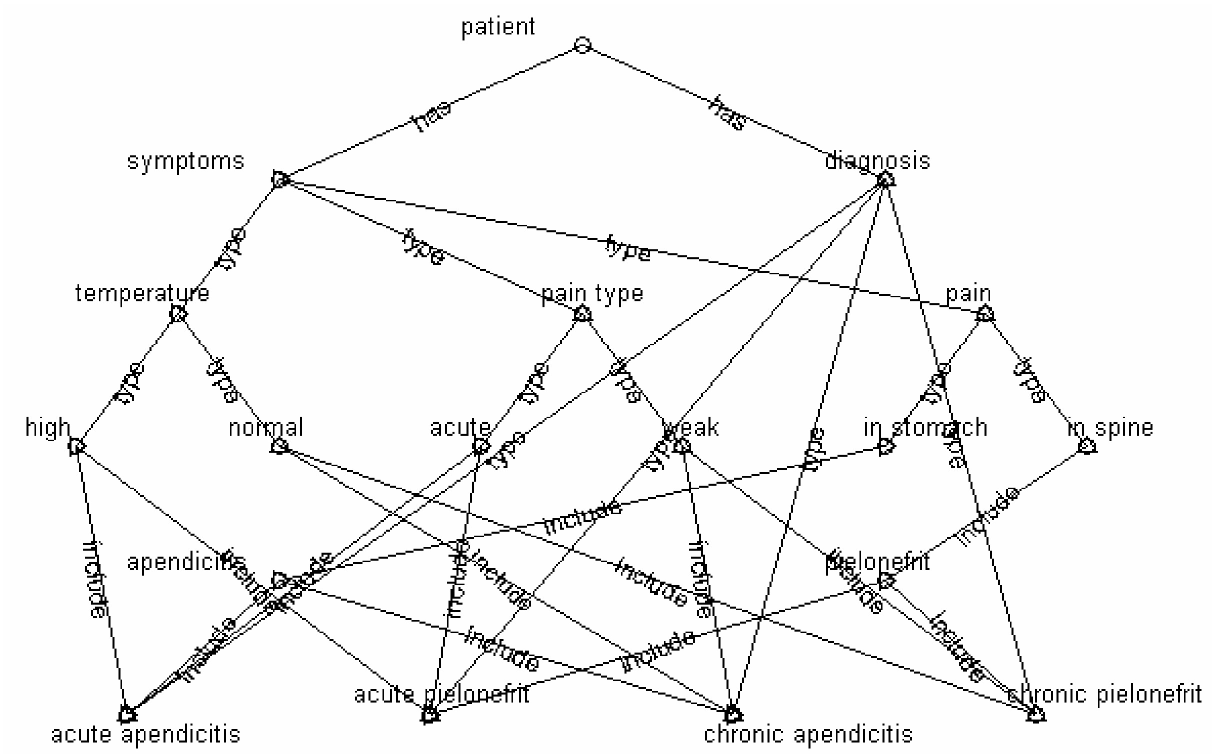


Рисунок 2.7 – Програмно побудована схема семантичної мережі з ієрархічним розташуванням вузлів

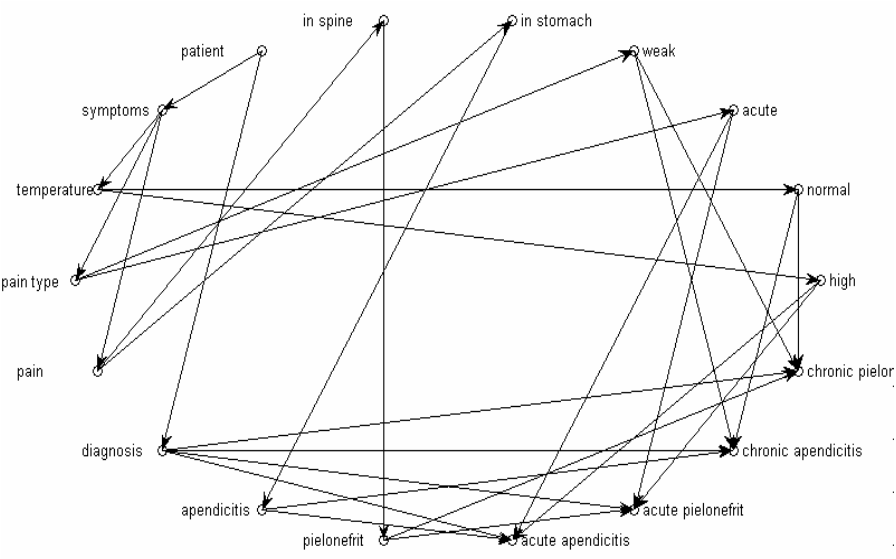


Рисунок 2.8 – Програмно побудована схема
семантичної мережі з круговим розташуванням вузлів

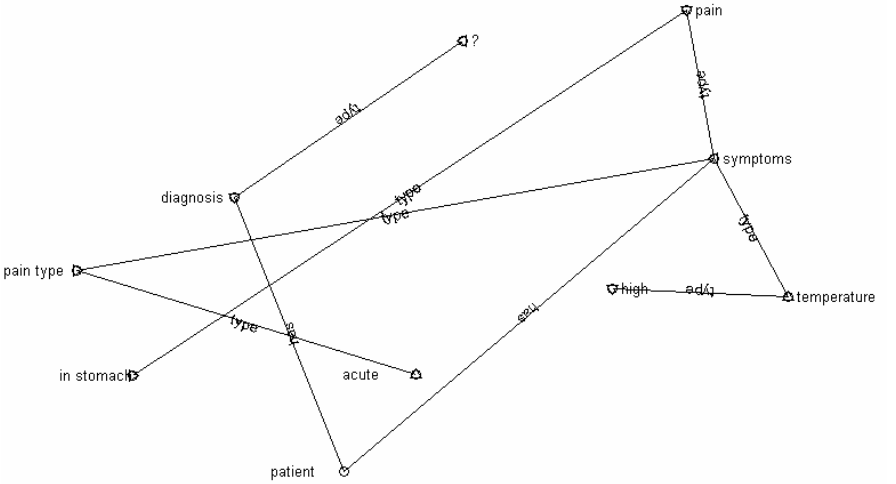


Рисунок 2.9 – Програмно побудована схема семантичної
мережі-запиту з випадковим розташуванням вузлів

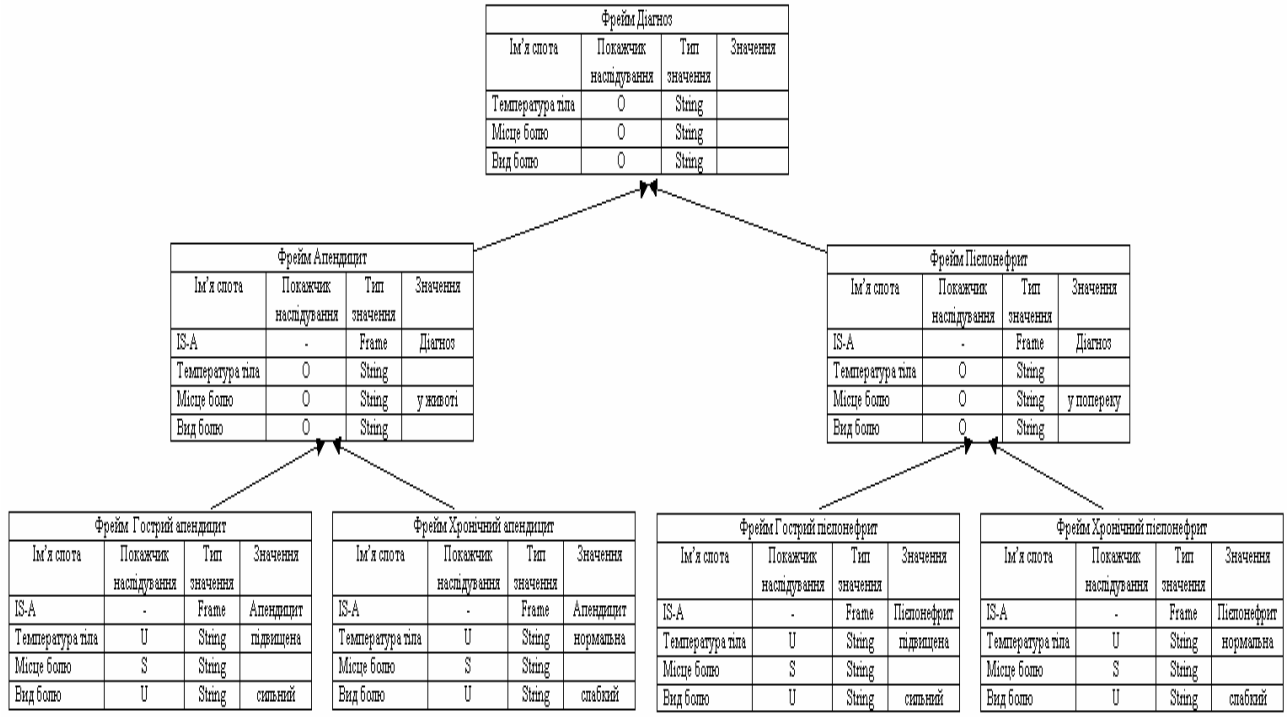


Рисунок 2.10 – Схема ієрархічної фреймової моделі

$$D_3 = A_1 \wedge B_2 \wedge C_1,\ D_4 = A_2 \wedge B_2 \wedge C_2.$$

Нехай тепер ми маємо спостереження за пацієнтом для якого: A_1 = істина, B_1 = істина, C_1 = істина, та, відповідно, A_2 = хибність, B_2 = хибність, C_2 = хибність. Тоді діагнози:

$$D_1 = \text{істина} \wedge \text{істина} \wedge \text{істина} = \text{істина},$$
$$D_2 = \text{хибність} \wedge \text{істина} \wedge \text{хибність} = \text{хибність},$$
$$D_3 = \text{істина} \wedge \text{хибність} \wedge \text{істина} = \text{хибність},$$
$$D_4 = \text{хибність} \wedge \text{хибність} \wedge \text{хибність} = \text{хибність}.$$

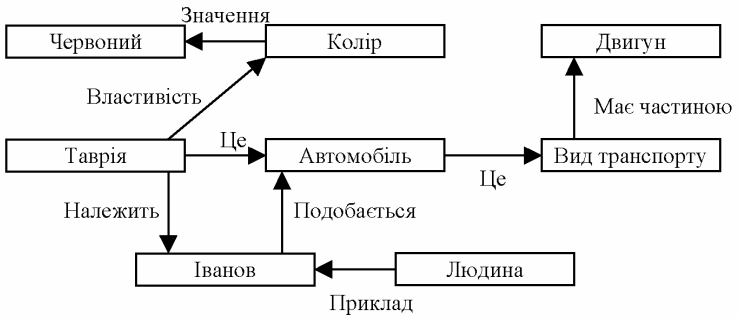
Приклад 6. Побудувати продукційну модель предметної області «Діагностика внутрішніх хвороб».

Лістинг бази знань мовою експертної системи «E2glite»

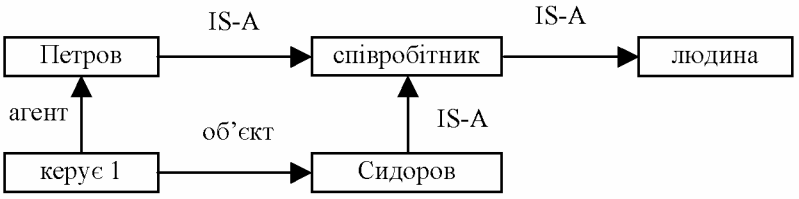
англійською мовою	**українською мовою**
REM Medical expert system	REM Медична експертна система
RULE [appendicitis] If [pain] = «in stomach» Then [disease] = «appendicitis»	RULE [апендицит] If [біль] = «у животі» Then [хвороба] = «апендицит»
RULE [pielonephritis] If [pain] = «in spine» Then [disease] = «pielonephritis»	RULE [пієлонефрит] If [біль] = «у попереку» Then [хвороба] = «пієлонефрит»
RULE [chronic appendicitis] If [temperature] = «normal» and [disease] = «appendicitis» and [pain type] = «weak» Then [diagnosis] = «chronic appendicitis»	RULE [хронічний апендицит] If [температура] = «нормальна» and [хвороба] = «апендицит» and [вид болю] = «слабкий» Then [діагноз] = «хронічний апендицит»
RULE [acute appendicitis] If [temperature] = «high» and [disease] = «appendicitis» and [pain type] = «acute» Then [diagnosis] = «acute appendicitis»	RULE [гострий апендицит] If [температура] = «підвищена» and [хвороба] = «апендицит» and [вид болю] = «сильний» Then [діагноз] = «гострий апендицит»
RULE [chronic pielonephritis] If [temperature] = «normal» and [disease] = «pielonephritis» and [pain type] = «weak» Then [diagnosis] = «chronic pielonephritis»	RULE [хронічний пієлонефрит] If [температура] = «нормальна» and [хвороба] = «пієлонефрит» and [вид болю] = «слабкий» Then [діагноз] = «хронічний пієлонефрит»
RULE [acute pielonephritis] If [temperature] = «high» and [disease] = «pielonephritis» and [pain type] = «acute» Then [diagnosis] = «acute pielonephritis»	RULE [гострий пієлонефрит] If [температура] = «підвищена» and [хвороба] = «пієлонефрит» and [вид болю] = «сильний» Then [діагноз] = «гострий пієлонефрит»
REM User dialogs	REM Діалоги користувача

PROMPT [pain] MultChoice «Where is the pain sensed?» «in stomach» «in spine» PROMPT [temperature] MultChoice «What is the body temperature?» «normal» «high» PROMPT [pain type] MultChoice «What is the type of pain?» «acute» «weak» REM The goal of decision search in knowledgebase GOAL [diagnosis]	PROMPT [біль] MultChoice «Де відчувається біль?» «у животі» «у попереку» PROMPT [температура] MultChoice «Яка температура тіла?» «нормальна» «підвищена» PROMPT [вид болю] MultChoice «Який біль?» «сильний» «слабкий» REM Мета пошуку рішення у базі знань GOAL [діагноз]

Приклад 7. Зобразимо у вигляді семантичної мережі відношення між поняттями «людина», «Іванов», «Таврія», «автомобіль», «вид транспорту» та «двигун».

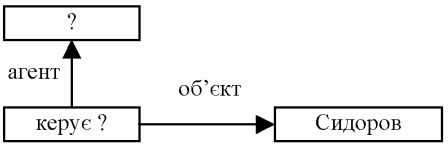


Приклад 8. Підпорядкованість співробітників організації може бути продана такою семантичною мережею.



Приведені зв'язки показують підпорядкованість першого співробітника. Інші співробітники зв'язуються через вершини мережі «керує 2", «керує 3" і т. д.

Запит: «Хто керує Сидоровим?" подамо у вигляді такої підмережі.



Зіставлення загальної мережі з мережею запиту починається з пошуку вершини «керує", що має гілку «об'єкт", спрямовану до вершини «Сидоров". Потім здійснюється перехід по гілці «агент", що і приводить до відповіді «Петров".

Приклад 9. Створити дерево вирішальних правил за допомогою Statistics Toolbox на основі масивів даних meas та species, що зберігаються в структурі fisheriris.

```
% завантажити змінні meas та species для створення
% дерева вирішальних правил
load fisheriris;

% створити дерево на основі змінних meas та species
t = treefit(meas,species);

% вивести на екран побудоване дерево в графічному вигляді
% При цьому для підпису значень незалежних змінних, на основі
% яких було побудоване дерево t, використовується параметр names
treedisp(t,'names',{'SL' 'SW' 'PL' 'PW'});

% протестувати дерево t
[c,s,n,best] = treetest(t,'cross',meas,species);

% мінімізувати (спростити) дерево
tmin = treeprune(t,'level',best);

% за допомогою дерева t отримати відповідні числові
% значення класів для екземплярів з масиву meas
sfit = treeval(t,meas);
sfit = t.classname(sfit); % отримати відповідні назви класів
mean(strcmp(sfit,species))% розрахунок точності класифікації
```

? 2.6 Контрольні питання

1. У чому полягає відміна знань від даних?
2. Які властивості мають знання?
3. Що таке інформація?
4. Що таке факт?
5. Ієрархія способів подання інформації.
6. Що таке дані, знання, метазнання?
7. Класифікація знань.
8. Особливості знань.
9. Поняття екстенсіоналу та інтенсіоналу.
10. Що таке інженерія знань?
11. Що таке набуття знань?
12. Які ви знаєте методи витягу знань?
13. Що таке теорія подання знань?
14. Класифікація моделей подання знань.
15. Порівняйте різні моделі подання знань.
16. Що таке семантична мережа?
17. Компоненти семантичних мереж.
18. Класифікація семантичних мереж.
19. Прості та ієрархічні семантичні мережі.
20. Основні типи об'єктів та зв'язків між ними.
21. Види семантичних відношень.
22. Модифікація баз знань на семантичних мережах.
23. Операція порівняння зі зразком.
24. Принципи наслідування інформації в мережній моделі.
25. Побудова та використання семантичних мереж.
26. Бібліотека SNToolbox для моделювання семантичних мереж.
27. Переваги та недоліки семантичних мереж.
28. Що таке фреймова модель?
29. Що таке фрейм?
30. Формальний опис фрейму.
31. Класифікація фреймів.
32. Структури даних фрейму.
33. Фреймові мережі.
34. Що таке агрегат?
35. Що таке приєднана процедура?
36. Що таке фрейм-зразок, фрейм-екземпляр?
37. У чому полягає пошук за зразком?
38. Принципи наслідування інформації у фреймовій мережній моделі
39. Як здійснюється пошук інформації в базі знань на основі фреймів?

40. Яка інформація може бути подана у слотах фреймів?
41. Створіть фрейм-опис або рольовий фрейм поняття, події.
42. Переваги і недоліки фреймів.
43. Що таке сценарій?
44. Що таке логічна модель?
45. Що таке логічне зв'язування?
46. Що таке числення висловлень?
47. Що таке логіка предикатів першого порядку?
48. Переваги і недоліки логічних моделей.
49. Що таке продукція?
50. Що таке продукційна система?
51. Що таке машина логічного виведення?
52. Опишіть метод роботи машини логічного виведення.
53. Які ви знаєте стратегії вирішення конфліктів правил?
54. Що таке модель дошки оголошень?
55. Переваги і недоліки продукційних моделей.
56. Що таке дерева рішень?
57. Методи побудови дерев рішень.
58. Що таке асоціативні правила?
59. Методи видобування асоціативних правил.

2.7 Практичні завдання

 *Завдання 1.* Створити базу знань у вигляді семантичної мережі для експертної системи. Зобразити її схему вручну (показати вузли із підписаними назвами та відношення із підписаними назвами). Розробити мережу запиту для семантичної мережі. Зобразити її вручну.

 *Завдання 2.* Розробити програму на мові пакету MATLAB із використанням функцій бібліотеки SNToolbox, що реалізує базу знань у вигляді семантичної мережі та графічно зображує її схему як ієрархію. Доповнити MATLAB-програму реалізацією мережі-запиту, зобразити мережу-запит програмно із круговим розташуванням вузлів. Виконати запит до семантичної мережі у пакеті MATLAB.

 *Завдання 3.* Побудувати фреймову модель для предметної області. Порівняти її із семантичною мережею.

 *Завдання 4.* Побудувати продукційну модель для експертної системи з використанням аплету E2glite. Провести тестування продукційної моделі шляхом виконання позитивного та негативного сценаріїв.

Завдання 5. Написати реферат на одну з таких тем.

1. Моделі подання знань для побудови експертних систем.
2. Мережа Інтернет як надвелика семантична мережа.
3. Програмні засоби для роботи із мережними моделями.
4. Мови для опису та обробки знань.
5. Сучасні оболонки експертних систем.
6. Методи та засоби видобування знань з даних.
7. Методи витягу знань з експертів.

2.8 Література до розділу

Визначення, класифікацію та властивості знань описано в [17, 22, 46, 47, 91, 107, 125, 126, 133, 142, 145, 148–150, 187], методи витягу знань наведено у [31–47, 88, 89, 116, 148, 149], порівняльний опис різних моделей подання знань подано в [3, 17, 22, 46, 47, 51, 82, 91, 99, 107, 133, 142, 145, 148–150, 184, 187, 207], семантичні мережі розглянуто в [17, 22, 23, 46, 47, 89, 91, 107, 119, 133, 135–140, 142], фреймові моделі описано в [17, 22, 46, 47, 91, 107, 187, 133, 142, 148–150], сценарії – в [46, 47], продукційним моделям присвячено [17, 22, 31, 45–47, 91, 104, 106, 107, 118, 125, 126, 133, 135–142, 149, 150, 168, 183, 187, 190, 201, 202], опис логічних моделей міститься в [17, 22, 46, 47, 72, 91, 106, 107, 111, 133, 135-140, 142, 148-150, 160, 187], дерева вирішальних правил розглянуто в [1, 12, 18, 29, 148, 192, 196], методи витягу асоціативних правил описано в [15, 18, 29, 61, 103, 194, 192], описи програмних засобів для подання та обробки знань наведено у [46, 47, 61, 176, 187], застосування чітких моделей подання знань розглянуто в [32, 46, 47, 61, 62, 65-70, 79, 90, 146, 148, 157].

Навчально-методичні матеріали до розділу доступні на сайті автора в мережі Інтернет за адресою: `http://csit.narod.ru`.

3. МОДЕЛІ ТА МЕТОДИ ОБРОБКИ НЕЧІТКИХ ЗНАНЬ

3.1 Нечіткість знань

При розробці інтелектуальних систем знання про конкретну предметну область, для якої створюється система, рідко бувають повними й абсолютно достовірними. Навіть кількісні дані, отримані шляхом досить точних експериментів, мають статистичні оцінки вірогідності, надійності, значимості і т. д. Інформація, якою заповнюються експертні системи, отримується у результаті опитування експертів, думки яких є суб'єктивними і можуть розходитися. Поряд із кількісними характеристиками в базах знань інтелектуальних систем повинні зберігатися якісні показники, евристичні правила, текстові знання і т. д. При обробці знань із застосуванням твердих механізмів формальної логіки виникає протиріччя між нечіткими знаннями і чіткими методами логічного виведення. Розв'язати це протиріччя можна шляхом подолання нечіткості знань (коли це можливо) або використанням спеціальних методів подання й обробки нечітких знань.

Зміст терміна *нечіткість* багатозначний та включає такі основні компоненти: недетермінованість виведень, багатозначність, ненадійність, неповнота, неточність.

Недетермінованість виведень – це характерна риса більшості систем штучного інтелекту. Недетермінованість означає, що заздалегідь шлях вирішення конкретної задачі в просторі її станів визначити неможливо. Тому в більшості випадків методом проб і помилок вибирається деякий ланцюжок логічних висновків, що узгоджуються з наявними знаннями, а у випадку якщо він не приводить до успіху, організується перебір з поверненням для пошуку іншого ланцюжка і т. д. Такий підхід припускає визначення деякого первісного шляху. Для вирішення подібних задач запропоновано багато евристичних методів. Недетермінованість виведень варто враховувати при розробці ефективних способів подання і збереження знань, а також при побудові методів пошуку й обробки знань, що дозволяють одержати рішення задачі за найменше число кроків. Для побудови таких методів звичайно застосовуються евристичні *метазнання* (знання про знання).

Багатозначність інтерпретації – звичайне явище в задачах розпізнавання. При розумінні природної мови серйозними проблемами стають багатозначність змісту слів, їхня підпорядкованість, порядок слів у реченні і т. п. Проблеми розуміння змісту виникають у будь-якій системі, що взаємодіє з користувачем природною мовою. Розпізнавання графічних образів також пов'язано з вирішенням проблеми багатозначної інтерпретації. При комп'ютерній обробці знань багатозначність необхідно усувати шляхом вибору правильної інтерпретації, для чого розроблено спеціальні методи.

Ненадійність знань і виведень означає, що для оцінки вірогідності знань не можна застосувати двобальну шкалу (1 – абсолютно достовірні знання, 0 – недостовірні знання). Для більш тонкої оцінки вірогідності знань застосовується імовірнісний підхід, заснований на теоремі Байєса, і інші методи (наприклад, метод виведення з використанням коефіцієнтів упевненості). Широке застосування на практиці одержали нечіткі виведення, які будуються на базі нечіткої логіки, що веде своє походження від теорії нечітких множин.

Неповнота знань і немонотонна логіка. Абсолютно повних знань не буває, оскільки процес пізнання нескінченний. У зв'язку з цим стан бази знань повинний змінюватися з часом. На відміну від простого додавання інформації, як у базах даних, при додаванні нових знань виникає небезпека одержання суперечливих висновків: тобто висновки, отримані з використанням нових знань, можуть спростовувати ті, що були отримані раніше. Ще гірше, якщо нові знання будуть знаходитися в протиріччі з старими, тоді механізм виведення може стати непрацездатним.

Більшість експертних систем першого покоління були засновані на моделі закритого світу, обумовленій застосуванням апарату формальної логіки для обробки знань.

Модель закритого світу припускає твердий добір знань, що включаються в базу, а саме: база знань заповнюється винятково вірними поняттями, а усе, що ненадійно або невиразно, свідомо вважається помилковим. Така модель має обмежені можливості подання знань і таїть у собі небезпеку одержання протиріч при додаванні нової інформації. Недоліки моделі закритого світу пов'язані з тим, що формальна логіка виходить з передумови, відповідно до якої набір визначених у системі аксіом (знань) є *повним* (теорія є повною, якщо кожний її факт можна довести, виходячи з аксіом цієї теорії). Для повного набору знань справедливість раніше отриманих виведень не порушується з додаванням нових фактів. Ця властивість логічних виведень називається *монотонністю*. На жаль, реальні знання, що закладаються в експертні системи, украй рідко бувають повними.

*Неточність знань***.** Відомо, що кількісні дані (знання) можуть бути неточними, при цьому існують кількісні оцінки такої неточності (довірчий інтервал, рівень значимості, ступінь адекватності і т. д.). Лінгвістичні знання також можуть бути неточними. Для врахування неточності лінгвістичних знань використовується теорія нечітких множин. Фактично нечіткість може бути ключем до розуміння здатності людини справлятися з задачами, що занадто складні для вирішення на ЕОМ. Розвиток досліджень в області нечіткої математики призвів до появи нечіткої логіки і нечітких виведень, що виконуються з використанням знань, представлених нечіткими множинами, нечіткими відношеннями, нечіткими відповідностями і т. д.

3.2 Теорія нечітких множин

Теорія нечітких множин (fuzzy sets theory) бере свій початок з 1965 року, коли професор Лотфі Заде (Lotfi Zadeh) з університету Берклі опублікував основну роботу «Fuzzy Sets» у журналі «Information and Control». Прикметник «fuzzy» (нечіткий, розмитий), введено в назву нової теорії з метою відокремлення від традиційної чіткої математики й аристотелевої логіки, що оперують з чіткими поняттями: «належить – не належить», «істина – хибність». Концепція нечіткої множини зародилася у Заде «як незадоволеність математичними методами класичної теорії систем, що змушувала домагатися штучної точності, недоречної в багатьох системах реального світу, особливо в так званих гуманістичних системах, що включають людей».

Початком практичного застосування теорії нечітких множин можна вважати 1975 рік, коли Е. Мамдані (E. Mamdani) побудував перший нечіткий контролер. Успіх першого промислового контролера, заснованого на нечітких лінгвістичних правилах «Якщо – то» привів до сплеску інтересу до теорії нечітких множин серед математиків та інженерів.

Можливість використання нечіткої логіки базується на таких результатах.

У 1992 р. Коско (B. Kosko) була доведена *теорема про нечітку апроксимацію* (Fuzzy Approximation Theorem), відповідно до якої будь-яка математична система може бути апроксимована системою, заснованою на нечіткій логіці. Іншими словами, за допомогою природно-мовних висловлень-правил «Якщо – то», з подальшою їх формалізацією засобами теорії нечітких множин, можна скільки завгодно точно відбити довільний взаємозв'язок «вхід – вихід» без використання складного апарата диференціального й інтегрального числень, традиційно застосовуваного в керуванні й ідентифікації.

У 1992 р. Ванг (L. Wang) показав, що нечітка система є універсальним апроксиматором, тобто може апроксимувати будь-яку неперервну функцію з довільною точністю, якщо використовує набір з n ($n \to \infty$) правил виду «Якщо – то», гаусові функції приналежності, композиції у вигляді добутку, імплікації у формі Ларсена та центроїдний метод приведення до чіткості.

У 1995 р. Кастро (J. Castro) показав, що логічний контролер Мамдані також є універсальним апроксиматором при симетричних трикутних функціях приналежності, композиції з використанням операції мінімум, імплікації у формі Мамдани і центроїдного методу приведення до чіткості.

Системи з нечіткою логікою *доцільно застосовувати* для складних процесів, коли немає простої математичної моделі, а також якщо експертні знання про об'єкт або про процес можна сформулювати тільки в лінгвістичній формі.

Системи, що базуються на нечіткій логіці, *застосовувати недоцільно* якщо необхідний результат може бути отриманий яким-небудь іншим (стан-

дартним) шляхом, або якщо для об'єкта або процесу вже знайдена адекватна і легко досліджувана математична модель.

Основні *недоліки систем з нечіткою логікою*: вихідний набір нечітких правил, що постулюються, формулюється експертом-людиною і може виявитися неповним або суперечливим; вид і параметри функцій приналежності, що описують вхідні і вихідні змінні системи, вибираються суб'єктивно і можуть виявитися такими, що цілком не відбивають реальну дійсність.

3.3 Нечіткі множини та змінні

Нехай E – універсальна множина, x – елемент E, а G – деяка властивість. Звичайна *(чітка) підмножина A* універсальної множини E, елементи якої мають властивість G, визначається як множина впорядкованих пар $\{<\mu_A(x)|x>\}$, де $\mu_A(x)$ – характеристична функція приналежності, що приймає значення 1, якщо x має властивість G, та 0 – у протилежному випадку.

Нечітка підмножина відрізняється від звичайної тим, що для елементів x з E немає однозначної відповіді «ні» або «так» щодо властивості G. У зв'язку з цим *нечітка підмножина A* універсальної множини E визначається як множина впорядкованих пар $A = \{<\mu_A(x)|x>\}$, де $\mu_A(x)$ – характеристична *функція приналежності* (або просто функція приналежності), що приймає значення в деякій цілком впорядкованій множині M (наприклад, $M = [0; 1]$).

Функція приналежності вказує ступінь приналежності елемента x підмножині A. Множину M називають *множиною приналежностей*. Якщо $M = \{0, 1\}$, то нечітка підмножина A може розглядатися як чітка множина.

Нечітка змінна визначається як $<a, E, A>$, де a – найменування змінної, $E = \{x\}$ – область визначення змінної, набір можливих значень x, $A=\{<\mu_A(x)|x>\}$ – нечітка множина, що описує обмеження на можливі значення змінної a (семантику). Нечітка змінна – це теж саме, що і нечітке число, тільки з додаванням імені, яким формалізується поняття, що описується цим числом.

Лінгвістична змінна визначається як $<B, T, X, G, M>$, де B – найменування змінної, T – множина її значень (базова терм-множина), що складається з найменувань нечітких змінних, областю визначення кожної з яких є множина X; G – синтаксична процедура (граматика), що дозволяє оперувати елементами терм-множини T, зокрема – генерувати нові осмислені терми; $T'=T \cup G(T)$ задає розширену терм-множину ($\cup$ – знак об'єднання); M – семантична процедура, що дозволяє приписати кожному новому значенню лінгвістичної змінної нечітку семантику, шляхом формування нової нечіткої множини.

Лінгвістична змінна – це множина нечітких змінних, вона використовується для того, щоб дати словесний опис деякому нечіткому числу, отриманому в результаті деяких операцій.

Терм-множина – це множина всіх можливих значень лінгвістичної змінної.

Терм – будь-який елемент терм-множини. У теорії нечітких множин терм формалізується нечіткою множиною за допомогою функції приналежності.

Нечіткий терм – це нечітка множина, яка має властивість, якій відповідає певне поняття.

3.4 Функції приналежності

Функції приналежності нерозривно пов'язані із нечіткими множинами. Тип функції приналежності в значному ступені визначає властивості нечіткої системи.

Задавання функцій приналежності можна здійснювати у вигляді списку з явним перерахуванням усіх елементів та відповідних ним значень функції приналежності (наприклад, використовуючи відносні частоти за даними експерименту як значення приналежності), або аналітично у вигляді формул (наприклад, використовуючи типові форми кривих для завдання функцій приналежності (у формі (*L*-*R*)-типу) з уточненням їхніх параметрів відповідно до даних експерименту).

Існують прямі та непрямі *методи побудови функцій приналежності*.

При використанні *прямих методів* експерт просто задає для кожного $x \in E$ значення $\mu(x)$. Як правило, прямі методи побудови функції приналежності використовуються для вимірних понять, таких як швидкість, час, відстань, тиск, температура і т. д., або тоді, коли виділяються полярні значення.

У багатьох задачах при характеристиці об'єкта можна виділити набір ознак і для кожної з них визначити полярні значення, що відповідають значенням функції приналежності 0 або 1. Для конкретного об'єкта експерт, виходячи з приведеної шкали, задає $\mu_A(x) \in [0, 1]$, формуючи векторну функцію приналежності $\{\mu_A(x_1), \mu_A(x_2), ..., \mu_A(x_n)\}$.

Різновидом прямих методів побудови функцій приналежності є *прямі групові методи*, коли, наприклад, групі експертів пред'являють конкретний об'єкт, і кожен повинний дати одну з двох відповідей: належить чи не належить цей об'єкт до заданої множини. Тоді число позитивних відповідей, поділене на загальне число експертів, дає значення функції приналежності об'єкта до даної нечіткої множини.

Непрямі методи визначення значень функції приналежності використовуються у випадках, коли немає вимірних елементарних властивостей, через які визначається нечітка множина. Як правило, це *методи попарних порівнянь*. Якщо значення функцій приналежності відомі, наприклад, $\mu_A(x_i) = w_i$, $i = 1, 2, ..., n$, то попарні порівняння можна подати матрицею відношень $A = \{a_{ij}\}$, де $a_{ij} = w_i/w_j$ (операція розподілу).

На практиці експерт сам формує матрицю А, при цьому передбачається, що діагональні елементи дорівнюють 1, а для елементів, симетричних щодо головної діагоналі, $a_{ij} = 1/a_{ji}$, тобто якщо один елемент оцінюється як в а разів більш значущий ніж інший, то цей останній повинний бути в $1/a$ разів більш значущим, ніж перший. У загальному випадку задача зводиться до пошуку вектора w, що задовольняє рівнянню виду: $Aw = \lambda_{max} w$, де λ_{max} – найбільше власне значення матриці A. Оскільки матриця A позитивна за побудовою, розв'язок даної задачі існує і є позитивним.

Обмеженням методів попарного порівняння є використання суб'єктивної інформації і деяких допущень при перетворенні її в ступені приналежності нечітких множин.

Оптимізаційні методи побудови функцій приналежності засновані на параметричній ідентифікації нечітких моделей за експериментальними даними «входи – вихід», при якій оптимізують параметри функцій приналежності з метою мінімізації відхилення між експериментальними даними і результатами нечіткого моделювання. Використання оптимізаційного підходу знімає суб'єктивізм побудови функцій приналежності, однак замість цього вимагає навчаючої вибірки та нечіткої моделі «входи – вихід». Недоліком даних методів є те, що функції приналежності однакових за змістом нечітких множин виходять різними в результаті ідентифікації різних залежностей «входи – вихід». Таким чином, функція приналежності стає сильно чуттєвою до навчаючої вибірки та структури нечіткої моделі.

Метод побудови функцій приналежності шляхом кластеризації експериментальних даних. Будемо вважати, що відомі числові значення певного показника: $(y_1, y_2, ..., y_v)$, де v – кількість значень. Розглядаються дві задачі побудови функцій приналежності за цими даними.

Перша задача – синтез нечітких множин $Y_1, Y_2, \ldots, Y_C$, функції приналежності яких відповідають скупченням даних $(y_1, y_2, ..., y_v)$: $(y_1, y_2, ..., y_v) \rightarrow (\mu_{ij})$, $i = 1, 2, \ldots, v$, $j = 1, 2, \ldots, C$, де μ_{ij} – ступінь приналежності елемента нечіткій множині Y_j. Для вирішення даної задачі використовують нечітку кластеризацію за методом *нечітких с-середніх*.

Друга задача – синтез однієї нечіткої множини Y, функція приналежності якої відповідає розподілу даних $(y_1, y_2, ..., y_v)$, що відповідає відображенню виду: $(y_1, y_2, ..., y_v) \rightarrow (\mu_1, \mu_2, \ldots, \mu_v)$, $i = 1, 2, \ldots, v$, $j = 1, 2, \ldots, C$, де де μ_i – ступінь приналежності елемента нечіткій множині Y. Для вирішення даної задачі використовують *потенційну функцію* з *гірського методу кластеризації*.

Потенціал точки – це число, що показує наскільки щільно в її околиці розташовані експериментальні дані. Чим вище потенціал точки, тим ближче вона до центра кластера. Потенціал точки y_i, $i = 1, 2, \ldots, v$, визначається за формулою:

$$\varphi_i = \sum_{j=1}^{v} e^{-4\alpha^2 (y_i - y_j)^2},$$

де $\alpha > 0$ – коефіцієнт, що задає ступінь компактності кластера. Перед застосуванням цієї формули експериментальні дані слід відобразити на відрізок [0, 1].

Ступені приналежності нечіткій множині Y розраховують у такий спосіб:

$$\mu_Y(y_i) = \frac{\varphi_i}{\max\limits_{j=1,2,...,v} \varphi_j}.$$

При необхідності знайдену нечітку множину *Y* можна апроксимувати придатною параметричною функцією приналежності.

Візуалізувати функції приналежності нечітких множин можна шляхом побудови графіку залежності значення функції приналежності μ від значення елемента нечіткої множини *x*.

Виділяють такі основні *типи функцій приналежності*: кусочно-лінійні функції, *Z*-образні та *S*-образні функції, *П*-образні функції.

Розглянемо основні *функції приналежності*. При цьому визначимо формат виклику функцій пакету MATLAB, що реалізують відповідні функції приналежності.

Кусочно-лінійні функції.

Трикутна функція:

$$\mu(x) = \begin{cases} 0, x \le a, \\ \dfrac{x-a}{b-a}, a < x < b, \\ \dfrac{c-x}{c-b}, b \le x < c, \\ 0, c \le x, \end{cases}$$

де *a*, *b*, *c* – числові параметри, що приймають довільні дійсні значення й упорядковані відношенням: $a \le b \le c$. Ця функція в пакеті Matlab має ім'я trimf, порядок її параметрів:[*a*, *b*, *c*].

Трапецієподібна функція:

$$\mu(x) = \begin{cases} 0, x < a, \\ \dfrac{x-a}{b-a}, a \le x < b, \\ 1, b \le x \le c, \\ \dfrac{d-x}{d-c}, c < x \le d, \\ 0, d < x, \end{cases}$$

де *a*, *b*, *c*, *d* – числові параметри, що приймають довільні дійсні значення й упорядковані відношенням: $a \le b \le c \le d$. Ця функція в пакеті Matlab має ім'я trapmf, порядок її параметрів:[*a*, *b*, *c*, *d*].

Z-образні та S-образні функції

Z-образна крива (*сплайн-функція*):

$$\mu(x)=\begin{cases}1, x<a,\\ \frac{1}{2}+\frac{1}{2}cos\left(\frac{x-a}{b-a}\pi\right), a\le x\le b,\\ 0, x>b,\end{cases}$$

де *a*, *b* – числові параметри, що приймають довільні дійсні значення й упорядковані відношенням: *a* < *b*.

Сплайн-функція може бути також задана іншим виразом:

$$\mu(x)=\begin{cases}1, x\le a,\\ 1-2\left(\frac{x-a}{b-a}\right)^2, a<x\le\frac{a+b}{2},\\ 2\left(\frac{b-x}{b-a}\right)^2, \frac{a+b}{2}<x<b,\\ 0, b\le x,\end{cases}$$

де *a*, *b* – числові параметри, що приймають довільні дійсні значення й упорядковані відношенням: *a* < *b*. Ця функція в пакеті Matlab має ім'я zmf, порядок її параметрів:[*a*, *b*].

Лінійна Z-образна функція:

$$\mu(x)=\begin{cases}1, x\le a,\\ \frac{b-x}{b-a}, a<x<b,\\ 0, b\le x,\end{cases}$$

де *a*, *b* – числові параметри, що приймають довільні дійсні значення й упорядковані відношенням: *a* < *b*.

S-образна крива (сплайн-функція):

$$\mu(x)=\begin{cases}0, x<a,\\ \frac{1}{2}+\frac{1}{2}\cos\left(\frac{x-b}{b-a}\pi\right), a\le x\le b,\\ 1, x>b,\end{cases}$$

де *a*, *b* – числові параметри, що приймають довільні дійсні значення й упорядковані відношенням: *a* < *b*.

Сплайн-функція може бути також задана іншим виразом:

$$\mu(x)=\begin{cases}0, x\le a,\\ 2(x-a)^2(b-a)^{-2}, a<x\le 0{,}5(a+b),\\ 1-2(b-x)^2(b-a)^{-2}, 0{,}5(a+b)<x<b,\\ 1, b\le x,\end{cases}$$

де *a*, *b* – числові параметри, що приймають довільні дійсні значення й упорядковані відношенням: *a* < *b*. Ця функція в пакеті Matlab має ім'я smf, порядок її параметрів:[*a*, *b*].

Лінійна S-образна функція:

$$\mu(x)=\begin{cases}0, x\le a,\\ \dfrac{x-a}{b-a}, a<x<b,\\ 1, b\le x,\end{cases}$$

де *a*, *b* – числові параметри, що приймають довільні дійсні значення й упорядковані відношенням: *a* < *b*.

Сигмоїдна (сигмоїдальна) функція може бути віднесена одночасно до *Z*-образних і *S*-образних функцій: $\mu(x)=\dfrac{1}{1+e^{-a(x-b)}}$, де *a*, *b* – числові параметри, що приймають довільні дійсні значення й упорядковані відношенням: *a* < *b*, *e* – основа натуральних логарифмів. При *a* > 0 може бути отримана *S*-образна функція, при *a* < 0 – *Z*-образна функція. Ця функція в пакеті Matlab має ім'я sigmf, порядок її параметрів:[*a*, *b*].

П-образні функції

П-образна (пі-образна) функція: $\mu(x)=\mu_S(x)\mu_Z(x)$, де $\mu_S(x)$ – *S*-образна функція, $\mu_Z(x)$ – *Z*-образна функція. Ця функція в пакеті Matlab має ім'я pimf, порядок її параметрів:[*a b c d*], де [*a d*] – носій нечіткої множини; [*b c*] – ядро нечіткої множини; [*a b*] – параметри функції smf, [*c d*] – параметри функції zmf.

Добуток двох сигмоїдних функцій: $\mu(x)=\dfrac{1}{1+e^{-a(x-b)}}\cdot\dfrac{1}{1+e^{-c(x-d)}}$,

де *a*, *b*, *c*, *d* – числові параметри, що приймають довільні дійсні значення, причому *a* > 0, *c* < 0, і упорядковані відношенням: $a \le b < |c| \le d$, *e* – основа натуральних логарифмів. Ця функція в пакеті Matlab має ім'я psigmf, порядок її параметрів:[*a b c d*].

Різниця між двома сигмоїдними функціями:

$$\mu(x)=\frac{1}{1+e^{-a(x-b)}}-\frac{1}{1+e^{-c(x-d)}},$$

де *a*, *b*, *c*, *d* – числові параметри, що приймають довільні дійсні значення й упорядковані відношенням: *a*<*b*, *c*<*d*, *e* – основа натуральних логарифмів. Ця функція в пакеті Matlab має ім'я dsigmf, порядок її параметрів:[*a b c d*].

Узагальнена колоколообразна функція:

$$\mu(x)=\frac{1}{1+\left|\dfrac{x-c}{a}\right|^{2b}},$$

де a, b, c – числові параметри, що приймають довільні дійсні значення й упорядковані відношенням: $a < b < c$, при чому $b > 0$. Ця функція в пакеті Matlab має ім'я gbellmf, порядок її параметрів:[a b c].

Симетрична гаусівська функція щільності нормального розподілу:

$$\mu(x) = e^{-\frac{(x-b)^2}{2a^2}},$$

де a^2 – дисперсія розподілу, b – математичне сподівання. Ця функція в пакеті Matlab має ім'я gaussmf, порядок її параметрів:[a b].

Двостороння гаусівська функція приналежності:

$$\mu(x) = e^{-\frac{(x-b)^2}{2a^2}} \cdot e^{-\frac{(x-d)^2}{2c^2}},$$

де a^2, c^2 – дисперсії розподілів, b, d – математичні сподівання. Ця функція в пакеті Matlab має ім'я gauss2mf, порядок її параметрів:[a b c d].

3.5 Основні характеристики та властивості нечітких множин

Нехай A – нечітка множина з елементами з універсальної множини E та множиною приналежностей $M = [0; 1]$.

Висотою (супремумом) нечіткої множини A називається величина $\sup(\mu_A) = \max\limits_{x \in E} \mu_A(x)$ – це точна верхня грань або максимальне значення приналежності, що є у множині.

Нормальною є нечітка множина A, якщо її висота дорівнює 1, тобто верхня межа її функції приналежності дорівнює 1 ($\max\limits_{x \in E} \mu_A(x) = 1$).

Субнормальною називається нечітка множина A при $\max\limits_{x \in E} \mu_A(x) < 1$.

Порожньою називається нечітка множина A (порожня множина позначається як $\varnothing$), якщо $\forall x \in E : \mu_A(x) = 0$.

Унімодальною є нечітка множина A, якщо $\mu_A(x) = 1$ тільки на одному x з E.

Ядром нечіткої множини A називають таку звичайну множину A_1, елементи якої задовольняють умові: $A_1 = \{ x \in E \mid \mu_A(x) = 1\}$.

Носієм нечіткої множини A є звичайна підмножина B з властивістю $\mu_A(x) > 0$, тобто $B = \{ x \in E \mid \mu_A(x) > 0\}$.

Скінченою є нечітка множина, якщо її носій є скінченою множиною.

Нескінченою є нечітка множина, якщо її носій є нескінченою множиною.

Межами нечіткої множини A є такі елементи універсальної множини E, для яких значення функції приналежності $\mu_A(x)$ відрізняються від 0 та 1: $0 < \mu_A(x) < 1$.

Точками переходу множини A називаються такі елементи $x \in E$, для яких $\mu_A(x) = 0{,}5$.

Найближчою чіткою множиною A_1 до нечіткої множини A є множина з характеристичною функцією:

$$\mu_{A_1}(x) = \begin{cases} 0, \text{якщо } \mu_A(x) < 0{,}5, \\ 1, \text{якщо } \mu_A(x) > 0{,}5, \\ 0 \text{ або } 1, \text{якщо } \mu_A(x) = 0{,}5. \end{cases}$$

Опуклою називають нечітку множину A, якщо її функція приналежності задовольняє нерівності: $\mu_A(x) \geq \min\{\mu_A(a), \mu_A(b)\}$, для будь-яких $x, a, b \in E$, при котрих: $a < x < b$ та $a \neq b$.

Міру нечіткості Ягера нечіткої множини A у метриці p визначають як:

$$\text{Fuz}_p(A) = 1 - \frac{D_p(A, \overline{A})}{n^{1/p}},$$

де $D_p(A, \overline{A})$ – міра відстані між множинами $A та \overline{A}$, що містять n елементів. Значення $p = 1$ відповідає метриці Хеммінга: $D_1(A, \overline{A}) = \sum_{i=1}^{n} |2\mu_A(x_i) - 1|$, а $p = 2$ – метриці Евкліда: $D_2(A, \overline{A}) = \sqrt{\sum_{i=1}^{n} (2\mu_A(x_i) - 1)^2}$.

Лінійний індекс нечіткості визначається за формулою:

$$\nu(A) = \frac{2}{n} \sum_{i=1}^{n} \min\{\mu_A(x_i), 1 - \mu_A(x_i)\}.$$

Квадратичний індекс нечіткості визначається за формулою:

$$\eta(A) = \frac{2}{\sqrt{n}} \sqrt{\sum_{i=1}^{n} \min\{\mu_A^2(x_i), 1 - \mu_A(x_i)\}}.$$

Ентропія нечіткої множини A визначається за формулою:

$$H(A) = -\frac{1}{\ln n} \sum_{i=1}^{n} (\pi_A(x_i) \ln \pi_A(x_i)), \quad \pi_A(x_i) = \frac{\mu_A(x_i)}{\sum_{i=1}^{n} \mu_A(x_i)}.$$

Міру нечіткості Коско (ентропійну) визначають як:

$$\text{Fuz}(A) = \frac{\text{card}(A \cap \overline{A})}{\text{card}(A \cup \overline{A})},$$

де card(F) – кардинальне число множини F.

Кардинальне число чіткої скінченої множини F_1, що складається з n елементів, визначається за формулою: card(F_1) = n.

Кардинальне число нечіткої множини F визначається за формулою:

$$\text{card}(F) = \sum_{i=1}^{n} \mu_A(x_i) \cdot$$

Чітка множина α-рівня (*альфа-зріз*) A_α нечіткої множини A універсальної множини E – елементи, приналежність яких вище або дорівнює заданому порогу: $A_\alpha = \{x | \mu_A(x) \geq \alpha\}$, де α – поріг: $\alpha \leq 1$, (поріг, що дорівнює 1/2, називають *точкою переходу*). Властивість множини α-*рівня*: якщо $\alpha_1 \geq \alpha_2$, то $A_{\alpha_1} \leq A_{\alpha_2}$.

3.6 Операції над нечіткими множинами

Логічні операції

Нехай A та B – нечіткі множини на універсальній множині E.

Доповнення (*заперечення множини*) $\overline{A}$*:* інвертується приналежність кожного елемента: $\forall x \in E$, $M = [0, 1]$: $\mu_{\overline{A}}(x) = 1 - \mu_A(x)$. Тут доповнення визначене для $M = [0, 1]$, але очевидно, що його можна визначити для будь-якого упорядкованого M).

Інволюція – подвійне доповнення: $\overline{(\overline{A})} = A$.

Включення (*підмножина*, *домінування*) $A \subseteq B$: A міститься в B (B домінує над A), якщо $\forall x \in E$: $\mu_A(x) \leq \mu_B(x)$.

Непорівненими є нечіткі множини A та B, задані на E, якщо для них не виконуються ані $A \subseteq B$, ані $B \subseteq A$.

Рівність $A = B$: A та B рівні, якщо $\forall x \in E$: $\mu_A(x) = \mu_B(x)$.

Власною нечіткою підмножиною A нечіткої множини B ($A \subset B$) називають, якщо $\forall x \in E$: $\mu_A(x) < \mu_B(x)$.

Об'єднання (*логічна сума множин*): $A \cup B$ – найменша нечітка підмножина, що містить як A, так і B, з функцією приналежності: $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$ – створюється нова множина з елементів вихідних множин, причому для однакових елементів приналежність береться максимальною. Операція об'єднання моделює логічне зв'язування «АБО».

Перетинання (*логічний добуток множин*): $A \cap B$ – найбільша нечітка підмножина, що міститься одночасно в A та B: $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$ – створюється нова множина з однакових елементів вихідних множин, приналежність яких береться мінімальною. Операція перетинання моделює логічне зв'язування «ТА».

Властивості операцій об'єднання і перетинання. Нехай A, B, C – нечіткі множини, тоді виконуються наступні властивості:

- комутативність: $A \cup B = B \cup A, A \cap B = B \cap A$;
- асоціативність: $(A \cup B) \cup C = A \cup (B \cup C), (A \cap B) \cap C = A \cap (B \cap C)$;

- ідемпотентність: $A \cup A = A, A \cap A = A$;
- дистрибутивність: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C), A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$;
- поглинання: $A \cap (A \cup B) = A \cup (A \cap B) = A$;
- універсальні верхня та нижня межі: $A \cup \varnothing = A$, $A \cap \varnothing = \varnothing$, $A \cap E = A$, $A \cup E = E$;
- закони де Моргана: $\overline{A \cup B} = \overline{A} \cap \overline{B}, \overline{A \cap B} = \overline{A} \cup \overline{B}$.

На відміну від чітких множин, для нечітких множин у загальному випадку не виконуються закон виключення третього і закон тотожності, тобто: $A \cap \overline{A} \neq \varnothing$ та $A \cup \overline{A} \neq E$.

Різниця $A \setminus B$ або $A - B$: $\mu_{A \setminus B}(x) = \max\{\mu_A(x) - \mu_B(x), 0\}$. Зауважимо, що $A \setminus B \neq B \setminus A$.

Симетрична різниця: $A \Theta B = (A \setminus B) \cup (B \setminus A)$, $A \Theta B = (A \cup B) \cap (\overline{A} \cup \overline{B})$: $\mu_{A \Theta B}(x) = |\mu_A(x) - \mu_B(x)|$.

Диз'юнктивна сума: $A \oplus B = (A \cap \overline{B}) \cup (\overline{A} \cap B)$ з функцією приналежності: $\mu_{A \oplus B}(x) = \max(\min(\mu_A(x), 1 - \mu_B(x)), \min(1 - \mu_A(x), \mu_B(x)))$.

Алгебраїчні операції над нечіткими множинами

Алгебраїчний добуток (алгебраїчне перетинання) $A \cdot B$: $\forall x \in E$: $\mu_{AB}(x) = \mu_A(x)\mu_B(x)$.

Алгебраїчна сума (алгебраїчне об'єднання) $A \hat{+} B$: $\forall x \in E : \mu_{A \hat{+} B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$.

Властивості операцій алгебраїчної суми й алгебраїчного добутку:

- комутативність: $A \cdot B = B \cdot A, A \hat{+} B = B \hat{+} A$;
- асоціативність: $(A \cdot B) \cdot C = A \cdot (B \cdot C), (A \hat{+} B) \hat{+} C = A \hat{+} (B \hat{+} C)$;
- універсальні верхня та нижня межі: $A \hat{+} \varnothing = A$, $A \cdot \varnothing = \varnothing$, $A \cdot E = A$, $A \hat{+} E = E$;
- закони де Моргана: $\overline{A \hat{+} B} = \overline{A} \cdot \overline{B}, \overline{A \cdot B} = \overline{A} \hat{+} \overline{B}$.

Не виконуються:

- ідемпотентність, тобто: $A \hat{+} A \neq A$, $A \cdot A \neq A$;
- дистрибутивність, тобто: $A \cdot (B \hat{+} C) \neq (A \cdot B) \hat{+} (A \cdot C), A \hat{+} (B \cdot C) \neq (A \hat{+} B) \cdot (A \hat{+} C)$;
- поглинання, тобто: $A \cdot (A \hat{+} B) \neq A, A \hat{+} (A \cdot B) \neq A$;
- закон виключення третього і закон тотожності, тобто: $A \cdot \overline{A} \neq \varnothing, A \hat{+} \overline{A} \neq E$.

При спільному використанні операцій $\{\cup,\cap,\hat{+},\cdot\}$ справедливі властивості: $A\cdot(B\cup C)=(A\cdot B)\cup(A\cdot C)$, $A\cdot(B\cap C)=(A\cdot B)\cap(A\cdot C)$, $A\hat{+}(B\cup C)=(A\hat{+}B)\cup(A\hat{+}C)$, $A\hat{+}(B\cap C)=(A\hat{+}B)\cap(A\hat{+}C)$.

Обмежена сума (граничне об'єднання) $A|+|B$:

$$\mu_{A|+|B}(x) = \min\{1, \mu_A(x)+\mu_B(x)\}.$$

Обмежена різниця $A|–|B$: $\mu_{A|–|B} = \max\{0, \mu_A(x) – \mu_B(x)\}$.

Обмежений добуток (граничне перетинання) $A|\cdot|B$:

$$\mu_{A|\cdot|B}(x) = \max\{0, \mu_A(x)+\mu_B(x)–1\}.$$

Драстичне перетинання (від англ. drastic – радикальний) $A\,\Delta\,B$:

$$\mu_{A\Delta B}(x)=\begin{cases}\mu_B(x), \text{якщо } \mu_A(x)=1,\\ \mu_A(x), \text{якщо } \mu_B(x)=1,\\ 0, \text{інакше.}\end{cases}$$

Драстичне об'єднання $A\nabla B$:

$$\mu_{A\nabla B}(x)=\begin{cases}\mu_B(x), \text{якщо } \mu_A(x)=0,\\ \mu_A(x), \text{якщо } \mu_B(x)=0,\\ 1, \text{інакше.}\end{cases}$$

Операція λ-суми $A +_\lambda B$: $\mu_{A+_\lambda B}(x) = \lambda\mu_A(x)+(1–\lambda)\mu_B(x)$, де $\lambda\in[0,1]$.

Операція зведення в ступінь a нечіткої множини A записується як A^a, де a – позитивне число, і визначена на основі операції алгебраїчного добутку: $\mu_{A^a}(x)=\mu_A^a(x)$. Окремими випадками зведення в ступінь є:

– CON(A) = A^2 – *операція концентрування* (концентрації, ущільнення – concentration) зменшує ступінь нечіткості, її лінгвістичне значення – «дуже»;
– DIL(A) = $A^{0,5}$ – *операція розтягання* (dilation) – збільшує ступінь нечіткості, її лінгвістичне значення – «приблизно».

Операція інтенсифікації:

$$\mu_{\mathrm{INT}(A)}(x)=\begin{cases}2(\mu_A(x))^2, 0\le\mu_A(x)\le 0{,}5;\\ 1-2(1-\mu_A(x))^2, 0{,}5\le\mu_A(x)\le 1.\end{cases}$$

Множення на число a: приналежності елементів помножаються на число: $\mu_{aA}(x) = a\mu_A(x)$, де a – позитивне число, таке, що $\max\limits_{x\in A}\mu_{aA}(x)\le 1$.

Опукла комбінація нечітких множин $A_1, A_2, ..., A_n$ є узагальненням операції λ-суми: $\mu_A(x_1, x_2, ..., x_n) = \lambda_1\mu_{A_1}(x) + \lambda_2\mu_{A_2}(x) + ... + \lambda_n\mu_{A_n}(x)$, де $A_1, A_2, ..., A_n$ – нечіткі множини універсальної множини E, $\forall x\in E$, а $\lambda_1, \lambda_2, ... \lambda_n$ – ненегативні числа, сума яких дорівнює 1.

Декартовий (прямий) добуток нечітких множин $A = A_1 \times A_2 \times ... \times A_n$, є нечіткою підмножиною множини $E = E_1 \times E_2 \times ... \times E_n$ з функцією приналежності: $\mu_A(x_1, x_2, ..., x_n) = \min\{\mu_{A_1}(x_1), \mu_{A_2}(x_2), ..., \mu_{A_n}(x_n)\}$, де $A_1, A_2, ..., A_n$ – нечіткі підмножини універсальних множин $E_1, E_2, ..., E_n$ відповідно.

Нормалізація непорожньої субнормальної множини: перераховують приналежності елементів за формулою:

$$\mu_A(x) = \frac{\mu_A(x)}{\max\limits_{x \in E} \mu_A(x)}.$$

Нечітка імплікація $A \to B$ визначає причинно-наслідкове відношення між умовами та наслідками правил. Операції нечіткої імплікації можна розділити на три основні класи:

S-імплікація: $\mu_{A\to B}(x) = S\{1-\mu_A(x), 1-\mu_B(x)\}$, де S – оператор S-норми;

R-імплікація: $\mu_{A\to B}(x) = \sup\{z \in [0, 1] | T(\mu_A(x), z) \le \mu_B(x)\}$,

T-імплікація: $\mu_{A\to B}(x) = T\{\mu_A(x), \mu_B(x)\}$, де T – оператор T-норми.

Окремо виділяють нечіткі імплікації, які визначаються:

за Заде: $\mu_{A\to B}(x) = \max\{\min\{\mu_A(x), \mu_B(x)\}, (1-\mu_A(x))\}$,

за Кліне-Дайєнісом: $\mu_{A\to B}(x) = \max\{(1-\mu_A(x)), \mu_B(x)\}$,

за Мамдані: $\mu_{A\to B}(x) = \min\{\mu_A(x), \mu_B(x)\}$,

за Лукасевичем:

$\mu_{A\to B}(x) = \min\{1, 1-\mu_A(x)+\mu_B(x)\}$ або $\mu_{A\to B}(x) = \max\{0, \mu_A(x)+\mu_B(x) - 1\}$,

за Гогуеном: $\mu_{A\to B}(x) = \min\{1, \mu_B(x) / \mu_A(x)\}$, $\mu_A(x) > 0$,

за граничною сумою: $\mu_{A\to B}(x) = \min\{1, \mu_A(x)+\mu_B(x)\}$,

за Ларсеном: $\mu_{A\to B}(x) = \mu_A(x)\mu_B(x)$,

за Віллмоттом: $\mu_{A\to B}(x) = \min\{\max\{1-\mu_A(x), \mu_B(x)\}, \max\{\mu_A(x), 1-\mu_B(x), \min\{1-\mu_A(x), \mu_B(x)\}\}\}$,

за Кліне-Дайєнісом-Лукасевичем-Рейхенбахом:

$$\mu_{A\to B}(x) = 1 - \mu_A(x) + \mu_A(x)\mu_B(x),$$

за Ваді: $\mu_{A\to B}(x) = \max\{\mu_A(x)\mu_B(x), 1 - \mu_A(x)\}$,

за Ягером: $\mu_{A\to B}(x) = \mu_A(x)^{\mu_B(x)}$,

за Гейнсом: $\mu_{A\to B}(x) = \begin{cases} 1, \text{якщо } \mu_A(x) \le \mu_B(x), \\ \mu_B(x)/\mu_A(x), \text{інакше}, \end{cases}$

за Гьоделем: $\mu_{A\to B}(x) = \begin{cases} 1, \text{якщо } \mu_A(x) \le \mu_B(x), \\ \mu_B(x), \text{інакше}, \end{cases}$

за Шарпом (стандартною логікою послідовностей R-SEQ):

$$\mu_{A\to B}(x) = \begin{cases} 1, \text{якщо } \mu_A(x) \le \mu_B(x), \\ 0, \text{інакше}, \end{cases}$$

Нечітке включення: ступінь включення нечіткої множини:

$$V(A, B) = (\mu_A(x_0) \to \mu_B(x_0)) \cap (\mu_A(x_1) \to \mu_B(x_1)) \cap \dots$$

Нечітка еквівалентність $A \equiv B$ визначається за формулою:

$$\mu_{A\equiv B}(x) = \min\{\max\{1-\mu_A(x), \mu_B(x)\}, \max\{\mu_A(x), 1-\mu_B(x)\}\}.$$

Оператор збільшення нечіткості використовується для перетворення чітких множин у нечіткі та для збільшення нечіткості нечіткої множини. Нехай A – нечітка множина, E – універсальна множина і для всіх $x \in E$ визначені нечіткі множини $K(x)$. Сукупність усіх $K(x)$ називається *ядром оператора збільшення нечіткості* H. Результатом дії оператора H на нечітку множину A є нечітка множина виду: $H(A,K) = \bigcup\limits_{x\in E} \mu_A(x)K(x)$, де $\mu_A(x)K(x)$ – добуток числа на нечітку множину.

Узагальнення нечітких операцій

Уведені вище операції над нечіткими множинами засновані на використанні операцій max і min. У теорії нечітких множин розглянуті питання побудови узагальнених, параметризованих операцій перетинання, об'єднання і доповнення, що дозволяють врахувати різноманітні значеннєві відтінки відповідних їм зв'язувань «ТА», «АБО», «НЕ».

Один з підходів до узагальнення операцій перетинання й об'єднання полягає в їхньому визначенні в класі трикутних норм і конорм.

Трикутною нормою (t-нормою) називається двомісна дійсна функція T: $[0, 1]\times[0, 1]\to[0, 1]$, що задовольняє таким умовам:

1) $T(0, 0)=0$; $T(\mu_A, 1) = \mu_A$; $T(1, \mu_A) = \mu_A$ – обмеженість;

2) $T(\mu_A, \mu_B) \le T(\mu_C, \mu_D)$, якщо $\mu_A < \mu_C$, $\mu_B < \mu_D$ – монотонність;

3) $T(\mu_A, \mu_B) = T(\mu_B, \mu_A)$ – комутативність;

4) $T(\mu_A, T(\mu_B, \mu_C))= T(T(\mu_A, \mu_B), \mu_C)$ – асоціативність;

Трикутною конормою (t-конормою) називається двомісна дійсна функція S:$[0, 1]\times[0, 1]\to[0, 1]$, із властивостями:

1) $S(1, 1) = 1$; $S(\mu_A, 0) = \mu_A$; $S(0, \mu_A) = \mu_A$ – обмеженість;

2) $S(\mu_A, \mu_B) \ge S(\mu_C, \mu_D)$, якщо $\mu_A > \mu_C$, $\mu_B > \mu_D$ – монотонність;

3) $S(\mu_A, \mu_B) = S(\mu_B, \mu_A)$ – комутативність;

4) $S(\mu_A, S(\mu_B, \mu_C)) = S(S(\mu_A, \mu_B), \mu_C)$ – асоціативність.

Відомі різні *визначення нечітких операторів*:

за Заде: $T(\mu_A, \mu_B) = \min(\mu_A, \mu_B)$, $S(\mu_A, \mu_B) = \max(\mu_A, \mu_B)$;

імовірнісні: $T(\mu_A, \mu_B) = \mu_A\cdot\mu_B$, $S(\mu_A, \mu_B) = \mu_A+\mu_B-\mu_A\cdot\mu_B$;

$$T(\mu_A,\mu_B) = \frac{\mu_A\mu_B}{\max(\mu_A,\mu_B,\alpha)},\ S(\mu_A,\mu_B) = \frac{\mu_A+\mu_B-\mu_A\mu_B-\min(\mu_A,\mu_B,1-\alpha)}{\max(1-\mu_A,1-\mu_B,\alpha)},\ \alpha\in[0,1];$$

$$T(\mu_A,\mu_B) = \frac{\mu_A\mu_B}{\alpha+(1-\alpha)(\mu_A+\mu_B-\mu_A\mu_B)},\ S(\mu_A,\mu_B) = \frac{\mu_A+\mu_B-(2-\alpha)\mu_A\mu_B}{1-(1-\alpha)\mu_A\mu_B},\ \alpha>0;$$

за Лукасевичем: $T(\mu_A, \mu_B) = \max(0, \mu_A+\mu_B-1)$, $S(\mu_A, \mu_B) = \min(1, \mu_A+\mu_B)$;

$$T(\mu_A,\mu_B) = \max\left(\frac{\mu_A+\mu_B-1+\alpha\mu_A\mu_B}{1+\alpha},0\right),\ S(\mu_A, \mu_B) = \min(1, \mu_A+\mu_B+\alpha\mu_A\mu_B),\ \alpha\ge -1;$$

драстичні: $T(\mu_A,\mu_B)=\begin{cases}\mu_A,\mu_B=1,\\ \mu_B,\mu_A=1,\\ 0,\mu_A,\mu_B\neq 1,\end{cases}$ $S(\mu_A,\mu_B)=\begin{cases}\mu_A,\mu_B=0,\\ \mu_B,\mu_A=0,\\ 1,\mu_A,\mu_B\neq 0;\end{cases}$

$T(\mu_A,\mu_B)=1-\sqrt[\alpha]{(1-\mu_A)^\alpha+(1-\mu_B)^\alpha-(1-\mu_A)^\alpha(1-\mu_B)^\alpha}$, $S(\mu_A,\mu_B)=\sqrt[\alpha]{\mu_A{}^\alpha+\mu_B{}^\alpha-\mu_A{}^\alpha\mu_B{}^\alpha}$, $\alpha>0$;

$T(\mu_A,\mu_B)=\left(1+\sqrt[\alpha]{\left(\frac{1}{\mu_A}-1\right)^\alpha+\left(\frac{1}{\mu_B}-1\right)^\alpha}\right)^{-1}$, $S(\mu_A,\mu_B)=\left(1+\sqrt[\alpha]{\left(\frac{1}{\mu_A}-1\right)^{-\alpha}+\left(\frac{1}{\mu_B}-1\right)^{-\alpha}}\right)^{-1}$, $\alpha>0$;

$T(\mu_A,\mu_B)=\max(1-\sqrt[\alpha]{(1-\mu_A)^\alpha+(1-\mu_B)^\alpha},0)$, $S(\mu_A,\mu_B)=\min(1-\sqrt[\alpha]{\mu_A{}^\alpha+\mu_B{}^\alpha},1)$, $\alpha\geq 1$;

$T(\mu_A,\mu_B)=\log_\alpha\left(1+\frac{(\alpha^{\mu_A}-1)(\alpha^{\mu_B}-1)}{\alpha-1}\right)$, $S(\mu_A,\mu_B)=1-\log_\alpha\left(1+\frac{\alpha^{1-\mu_A}+\alpha^{1-\mu_B}}{\alpha-1}\right)$, $\alpha>0$, $\alpha\neq 1$.

3.7 Нечіткі величини та числа

Нечітка величина – довільна нечітка множина $A=\{<\mu_A(x)|x>\}$, що задана на множині дійсних чисел R.

Нечіткий інтервал – нечітка величина з опуклою функцією приналежності.

Нечітке число – нечітка величина, функція приналежності якої є опуклою та унімодальною. Нечітке число визначається як нечітка множина A на множині дійсних чисел R з функцією приналежності $\mu_A(x)\in[0,1]$, де $x\in R$.

Нечітким нулем називають нечітке число, якщо його мода дорівнює 0.

Позитивним є нечітке число, якщо воно має суворо позитивний носій.

Негативним є нечітке число, якщо воно має суворо негативний носій.

Операції над нечіткими числами A та B з функціями приналежності $\mu_A(x)$ та $\mu_B(y)$, відповідно, в узагальненому вигляді визначаються як: $A\circ B=C=\{z|\mu_C(z)\}$, де $\circ$ – символ операції, $\mu_C(z)$ – функція приналежності результату, що визначається за формулою:

$$\mu_C(z)=\sup_{z=x\circ y}\{\min\{\mu_A(x),\mu_B(y)\}\} \text{ або } \mu_C(z)=\sup_{z=\circ(x,y)}\{\min\{\mu_A(x),\mu_B(y)\}\}.$$

Як операцію замість символу $\circ$ можна задавати: + (додавання), – (віднімання), (множення), / (ділення), min (розширений мінімум), max (розширений максимум).

Нечіткі числа (*L*-*R*)-*типу* – це різновид нечітких чисел спеціального виду, що задаються за визначеними правилами з метою зниження обсягу обчислень при операціях над ними.

Функції приналежності нечітких чисел (*L*-*R*)-*типу* задаються за допомогою незростаючих на множині невід'ємних дійсних чисел функцій дійсної змінної $L(x)$ та $R(x)$, що задовольняють властивостям: $L(-x)=L(x)$; $R(-x)=R(x)$; $L(0)=R(0)$.

3.8 Нечіткі відношення

Нечітке n-арне відношення визначається як нечітка підмножина R на E, що приймає свої значення в M, де $E = E_1 \times E_2 \times ... \times E_n$ – прямий добуток універсальних множин, M – деяка множина приналежностей (наприклад, $M = [0; 1]$).

У випадку $n = 2$ і $M = [0, 1]$, *бінарним нечітким відношенням* R між множинами $X = E_1$ і $Y = E_2$ буде називатися функція R: $(X, Y) \rightarrow [0, 1]$, що ставить у відповідність кожній парі елементів $(x, y) \in X \times Y$ величину $\mu_R(x, y) \in [0; 1]$.

Нечітке відношення на $X \times Y$ записується у вигляді: $x \in X$, $y \in Y$: $x \, R \, y$.

У випадку, коли $X = Y$, тобто X і Y збігаються, нечітке відношення R: $X \times X \rightarrow [0,1]$ називається *нечітким відношенням на множині X*.

Пустим нечітким відношенням $\varnothing$ називають відношення, що не містить жодного *кортежу* – довільного набору впорядкованих елементів.

Повним нечітким відношенням є декартовий добуток універсумів $E = E_1 \times E_2 \times ... \times E_n$.

Способи задавання нечітких відношень використовують такі:

– *у формі списку з явним перерахуванням усіх кортежів нечіткого відношення та відповідних ним значень функції приналежності*: $R = \{(w_1, \mu_R(w_1)), \ldots, (w_r, \mu_R(w_r))\}$, де $w_i = \langle x_1, x_2, \ldots, x_n \rangle$ – i-ий кортеж елементів цього відношення, а r – число кортежів нечіткого відношення R;

– *аналітично* у формі певного математичного виразу для відповідної функції приналежності цього відношення.

Нечітке бінарне відношення може бути подане:

– *графічно* у вигляді певної поверхні або сукупності окремих точок у тривимірному просторі, при цьому вісі абсциси та ординати будуть відповідати універсумам E_1 та E_2, а вісь аплікати – інтервалу [0; 1];

– *у матричній формі:* строки матриці нечіткого відношення при цьому відповідають першим, а стовпці – другим елементам кортежів, елементами матриці є відповідні значення функції приналежності нечіткого відношення;

– *орієнтованим нечітким графом* $G = (V, E, \mu_G)$, що може бути заданий у вигляді двох звичайних скінчених множин: множини вершин нечіткого графа $V = \{v_1, v_2, \ldots, v_n\}$ та множини дуг нечіткого графа $E = \{e_1, e_2, \ldots, e_m\}$, а також певної функції приналежності дуг даному нечіткому графу $\mu_G : E \rightarrow [0;1]$.

Нечіткий предикат $P(\langle x_1, x_2, \ldots, x_n \rangle)$ – деяке відображення з декартового добутку універсумів $E_1 \times E_2 \times ... \times E_n$ у певну цілковито впорядковану множину значень істинності, зокрема, у інтервал [0; 1]. При цьому змінні $x_1, x_2, \ldots, x_n$ називають *предметними змінними* нечіткого предиката, а декартовий добуток універсумів $E_1 \times E_2 \times ... \times E_n$ – *предметною областю* нечіткого предиката.

3.9 Характеристики нечітких відношень

Характеристиками нечітких відношень є: носій нечіткого відношення, відношення α-рівня, висота нечіткого відношення, нормальність, субнормальність, ядро, найближче чітке відношення, межі, точки переходу, опуклість, що визначаються подібно до нечітких множин: при цьому замість x використовують кортеж $<x_1, x_2, \ldots, x_n>$, замість A використовують відношення R, а замість E – декартовий добуток $E_1 \times E_2 \times \ldots \times E_n$.

Модою нечіткого відношення R є кортеж $w_m \in E_1 \times E_2 \times \ldots \times E_n$, якщо цей кортеж є точкою локального максимуму відповідної функції приналежності $\mu_R(<x_1, x_2, \ldots, x_n>)$, тобто виконується умова: $w_m = \arg\max\{\mu_R(<x_1, x_2, \ldots, x_n>)\}$.

Скінченим є нечітке відношення, якщо його носій є скінченим відношенням.

Рефлексивним є нечітке відношення R на $X \times X$, якщо для будь-якого $x \in X$ виконується рівність $\mu_R(x,x) = 1$. У випадку кінцевої множини X всі елементи головної діагоналі матриці R дорівнюють 1.

Антирефлексивним є нечітке відношення R на $X \times X$, якщо для будь-якого $x \in X$ виконується рівність $\mu_R(x,x) = 0$. У випадку кінцевої множини X всі елементи головної діагоналі матриці R дорівнюють 0.

Симетричним є нечітке відношення R на $X \times Y$, якщо для будь-якої пари $(x, y) \in X \times Y$ виконується рівність $\mu_R(x,y) = \mu_R(y,x)$. Матриця симетричного нечіткого відношення, заданого на кінцевій множині, є симетричною.

Асиметричним є нечітке відношення R на $X \times Y$, якщо для будь-якої пари $(x, y) \in X \times Y$ справедливий вираз: $\mu_R(x,y) > 0 \Rightarrow \mu_R(y,x) = 0$.

Зворотними є нечіткі відношення R та R^{-1} на $X \times Y$, якщо для будь-якої пари $(x, y) \in X \times Y$ виконується рівність: $\mu_R(x,y) = \mu_{R^{-1}}(y,x)$.

Транзитивним є нечітке відношення R на $X \times Y$, якщо $R \circ R \subseteq R$. Іншими словами, для будь-якої пари $(x, y) \in X \times Y$ ступінь виконання відношення R повинна бути не менше за ступінь виконання відношення $R \circ R$.

Транзитивним замиканням нечіткого відношення R є відношення:

$$\hat{R} = \bigcup_{n=1,2,\ldots} R^n, \text{ де } R^n = \underbrace{R \circ R \circ \ldots \circ R}_{n \text{ разів}}.$$

3.10 Операції над нечіткими відношеннями

Об'єднання двох відношень $R_1 \cup R_2$ визначається з виразу:

$$\mu_{R_1 \cup R_2}(x,y) = \max(\mu_{R_1}(x,y), \mu_{R_2}(x,y)).$$

Перетинання двох відношень $R_1 \cap R_2$ визначається з виразу:

$$\mu_{R_1 \cap R_2}(x, y) = \min(\mu_{R_1}(x, y), \mu_{R_2}(x, y)).$$

Алгебраїчний добуток двох відношень $R_1 \cdot R_2$ визначається з виразу:

$$\mu_{R_1 \cdot R_2}(x, y) = \mu_{R_1}(x, y) \cdot \mu_{R_2}(x, y).$$

Алгебраїчна сума двох відношень $R_1 \hat{+} R_2$ визначається з виразу:

$$\mu_{R_1 \hat{+} R_2}(x, y) = \mu_{R_1}(x, y) + \mu_{R_2}(x, y) - \mu_{R_1}(x, y) \cdot \mu_{R_2}(x, y).$$

Для введених операцій справедливі такі *властивості дистрибутивності*:

$$R_1 \cap (R_2 \cup R_3) = (R_1 \cap R_2) \cup (R_1 \cap R_3),$$

$$R_1 \cup (R_2 \cap R_3) = (R_1 \cup R_2) \cap (R_1 \cup R_3),$$

$$R_1 \cdot (R_2 \cup R_3) = (R_1 \cdot R_2) \cup (R_1 \cdot R_3),$$

$$R_1 \cdot (R_2 \cap R_3) = (R_1 \cdot R_2) \cap (R_1 \cdot R_3),$$

$$R_1 \hat{+} (R_2 \cup R_3) = (R_1 \hat{+} R_2) \cup (R_1 \hat{+} R_3),$$

$$R_1 \hat{+} (R_2 \cap R_3) = (R_1 \hat{+} R_2) \cap (R_1 \hat{+} R_3).$$

Доповнення відношення R позначається $\overline{R}$ та визначається функцією приналежності: $\mu_{\overline{R}}(x, y) = 1 - \mu_R(x, y)$.

Диз'юнктивна сума двох відношень $R_1 \oplus R_2$ визначається з виразу:

$$R_1 \oplus R_2 = (R_1 \cap \overline{R_2}) \cup (\overline{R_1} \cap R_2).$$

Звичайне відношення, найближче до нечіткого відношення R з функцією приналежності $\mu_R(x, y)$, позначається $\underline{R}$ та визначається з виразу:

$$\mu_{\underline{R}}(x, y) = \begin{cases} 0, \mu_R(x, y) < 0{,}5, \\ 1, \mu_R(x, y) > 0{,}5, \\ 0 \text{ або } 1, \mu_R(x, y) = 0{,}5. \end{cases}$$

За домовленістю приймають: $\mu_R(x, y) = 0$ при $\mu_R(x, y) = 0{,}5$.

Для нечітких відношень подібно до нечітких множин можуть бути визначені й інші операції: включення, рівність, різниця, симетрична різниця, обмежена сума, обмежена різниця, обмежений добуток, драстичне перетинання, драстичне об'єднання, λ-сума, зведення в ступінь, CON, DIL, множення на число, опукла комбінація, нормалізація, нечітке включення. При цьому замість x використовують кортеж $<x_1, x_2, \ldots, x_n>$, замість A використовують відношення R, а замість E – декартовий добуток $E_1 \times E_2 \times \ldots \times E_n$.

(Max-min)-композицією або *(max-min)-згорткою* нечітких відношень R_1: $(X \times Y) \to [0, 1]$ між X і Y та R_2: $(Y \times Z) \to [0, 1]$ між Y та Z, називається $R_1 \bullet R_2$ – нечітке відношення між X і Z, визначене через R_1 та R_2 виразом:

$$\mu_{R_1 \bullet R_2}(x, z) = \max_y [\min(\mu_{R_1}(x, y), \mu_{R_2}(y, z))].$$

Властивості max-min композиції:

– асоціативність: $R_3 \bullet (R_2 \bullet R_1) = (R_3 \bullet R_2) \bullet R_1$;

– дистрибутивність щодо об'єднання, але недистрибутивність щодо перетинання:

$$R_3 \bullet (R_2 \cup R_1) = (R_3 \bullet R_2) \cup (R_3 \bullet R_1), \; R_3 \bullet (R_2 \cap R_1) \neq (R_3 \bullet R_2) \cap (R_3 \bullet R_1);$$

– якщо $R_1 \subset R_2$, то $R_3 \bullet R_1 \subset R_3 \bullet R_2$.

(Max-)-композиція* відношень R_1 та R_2:

$$\mu_{R_1 * R_2}(x, z) = \max_y [\mu_{R_1}(x, y) * \mu_{R_2}(y, z)],$$

де * – будь-яка операція, для якої виконуються ті ж обмеження, що і для min: асоціативність і монотонність (у змісті неубування) за кожним аргументом. Зокрема, операція min може бути замінена алгебраїчним множенням prod – тоді говорять про (*max-prod*)-*композицію*, або операцією максимуму max – тоді говорять про (*max-max*)-*композицію*, або операцією середнього арифметичного average – тоді говорять про (*max-average*)-*композицію*.

(*Min-**)-*композиція* відношень R_1 та R_2:

$$\mu_{R_1 * R_2}(x, z) = \min_y [\mu_{R_1}(x, y) * \mu_{R_2}(y, z)],$$

де * – будь-яка операція, для якої виконуються ті ж обмеження, що і для max: асоціативність і монотонність (у змісті неубування) за кожним аргументом. Зокрема, операцією * може бути операція максимуму max – тоді говорять про (*min-max*)-*композицію*, або операція мінімуму min – тоді говорять про (m*in-min*)-*композицію*.

(*Sum-prod*)-*композиція* відношень R_1 та R_2:

$$\mu_{R_3}(x, z) = f\left(\sum_y (\mu_{R_1}(x, y)\mu_{R_2}(y, z))\right),$$

де *f*() – певна логістична функція типа сигмоїдної, що обмежує значення функції числом з інтервалу [0; 1].

3.11 Нечітке виведення

Нечітке виведення здійснюється на основі нечіткої бази знань.

Нечіткою базою знань (*fuzzy knowledge base*) про вплив факторів $x=\{x_1, x_2, ..., x_n\}$ на значення параметра y називається сукупність логічних висловлень типу:

Якщо $(x_1=a_1^{j1})$ та $(x_2=a_2^{j1})$ та ... та $(x_n=a_n^{j1})$ з вагою w_{j1}
або $(x_1=a_1^{j2})$ та $(x_2=a_2^{j2})$ та ... та $(x_n=a_n^{j2})$ з вагою w_{j2}
. . . . . .
або $(x_1=a_1^{j k_j})$ та $(x_2=a_2^{j k_j})$ та ... та $(x_n=a_n^{j k_j})$ з вагою w_{jk_j}
то $y = d_j$ для всіх $j = 1, 2, ..., m$,

де a_i^{jp} – нечіткий терм, яким оцінюється змінна x_i у рядку з номером jp ($p = 1, 2,...,k_j$); k_j – кількість рядків-кон'юнкцій, у яких вихід у оцінюється нечітким термом d_j, $j = 1, 2 ,..., m$; m – кількість термів, використовуваних для лінгвістичної оцінки вихідного параметра у, w_{jp} – вага p-го рядка кон'юнкцій j-го правила бази знань.

Вага правила – число в інтервалі [0,1], що характеризує суб'єктивну міру впевненості експерта у цьому правилі та використовується для відображення значимості правила.

За допомогою операцій $\cup$ (АБО) і $\cap$ (ТА) нечітку базу знань можна записати у більш компактному вигляді:

$$\bigcup_{p=1}^{k_j}\left[w_{jp} \bigcap_{i=1}^{n}\left(x_i = a_i^{jp}\right)\right] \to y = d_j,\ j = 1, 2,..., m.$$

В *правилах Мамдані*, які мають вигляд: Якщо $x_1 \in A_1$ та $x_2 \in A_2$ та … , то $y \in B$: $\{x_i\}$ – це вхідні змінні, y – вихідна змінна, $\{A_i\}$ та B – нечіткі терми, визначені відповідно для $\{x_i\}$ та y. Зазвичай при прийнятті рішень людина користується такими правилами, що допомагає їй визначитися з подальшими діями в умовах невизначеності.

Правила Такагі-Сугено є іншим типом нечітких правил та мають такий вигляд: Якщо $x_1 \in A_1$ та $x_2 \in A_2$ та … та $x_n \in A_n$, то $y = f(x_1, x_2, \ldots, x_n)$. В правилах такої форми $\{x_i\}$ – це вхідні змінні; y – вихідна змінна; $\{A_i\}$ – нечіткі терми, визначені на $\{x_i\}$, $f(x_1, x_2, \ldots, x_n)$ – лінійна функція, що залежить від вхідних змінних.

Антецедентом правила називають множину нечітких термів, визначених для входів, що є умовою спрацьовування правила.

Консеквентом правила називають множину нечітких термів, визначених для виходів, які будуть присвоєні вихідним змінним при спрацьовуванні правила.

Імплікація – це модифікація нечітких множин виходів за допомогою ступеню виконання правила.

Налагодження параметрів нечіткої бази знань являє собою процес визначення значень параметрів функцій приналежності нечітких термів і ваг правил на основі експериментальних даних.

Нечітка модель типу «вхід-вихід» подається як $y = f(x, w, b)$, де $x = (x_1, x_2, \ldots, x_N)$ – вектор вхідних змінних, f – оператор зв'язку вхід-вихід, $w = (w_1, w_2, \ldots, w_n)$ – вектор ваг правил нечіткої бази знань, $b = \{b_j\}$ – вектор параметрів настроювання функцій приналежності ($j = 1, 2, \ldots, q$), q – загальне число термів у базі знань, λ – загальне число рядків у базі знань: $\lambda = \sum_{j=1}^{m} k_j$.

Об'єкт із неперервним виходом подається навчаючою вибіркою у вигляді S пар експериментальних даних: $\langle x^s, y^s \rangle$, $s = 1, 2, \ldots, S$, $x^s = \{x^s_j\}$ та y^s – вхідний вектор і відповідне значення вихідної змінної y для s-ої пари $\langle x^s, y^s \rangle$, $y^s \in [\underline{y}, \overline{y}]$.

Задача оптимального налагодження нечіткої моделі для об'єкта з неперервним виходом відповідно до методу найменших квадратів може бути сформульована в такий спосіб: знайти вектор (w, b), що задовольняє обмеженням $w_i \in [\underline{w_i}, \overline{w_i}]$, $i = 1, 2, \ldots, \lambda$, $b_j \in [\underline{b}_j, \overline{b}_j]$, $j = 1, 2, \ldots, q$, та забезпечує:

$$E = \sum_{s=1}^{S} (f(x^s, w, b) - y^s)^2 \to \min.$$

Об'єкт із дискретним виходом подається навчаючою вибіркою у вигляді S пар експериментальних даних: $<x^s, y^s>$, $s = 1, 2, \ldots, S$, $x^s = \{x^s_j\}$ та y^s – вхідний вектор і відповідне значення вихідної змінної у для s-ої пари $<x^s, y^s>$, $y^s \in \{d_1, d_2, \ldots, d_m\}$, де d_j – нечіткий терм вихідної змінної.

Задача оптимального настроювання нечіткої моделі для об'єкта з дискретним виходом відповідно до методу найменших квадратів може бути сформульована в такий спосіб: знайти вектор (w, b), що задовольняє обмеженням $w_i \in [\underline{w_i}, \overline{w_i}]$, $i = 1, 2, \ldots, \lambda$, $b_j \in [\underline{b}_j, \overline{b}_j]$, $j = 1, 2, \ldots, q$, та забезпечує:

$$E = \sum_{s=1}^{S} \sum_{j=1}^{m} (\mu_{d_j}(x^s, w, b) - \mu_{d_j}(x^s))^2 \to \min,$$

де $\mu_{d_j}(x^s, w, b)$ – розрахункове значення функції приналежності до j-го нечіткого терму вихідної змінної для s-го екземпляра, $\mu_{d_j}(x^s)$ – фактичне значення функції приналежності до j-го нечіткого терму вихідної змінної для s-го екземпляра:

$$\mu_{d_j}(x^s) = \begin{cases} 1, y^s \in d_j, \\ 0, y^s \notin d_j. \end{cases}$$

Процес та система нечіткого логічного виведення

Нечітким логічним виведенням (fuzzy logic inference) називається апроксимація залежності $y = f(x_1, x_2, \ldots, x_n)$ за допомогою нечіткої бази знань і операцій над нечіткими множинами.

Нехай $\mu_{jp}(x_i)$ – функція приналежності входу x_i нечіткому терму a_i^{jp}, $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$; $p = 1, 2, \ldots, k_j$; $\mu_{d_j}(y)$ – функція приналежності виходу у нечіткому терму d_j, $j = 1, 2, \ldots, m$. Тоді ступінь приналежності конкретного вхідного вектора $x^* = \{x_1^*, x_2^*, \ldots, x_n^*\}$ нечітким термам d_j з бази знань визначається такою системою нечітких логічних рівнянь:

$$\mu_{d_j}(x^*) = \max_{p=1,2,\ldots,k_j} \min_{i=1,2,\ldots,n} \left(\mu_{jp}(x_i^*)\right), \; j = 1, 2, \ldots, m.$$

Нечітка множина y, відповідна вхідному вектору x^*, яка має верхню і нижню межі діапазону значень $\bar{y}$ та $\underline{y}$, відповідно, визначається як:

$$y = \bigcup_{j=1}^{m} \int_{\underline{y}}^{\overline{y}} \min\left(\mu_{d_j}(x^*), \mu_{d_j}(y)\right) dy.$$

Чітке значення виходу y^*, що відповідає вхідному вектору x^*, визначається в результаті дефазифікації нечіткого y.

Знання експерта $A \rightarrow B$ відбиває нечітке причинне відношення передумови і висновку, тому його можна назвати нечітким відношенням і позначити через R: $R = A \rightarrow B$, де «→» називають *нечіткою імплікацією*.

Відношення R можна розглядати як нечітку підмножину прямого добутку $X \times B$ повної множини передумов X і висновків B. Таким чином, процес одержання (нечіткого) результату виведення B' з використанням даного спостереження A' і знання $A \rightarrow B$ можна подати у виді композиційного правила *нечіткий «modus ponens»*: $B' = A' \bullet R = A' \bullet (A \rightarrow B)$, де «•» – операція згортки.

Як операцію композиції, так і операцію імплікації в алгебрі нечітких множин можна реалізовувати по-різному (при цьому буде відрізнятися й одержуваний результат), але в будь-якому випадку *загальне логічне виведення* здійснюється за такі чотири етапи.

1) *Уведення нечіткості* (фазифікація – fuzzification). Функції приналежності, визначені на вхідних змінних, застосовуються до їхніх фактичних значень для визначення ступеня істинності кожної передумови кожного правила.

2) *Логічне виведення*. Обчислене значення істинності для передумов кожного правила застосовується до висновків кожного правила. Це приводить до однієї нечіткої підмножини, що буде призначена кожній змінній виведення для кожного правила. У якості правил логічного виведення звичайно використовуються тільки операції min (мінімум) або prod (множення). У логічному виведенні мінімуму функція приналежності виведення «відтинається» за висотою, що відповідає обчисленню ступеня істинності передумови правила (нечітка логіка «ТА»). У логічному виведенні множення функція приналежності виведення масштабується за допомогою обчисленого ступеня істинності передумови правила.

3) *Композиція*. Усі нечіткі підмножини, призначені до кожної змінної виведення (у всіх правилах), поєднуються разом, щоб сформувати одну нечітку підмножину для всіх змінних виведення. При подібному об'єднанні звичайно використовуються операції max (максимум) або sum (сума). При композиції максимуму комбіноване виведення нечіткої підмножини конструюється як поточковий максимум по всіх нечітких підмножинах (нечітка логіка «АБО»). При композиції суми комбіноване виведення нечіткої підмножини формується як поточкова сума по всіх нечітких підмножинах, призначених змінній виведення правилами логічного виведення.

4) *Приведення до чіткості* (дефазифікація – defuzzification) використовується, якщо потрібно перетворити нечіткий набір виведень у чітке число.

Система нечіткого виведення складається з п'яти функціональних блоків (рис. 3.1):

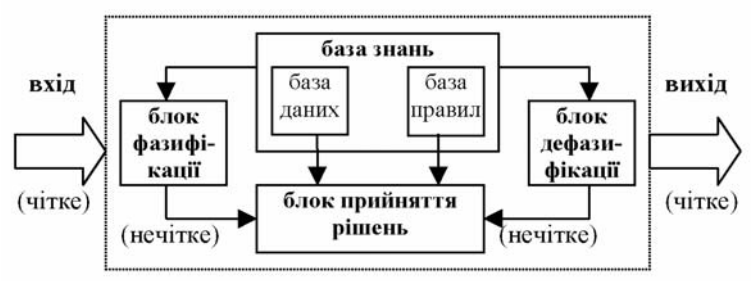


Рисунок 3.1 – Система нечіткого виведення

– *блок фазифікації*, що перетворює чисельні вхідні значення в ступінь відповідності лінгвістичним змінним;

– *база правил*, що містить набір нечітких правил типу якщо-то;

– *база даних*, у якій визначені функції приналежності нечітких множин, що використовуються в нечітких правилах;

– *блок прийняття рішень*, який виконує операції виведення на основі існуючих правил;

– *блок дефазифікації*, що перетворює результати виведення в чисельні значення.

Виділяють три основних *типи систем нечіткого виведення:*

– 1-й тип: вихідне значення знаходиться як зважене середнє результатів виконання кожного правила, для кожного з яких дефазифікація проводиться окремо; для таких систем вихідні функції приналежності повинні бути монотонно-неспадаючими;

– 2-й тип: вихідне нечітке значення – це результат об'єднання нечітких виходів кожного правила; кожний нечіткий вихід зважено за допомогою ваг спрацьовування правил; чітке вихідне значення знаходиться в результаті дефазифікації об'єднаного нечіткого виходу;

– 3-й тип: система, побудована на правилах типа Сугено; вихідне значення є лінійною комбінацією вхідних значень плюс деяке постійне значення, загальний вихід є середнім зваженим всіх правил.

В загальному випадку в якості значень вхідних та вихідних змінних правил можна використовувати нечіткі множини, з якими не пов'язано ніяке поняття – оскільки при проведенні нечіткого виведення нечіткі терми все одно представляються нечіткими множинами і пов'язане з нечітким термом поняття не відіграє ніякої ролі.

Фазифікація та дефазифікація

Фазифікація (fuzzification) – це визначення ступеня виконання антецедентів правил. За допомогою фазифікації чіткому значенню ставляться у відповідність ступені його приналежності до нечітких множин.

Дефазифікація (defuzzification) – процедура перетворення нечіткої множини в чітке число за ступенем приналежності.

У теорії нечітких множин процедура дефазифікації є аналогічною знаходженню характеристик положення (математичного сподівання, моди, медіани) випадкових величин у теорії ймовірностей. Найпростішим способом виконання процедури дефазифікації є вибір чіткого числа, що відповідає максимуму функції приналежності. Однак придатність цього способу обмежується лише одноекстремальними функціями приналежності.

В системах нечіткого виведення функції консеквенту, отримані в результаті виконання правил, об'єднуються в одну функцію $\mu(y)$. Існують різні методи дефазифікації цієї об'єднаної функції приналежності.

Нехай y – нечітка змінна, Y – область визначення змінної y, y^* – чітке значення нечіткої змінної y.

Методи дефазифікації можна записати у такому вигляді:

– *середній з максимальних* (MOM – mean of maximum):

$$y^* = \frac{1}{\left|\mathrm{MAX}(\mu)\right|} \sum_{y \in \mathrm{MAX}(\mu_Y(y))} y,$$

де $\mathrm{MAX}(\mu_Y(y)) = \{y \in Y \mid \forall y' \in Y : \mu(y') \le \mu(y)\}$ – це множина значень вихідної змінної, при яких функція приналежності приймає максимальне значення, ця множина має бути непустою; $\mathrm{MAX}(\mu_Y(y))$ – кількість елементів множини $\mathrm{MAX}(\mu_Y(y))$;

– *найбільший з максимальних* (LOM – largest of maximum):

$$y^* = \max(\mathrm{MAX}(\mu_Y(y)));$$

– *найменший з максимальних* (SOM – smallest of maximum):

$$y^* = \min(\mathrm{MAX}(\mu_Y(y)));$$

– *максимум функції приналежності*:

$$y^* = \arg\sup_{y} \mu_Y(y),$$

де $\mu_Y(y)$ – унімодальна функція;

– *центр тяжіння* (COG – center of gravity, центроїд – centroid):

$$y^* = \frac{\sum_{i=1}^{k} y_i \mu_Y(y_i)}{\sum_{i=1}^{k} \mu_Y(y_i)};$$

де y_i – *i*-й *сінглтон* (одноточкова нечітка множина), $\mu_Y(y_i)$ – значення функції приналежності для *i*-го елемента нечіткої множини *Y*;

– *центр площі* (center of area): чітке значення вихідної змінної y^* визначається з рівняння:

$$\int_{Y_{\min}}^{y^*} \mu_Y(y)dy = \int_{y^*}^{Y_{\max}} \mu_Y(y)dy;$$

– *метод медіани* (bisector):

$$y^* = \min_{\forall j: \sum_{i=1}^{j} \mu_Y(y_i) \geq \frac{1}{2} \sum_{i=1}^{k} \mu_Y(y_i)} (y_i);$$

– *висотна дефазифікація* (height defuzzification):

$$y^* = \frac{\sum_{Y_\alpha} y_i \mu_Y(y_i)}{\sum_{Y_\alpha} \mu_Y(y_i)},$$

де A_α – нечітка множина α-рівня. Елементи нечіткої множини, для котрих значення функції приналежності менше, ніж певний рівень α, до розрахунків не беруться.

Методи нечіткого виведення

Прямий (висхідний) метод нечіткого виведення заснований на використанні нечіткого узагальнення правила modus ponens, який стосовно до систем нечітких продукцій реалізується тим, що окремі факти проблемної області перетворюються у конкретні значення функцій приналежності умов нечітких продукцій. Після чого за одним з методів нечіткої композиції знаходяться значення функцій приналежності висновків правих частин за кожним з правил нечітких продукцій. Ці значення функцій приналежності або є шуканим результатом виведення, або можуть бути використані як додаткові умови у базі правил продукцій, що розглядається.

Розглянемо найбільш вживані методи нечіткого виведення.

Метод Мамдані (E. Mamdani) використовує базу знань із правилами у Мамдані та передбачає виконання таких дій:

1) Уведення нечіткості. Знаходяться ступені істинності для передумов кожного правила: $\mu_{jp}(x_i^*)$, $i = 1, 2, ..., n$; $j = 1, 2, ..., m$; $p = 1, 2, ..., k_j$.

2) Логічне виведення. Знаходяться рівні «відтинання» для передумов кожного з правил (з використанням операції мінімум):

$$\mu_j(y) = \min_{i,p} \mu_{jp}(x_i^*), \quad i = 1, 2, ..., n; \; j = 1, 2, ..., m; \; p = 1, 2, ..., k_j.$$

Потім знаходяться «усічені» функції приналежності:

$$\mu_j'(y) = \min(\mu_j(y), \mu_{d_j}(y)),\ j = 1, 2, ..., m.$$

3) Композиція. Здійснюється об'єднання знайдених усічених функцій з використанням операції максимум, що приводить до одержання підсумкової нечіткої підмножини для змінної виходу з функцією приналежності:

$$\mu(y) = \max_j(\mu_j'(y)),\ j = 1, 2, ..., m.$$

4) Приведення до чіткості – проводиться для отримання y^*, наприклад, центроїдним методом.

Метод Цукамото (Y. Tsukamoto): Вихідні посилки – як у попереднього методу, але тут передбачається, що функції $\mu_{d_j}(y)$ є монотонними.

1) Уведення нечіткості. Знаходяться ступені істинності для передумов кожного правила: $\mu_{jp}(x_i^*)$, $i = 1, 2, ..., n$; $j = 1, 2, ..., m$; $p = 1, 2, ..., k_j$.

2) Нечітке виведення. Знаходяться рівні «відтинання» для передумов кожного з правил:

$$\mu_j(y) = \min_{i,p} \mu_{jp}(x_i^*),\ i = 1, 2, ..., n; j = 1, 2, ..., m; p = 1, 2, ..., k_j,$$

а потім для кожного вихідного правила визначаються чіткі значення y_j шляхом розв'язку рівнянь: $\mu_j(y) = \mu_{d_j}(y_j)$, $j = 1, 2, ..., m$.

3) Визначається чітке значення змінної y^* на основі центроїдного методу.

Метод Ларсена (H. Larsen)*:* нечітка імплікація моделюється з використанням оператора множення.

1) Уведення нечіткості. Знаходяться ступені істинності для передумов кожного правила: $\mu_{jp}(x_i^*)$, $i = 1, 2, ..., n$; $j = 1, 2, ..., m$; $p = 1, 2, ..., k_j$.

2) Нечітке виведення. Знаходяться рівні «відтинання» для передумов кожного з правил:

$$\mu_j(y) = \min_{i,p} \mu_{jp}(x_i^*),\ i = 1, 2, ..., n; j = 1, 2, ..., m; p = 1, 2, ..., k_j,$$

а потім визначаються часткові нечіткі підмножини:

$$\mu_j(y)\mu_{d_j}(y),\ j = 1, 2, ..., m.$$

3) Знаходиться підсумкова нечітка підмножина:

$$\mu(y) = \max_j(\mu_j(y)\mu_{d_j}(y)),\ j = 1, 2, ..., m.$$

4) При необхідності здійснюється приведення до чіткості.

Спрощений метод нечіткого виведення: Вихідні правила в даному випадку задаються у виді: Якщо $(x_1 = a_1^{j1})$ та $(x_2 = a_2^{j1})$ та ... та $(x_n = a_n^{j1})$, то $y = d_j$ для всіх $j = 1, 2, ..., m$. Тут d_j – нечіткі числа.

1) Уведення нечіткості. Знаходяться ступені істинності для передумов кожного правила: $\mu_{jp}(x_i^*)$, $i = 1, 2, ..., n$; $j = 1, 2, ..., m$; $p = 1, 2, ..., k_j$.

2) Нечітке виведення. Знаходяться числа

$$\mu_j(y) = \min_{i,p} \mu_{jp}(x_i^*),\ i = 1, 2, ..., n;\ j = 1, 2, ..., m;\ p = 1, 2, ..., k.$$

3) Визначається чітке значення вихідної змінної y^* для нечіткої множини $y = \{\mu_j(y)|d_j\}$ на основі центроїдного методу.

Метод Сугено: М. Сугено (M. Sugeno) та Т. Такагі (T. Takagi) використовували набір правил у формі: Якщо $(x_1=a_1^{j1})$ та $(x_2=a_2^{j1})$ та ... та $(x_n=a_n^{j1})$, то $y = w_1x_1+w_2x_2+...w_ix_i+...+w_nx_n$, для всіх $i = 1, 2, ..., n$; $j = 1, 2, ..., m$. Тут w_i – деякі вагові коефіцієнти.

1) Уведення нечіткості. Знаходяться ступені істинності для передумов кожного правила: $\mu_{ip}(x_i^*)$, $i = 1, 2, ..., n$; $j = 1, 2, ..., m$; $p = 1, 2, ..., k_j$.

2) Нечітке виведення. Знаходяться рівні «відтинання» для передумов кожного з правил:

$$\mu_j(y) = \min_{i,p} \mu_{jp}(x_i^*),\ i = 1, 2, ..., n;\ j = 1, 2, ..., m;\ p = 1, 2, ..., k_j,$$

а також індивідуальні виходи правил: $d_j = w_1x_1 + w_2x_2 + ...w_ix_i +...+ w_nx_n$.

3) Визначається чітке значення змінної виведення y^* для нечіткої множини $y = \{\mu_j(y)|d_j\}$ на основі центроїдного методу.

Для налагодження параметрів баз знань Сугено запропоновано такі методи.

Метод налагодження параметрів нечіткої бази знань Сугено на основі фільтра Калмана передбачає знаходження таких коефіцієнтів у висновках правил, що забезпечують мінімальне відхилення між даними спостережень і результатами логічного виведення за нечіткою базою знань Сугено.

Нехай ми маємо навчаючу вибірку пар $<x^s, y^s>$, $x^s = \{x^s{}_i\}$, $s = 1, 2, \ldots, S$, $i = 1, 2, \ldots, N$, де $x^s{}_i$ – значення i-ої ознаки s-го екземпляра, y^s – значення цільової ознаки s-го екземпляра, а також правила нечіткої бази знань Сугено:

$$\bigcup_{p=1}^{k_j}\left[w_{jp} \bigcap_{i=1}^{N}\left(x_i = a_i^{jp}\right)\right] \to y = \sum_{i=1}^{N} b_{j,i}x_i,\ j = 1,2,...,m.$$

Необхідно знайти такі значення коефіцієнтів висновків правил $b=\{b_{j,i}\}$, $j = 1, 2, \ldots, m$, $i = 1, 2, \ldots, N$, які забезпечують мінімізацію помилки:

$$E = \frac{1}{2}\sum_{s=1}^{S}\left(y^s - y^{s*}\right)^2,$$

де y^{s*} – результат виведення за нечіткою базою знань з параметрами b для s-го екземпляра вибірки.

Екземпляру x^s відповідає результат нечіткого виведення:

$$y^{s*} = \sum_{j=1}^{m} \beta_j^s d_j = \sum_{j=1}^{m} \beta_j^s \left(b_{j,0} + \sum_{i=1}^{N} b_{j,i}x_i^s \right),$$

$$\beta_{j,s} = \frac{\mu_{d_j}\left(x^s\right)}{\sum\limits_{k=1}^{m} \mu_{d_k}\left(x^s\right)},\ \mu_{d_j}\left(x^s\right) = \bigcup_{p=1}^{k_j}\left[w_{jp} \bigcap_{i=1}^{N} \mu_{jp}\left(x_i^s\right)\right],\ j = 1,2,...,m.$$

Позначимо:

$$y = \{y^s\},\ y^* = \{y^{s*}\},\ A = \{A_{s,k}\},$$

$$A_{s,k} = \begin{cases} \beta_k^s,\ s = 1, 2, ..., S, k = 1, 2, ..., m, \\ x_i^s \beta_j^s,\ k = m + i \cdot j, i = 1, 2, ..., N,\ j = 1, 2, ..., m,\ s = 1, 2, ..., S. \end{cases}$$

Тоді в матричній формі задача ставиться таким чином: знайти такий вектор b, щоб: $E = (y - y^*)^T (y - y^*) \rightarrow \min$, де $y^* = Ab$.

Мінімальне значення E досягається при $y^* = y$, що відповідає розв'язку рівняння $y = Ab$. Для реальних задач кількість параметрів, що налагоджуються, менше обсягу вибірки даних $m(N+1) < S$, тому рівняння $y = Ab$ не має точного розв'язку. У цьому випадку розв'язок можна знайти, використовуючи псевдоінверсію матриці A: $b = (A^T A)^{-1} A^T y$. Проблеми знаходження b пов'язані з можливою сингулярністю матриці $(A^T A)$.

Для знаходження b за допомогою фільтра Калмана можна використовувати такі ітеративні формули:

$$b_{k+1} = b_k + w_{k+1} A_{k+1} (y^{k+1} - (A_{k+1})^T b_k,$$

$$w_{k+1} = w_k - \frac{w_k A_{k+1} (A_{k+1})^T w_k}{1 + (A_{k+1})^T w_k A_{k+1}},\ k = 0, 1, \ldots, S-1,$$

при початкових умовах: $b_0 = 0$ та $w_0 = w$, де w – велике позитивне число, I – одинична матриця; k – номер ітерації.

Кількість ітерацій методу ідентифікації дорівнює обсягу вибірки експериментальних даних. На першому кроці методу координати вектора b налагоджуються за першим екземпляром вибірки. На s-ій ітерації координати вектора b налагоджуються за s-им екземпляром.

Метод налагодження параметрів нечіткої бази знань Сугено на основі градієнтного методу оптимізації подамо такою послідовністю кроків.

Крок 1. Задати навчаючу вибірку $<x, y>$, $x = \{x^s\}$, $x^s = \{x^s_i\}$, $y = \{y^s\}$, де s – номер екземпляра вибірки, i – номер ознаки, $s = 1, 2, \ldots, S$, $i = 1, 2, \ldots, N$. Установити параметри методу: $E_{\max}$ – максимально припустиму помилку та Epochs – максимальну кількість епох навчання.

Крок 2. Розрахувати для кожного екземпляра навчаючої вибірки відносні ступені виконання висновків правил β^s_j, $s = 1, 2, \ldots, S$, j = 1, 2, …, m.

Крок 3. Установити лічильник ітерацій навчання і лічильник епох навчання: $s = 1$, epoch = 1.

Крок 4. Установити початкові значення параметрів, що налагоджуються: $b^s_{j,i} = 0$, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, m$.

Крок 5. Розрахувати значення «миттєвої помилки» для *r*-ої пари даних з вибірки і перерахувати значення параметрів, що налагоджуються:

$$b_{j,i}^{s+1} = b_{j,i}^{s} - \alpha \frac{\partial e_s}{\partial b_{j,i}^{s}},\ \frac{\partial e_s}{\partial b_{j,i}^{s}} = 2\sqrt{e_s} b_{j,i}^{s} x_i^{s},$$

$$i = 1, 2, \ldots, N, j = 1, 2, \ldots, m,$$

де $e_s=(y^s-y^{s*})^2$ – миттєва помилка для *s*-го екземпляра вибірки, $\alpha > 0$ – крок, що задає швидкість навчання. При малих значеннях параметра α навчання буде повільним. При великих значеннях цього параметра виникає небезпека перенавчання, коли на кожній ітерації методу нечітка модель буде налагоджуватися тільки на поточну пару «входи – вихід», при цьому забуваючи попередній досвід.

Крок 6. Якщо $s < S$, тоді збільшити лічильник ітерацій $s = s+1$ і перейти до кроку 5.

Крок 7. Розрахувати значення помилки для усієї вибірки даних на епосі навчання epoch:

$$E_{\text{epoch}} = \frac{1}{2} \sum_{s=1}^{S} e_s .$$

Крок 8. Якщо $E_{\text{epoch}} < E_{\max}$, тоді перейти до кроку 10.

Крок 9. Якщо epoch < Epochs, тоді установити: epoch = epoch+1, $s = 1$, та перейти до кроку 5.

Крок 10. Зупинення.

У наведеному методі використовується два критерії зупинення: перший – за досягненням припустимої помилки, другий – за перевищенням заданої кількості епох навчання. Протягом однієї епохи здійснюється ітераційне підстроювання параметрів за кожною парою «входи – вихід». Таким чином, крок 5 методу виконується рівно *S* разів. Навчання можна також припинити, якщо за одну епоху параметри, що налагоджуються, практично не змінюються:

$$\max_{\substack{j=1,2,\ldots,m,\\ i=1,2,\ldots,N}} | b_{j,i}^{s+1} - b_{j,i}^{s} | < \Delta b ,$$

або помилка майже не зменшується: $E_z - E_{z+1} < \Delta E$, де Δb та ΔE – малі величини.

Раніше розглянуті нечіткі логічні виведення є *висхідними виведеннями* від передумов до висновків. У діагностичних нечітких системах часто застосовуються спадні виведення.

Зворотний (спадний) метод нечіткого логічного виведення заснований на використанні нечіткого узагальнення правила виведення modus tollens.

Нехай заданий повний простір передумов $X = \{x_1, ..., x_m\}$ і повний простір висновків $Y = \{y_1, ..., y_n\}$. Між x_i та y_i існують нечіткі причинні відношення $x_i \rightarrow y_j$, які можна подати у вигляді певної матриці R з елементами $r_{ij} \in [0, 1]$,

i = 1, ..., m; j = 1, ..., n. Передумови і висновки можна розглядати як нечіткі множини A та B на просторах X та Y, відношення яких можна подати у вигляді: $B = A \bullet R$, де «•» позначає правило композиції нечітких виведень, наприклад, (max-min)-композицію. У даному випадку напрямок виведень є зворотним для правил, тобто задана матриця R (знання експерта), спостерігаються виходи B (висновки) і визначаються входи A (передумови).

3.12 Нечітка кластеризація

Кластерний аналіз (кластеризація) – це технологія, що дозволяє розподілити вхідні дані на *класи* – групи однотипних екземплярів вибірки, або *кластери* – компактні області групування екземплярів вибірки у просторі ознак.

Вихідною інформацією для кластеризації є вибірка спостережень $x=\{x^s_j\}$, де x^s_j – значення j-ої ознаки s-го екземпляра вибірки, s = 1, 2, …,S; j=1, 2, …, N, S – кількість екземплярів вибірки, N – кількість ознак, що характеризують екземпляри вибірки.

Задача кластеризації полягає в розбитті об'єктів з x на декілька кластерів, у яких об'єкти більш схожі між собою, ніж з об'єктами інших кластерів. У метричному просторі «схожість» звичайно визначають через відстань.

Методи кластеризації можна класифікувати на чіткі та нечіткі. *Чіткі методи кластеризації* розбивають вихідну множину об'єктів x на декілька непересічних підмножин. При цьому будь-який об'єкт із x належить тільки одному кластеру. *Нечіткі методи кластерного аналізу* дозволяють будь-якому екземпляру одночасно належати до всіх визначених кластерів, але з різним ступенем.

Нечіткий кластерний аналіз використовується при побудові нейронечітких систем для визначення нечітких множин, якщо вони невідомі апріорі. Нечіткі множини знаходяться як проекції кластерів на кожну розмірність. Можливо поєднувати апріорні знання з кластерним аналізом, використовуючи його для уточнення параметрів функції приналежності. Недоліком такого методу визначення нечітких множин є складність їхньої інтерпретації.

Більшість методів нечіткої кластеризації спрямовані на мінімізацію суми:

$$J(x,u,C)=\sum_{s=1}^{S}\sum_{v=1}^{V}\left(\left(u_v^s\right)^m d^2\left(x^s,C^v\right)\right)$$

при виконанні умов:

$$V>1,\ \sum_{s=1}^{S}u_v^s>0\,,\ \sum_{v=1}^{V}u_v^s=1,$$

де S – кількість екземплярів, N – кількість параметрів, що описують один екземпляр (або кластер), V – кількість кластерів; $x=(x^1, x^2, ..., x^S)^T$ – це матриця входів для екземплярів навчаючої вибірки, $x^s = (x^s_1, x^s_2, ..., x^s_N)$ – входи s-го екземпляра, s=1,2,...,S, $u = (u^1, u^2, ..., u^S)^T$ – матриця приналежностей екземплярів до кожного з кластерів, $u^s = (u^s_1, u^s_2, ..., u^s_V)$ – вектор приналежностей s-го ек-

земпляра до кожного з кластерів, $u^s_v \in [0,1]$, $C=(C^1,C^2,...,C^V)^T$ – матриця центрів кластерів, $C^v=(C^v_1,C^v_2,...,C^v_N)$ – центр v-го кластера, $v = 1, 2, ..., V$, $m > 1$ – ступінь нечіткості отриманого розподілу (зазвичай обирається рівним 2), $d(x^s,C^v)$ – відстань між s-м екземпляром та центром v-го кластера.

Координати центрів кластерів визначають за формулою:

$$C^v_j = \frac{\sum\limits_{s=1}^{S}\left(u^s_v\right)^m x^s_j}{\sum\limits_{s=1}^{S}\left(u^s_v\right)^m}.$$

Найбільш простим є метод, в якому відстань між екземпляром та кластером знаходиться як *евклідова відстань:*

$$d(x^s,C^v)=\sqrt{\sum_{j=1}^{N}\left(x^s_j-C^v_j\right)^2}.$$

Такий метод шукає кластери як сфери однакового розміру.

Більш складні методи кластеризації шукають кластери як гіпереліпсоїди різного розміру. Такі методи називають *частковими*, вони не можуть вірно опрацьовувати шуми та викиди і віднаходити кластери з неопуклими поверхнями. Для проведення кластерного аналізу за допомогою часткового методу необхідно задати його параметри: діапазон значень змінних, кількість кластерів для кожної із змінних (або їх ширину), функцію приналежності, що описує кластери та інші параметри в залежності від обраного методу кластеризації.

За допомогою *ієрархічних методів* можна віднайти кластери, об'єднуючи менші кластери та розподіляючи більші. Таким чином знаходиться дерево кластерів, на різних рівнях якого можна отримати різне розподілення на кластери.

Щільнісні методи та *сіткові методи* дозволяють розподіляти на кластери різного розміру довільно розподілені екземпляри. Вони також добре впізнають шуми та викиди, але потребують ретельного вибору параметрів, необхідних для реалізації методу.

Метод FCM (Fuzzy c-means – нечітких с-середніх), заснований на ідеях Дж. Дана (J. Dunn) та Дж. Беждека (J. Bezdek), для вирішення задачі нечіткої кластеризації має ітеративний характер послідовного поліпшення певного вихідного нечіткого розбиття $R(A) = \{A_v | A_v \subseteq A\}$, що задається користувачем або формується автоматично за певним евристичним правилом. На кожній з ітерацій рекурентно перераховуються значення функцій приналежності нечітких кластерів та їхніх типових представників.

Метод FCM закінчить роботу у випадку, коли відбудеться виконання заданого апріорі деякого кінцевого числа ітерацій, або коли мінімальна абсо-

лютна різниця між значеннями функцій приналежності на двох послідовних ітераціях не стане менше деякого апріорі заданого значення.

Формально метод FCM визначається у формі ітеративного виконання такої послідовності кроків.

Крок 1. Попередньо необхідно задати такі значення: кількість шуканих нечітких кластерів V ($V > 1$), максимальну кількість ітерацій методу Epochs, параметр збіжності методу ε, а також експонентну вагу розрахунку цільової функції і центрів кластерів m (як правило, $m = 2$). Як поточне нечітке розбиття на першій ітерації методу для вибірки даних x задати деяке вихідне нечітке розбиття $R(A) = \{A_v | A_v \subseteq A\}$ на V непорожніх нечітких кластерів, що описуються сукупністю функцій приналежності $u^s{}_v$, $s = 1, 2, ..., S$; $v = 1, 2, ..., V$. Нечітке розбиття отримують шляхом генерації випадковим чином елементів $u^s{}_v$, що задовольняють умовам цільової функції.

Крок 2. Для вихідного поточного нечіткого розбиття $R(A)=\{A_v | A_v \subseteq A\}$, розрахувати центри нечітких кластерів C^v_j, $v = 1, 2, ..., V$; $j = 1, 2, ..., N$, та значення цільової функції $J(x,u,C)$. Кількість виконаних ітерацій покласти рівною 1.

Крок 3. Сформувати нове нечітке розбиття $R'(A)=\{A_v | A_v \subseteq A\}$ вихідної множини об'єктів кластеризації A на V непорожніх нечітких кластери, що характеризуються сукупністю функцій приналежності $u^{s\prime}{}_v$, $v = 1, 2, ..., V$, $x^s \in A$, які визначаються за формулою:

$$u^{s\prime}_v = \left(\sum_{g=1}^{V} \left(\frac{d(x^s, C^v)}{d(x^s, C^g)} \right)^{\frac{2}{m-1}} \right)^{-1}, \quad v = 1,2,...,V.$$

Крок 4. При цьому якщо для деякого v та деякого x^s значення $d(x^s,C^v)=0$, то для відповідного нечіткого кластера встановимо $u^{s\prime}{}_v = 1$, а для інших кластерів $u^{h\prime}{}_g = 0$, $g = 1, 2, ..., V$; $g \neq v$; $h = 1, 2, ..., S$. Якщо ж таких v для деякого x^s виявиться декілька, тобто для них значення $d(x^s, C^v) = 0$, то евристично для меншого з v встановимо $u^{s\prime}{}_v = 1$, а для інших встановимо $u^{h\prime}{}_g = 0$, $g = 1, 2, ..., V$; $g \neq v$; $h = 1, 2, ..., S$.

Крок 5. Для нового нечіткого розбиття $R'(A)=\{A_v | A_v \subseteq A\}$ розрахувати центри нечітких кластерів C^v_j та значення цільової функції $J'(x,u,C)$.

Крок 6. Якщо кількість виконаних ітерацій перевищує задане число Epochs або ж модуль різниці $|J(x,u,C)–J'(x,u,C)| \leq \varepsilon$, то за шуканий результат нечіткої кластеризації прийняти нечітке розбиття $R'(A)$ і закінчити виконання методу. У протилежному випадку вважати поточним нечітким розбиттям $R(A) = R'(A)$ і перейти до кроку 3 методу, збільшивши на 1 кількість виконаних ітерацій.

Вибір кількості кластерів V є однією з найважливіших проблем у розглянутому методі. Правильно вибрати кількість кластерів для реальних задач

без будь-якої апріорної інформації про структуру даних досить складно. Існує два формальних підходи до вибору кількості кластерів.

Перший підхід заснований на *критерії компактності та роздільності* отриманих кластерів. Логічно припустити, що при правильному виборі кількості кластерів дані будуть розбиті на компактні і добре віддільні одна від іншої групи. У іншому випадку, кластери, імовірно, не будуть компактними і добре віддільними.

Існує кілька критеріїв оцінки компактності кластерів, однак питання про те, як формально і вірогідно визначити правильність вибору кількості кластерів для довільного набору даних залишається відкритим. Для методу FCM рекомендується використовувати *індекс Хіє-Бені* (Xie-Beni index):

$$\chi = \frac{\sum_{v=1}^{V}\sum_{s=1}^{S}\left(u_v^s\right)^m d^2(x^s, C^v)}{S \min d^2(x^s, C^v)}.$$

Другий підхід заснований на *редукції кількості кластерів* і пропонує починати кластеризацію при досить великій кількості кластерів, а потім послідовно поєднувати схожі суміжні кластери. При цьому використовуються різні формальні критерії схожості кластерів.

Експонентна вага m є також важливим параметром методу FCM. Чим більше m, тим кінцева матриця нечіткого розбиття $\{u^s_v\}$ стає більш «розмазаною», та при $m \to \infty$ її елементи: $u^s_v = V^{-1}$, що є дуже поганим рішенням, тому що всі об'єкти належать до всіх кластерів з одним і тим же ступенем. Крім того, експонентна вага дозволяє при формуванні координат центрів кластерів підсилити вплив об'єктів з великими значеннями ступенів приналежності й зменшити вплив об'єктів з малими значеннями ступенів приналежності. На сьогодні не існує теоретично обґрунтованого правила вибору значення експонентної ваги. Звичайно встановлюють: $m = 2$.

Вибір норми для визначення близькості є ще однією проблемою для методів кластер-аналізу. У базовому методі FCM відстань між об'єктом $x^s = \{x^s_j\}$ і центром кластера $C^v = \{C^v_j\}$, $j = 1, 2, \ldots, N$, розраховується через стандартну Евклідову норму. У кластерному аналізі застосовуються й інші норми, серед яких часто використовується діагональна норма і норма Махалонобіса.

Норму в загальному виді можна задати через симетричну позитивно визначену матрицю $B = \{B_{i,j}\}$, $i, j = 1, 2, \ldots, N$, як:

$$d^2(x^s, C^v)_B = (x^s - C^v) \cdot B \cdot (x^s - C^v)^T,$$

де x^s та C^v – вектори з координатами екземпляра та центра кластера, ^{T} – операція транспонування.

Евклідова норма дозволяє виділяти кластери у вигляді гіперсфер. Для Евклідової норми матриця B являє собою одиничну матрицю:

$$B_{i,j} = \begin{cases} 1, i = j; \\ 0, i \neq j. \end{cases}$$

Діагональна норма дозволяє виділяти кластери у вигляді гіпереліпсоїдів, орієнтованих уздовж координатних осей. Для діагональної норми матриця B задається у такий спосіб:

$$B_{i,j} = \begin{cases} w_i, i = j; \\ 0, i \neq j, \end{cases}$$

де w_i – елементи головної діагоналі матриці, що інтерпретуються як ваги координат.

Норма Махаланобіса дозволяє виділяти кластери у вигляді гіпереліпсоїдів, вісі яких можуть бути орієнтовані в довільних напрямках. Для норми Махалонобіса матриця B розраховується через коваріаційну матрицю від x:

$$B = G^{-1},\ G = \frac{1}{S}\sum_{s=1}^{S}\left(x^s - \bar{x}\right)^T\left(x^s - \bar{x}\right),\ \bar{x} = \frac{1}{S}\sum_{s=1}^{S}x^s,$$

де G – коваріаційна матриця; $\bar{x}$ – вектор середніх значень даних.

У результаті застосування методів кластеризації з фіксованою нормою форма всіх кластерів виходить однаковою. Методи кластеризації ніби нав'язують даним невластиву їм структуру, що приводить не тільки до неоптимальних, але й іноді до принципово неправильних результатів. Для усунення цього недоліку запропоновано декілька методів, серед яких виділимо метод Густавсона-Кесселя.

Метод Густавсона-Кесселя (Gustafson-Kessel method) використовує адаптивну норму для кожного кластера, тобто для кожного v-го кластера існує своя норм-породжуюча матриця B^v. У цьому методі при кластеризації оптимізуються не тільки координати центрів кластерів і матриця нечіткого розбитя, але також і норм-породжуючі матриці для всіх кластерів. Це дозволяє виділяти кластери різної геометричної форми. Критерій оптимальності $J(x,u,C)$ є лінійним відносно B^v, тому для одержання ненульових рішень уводять певні обмеження на норм-породжуючі матриці. В методі Густавсона-Кесселя це обмеження на значення визначника норм-породжуючих матриць: $\det(B^v) > 0$, $v = 1, 2, \ldots, V$.

Метод Густавсона-Кесселя може бути поданий таким чином.

Крок 1. Зробити початкове розміщення центрів кластерів у просторі ознак. Ця ініціалізація може бути випадковою або заснованою на результатах пікового або різницевого групування даних. Створити елементарну форму масштабуючої матриці B^v.

Крок 2. Сформувати матрицю коефіцієнтів приналежності усіх векторів x^s, $s = 1, 2, ..., S$, до центрів C^v, $v = 1, 2, ..., V$, шляхом розрахунку u^s_v:

$$u_v^s = \left(\sum_{g=1}^{V} \left(\frac{d(x^s, C^v)_B}{d(x^s, C^g)_B} \right)^{\frac{2}{m-1}} \right)^{-1}, \; v = 1, 2, ..., V.$$

Якщо $\exists k: \; d(x^k, C^v) = 0$, тоді прийняти: $u^k{}_v = 1$, $u^s{}_v = 0$, $s = 1, 2, \ldots, S$, $s \neq k$, $v = 1, 2, \ldots, V$.

Крок 3. Розрахувати нові координати центрів кластерів:

$$C_j^v = \frac{\sum_{s=1}^{S} \left(u_v^s\right)^m x_j^s}{\sum_{s=1}^{S} \left(u_v^s\right)^m} \; .$$

Крок 4. Знайти для кожного центра кластера матрицю коваріації Φ^v:

$$\Phi^v = \sum_{s=1}^{S} \left(u_v^s\right)^m \left(x^s - C^v\right)\left(x^s - C^v\right)^T, \; v = 1, 2, \ldots, V.$$

Крок 5. Для всіх $v = 1, 2, \ldots, V$, розрахувати нову масштабуючу матрицю $B^v = \sqrt[N]{\det(\Phi^v)}(\Phi^v)^{-1}$, де N – кількість ознак, що характеризують екземпляри навчаючої вибірки.

Крок 6. Якщо останні зміни положень центрів кластерів і матриці коваріації є досить малими відносно попереднього значення і не перевищують попередньо заданої граничної величини ε, тоді завершити ітераційний процес; у противному випадку – перейти до кроку 2.

Метод Густавсона-Кесселя має значно більшу обчислювальну трудомісткість у порівнянні з методом FCM.

Результати нечіткого кластер-аналізу можна використовувати для *синтезу нечітких правил*. Кожен кластер буде являти собою деяке нечітке правило, що узагальнює підмножину екземплярів навчаючої вибірки, найбільш тісно розташованих у просторі ознак. Функції приналежності термів у посилках правила отримують проектуванням ступенів приналежності відповідного кластера на вхідні змінні. Потім отримані множини ступенів приналежностей апроксимують придатними параметричними функціями приналежності. Як висновок правила сінглтонної бази знань вибирають координату центра кластера. Висновки правил бази знань Мамдані знаходять також як і функції приналежності термів вхідних змінних. Висновки правил бази знань Сугено знаходять за методом найменших квадратів. При кластеризації з використанням норми Махалонобіса як висновки правил типу Сугено можуть бути обрані рівняння довгих осей гіпереліпсоїдів.

Метод пікового групування (гірської кластеризації), запропонований Р. Ягером та Д. Фільовим, не вимагає задавання кількості кластерів. Кластe-

ризація гірським методом не є нечіткою, однак, її часто використовують при синтезі нечітких правил з даних.

Ідея методу полягає в тому, що спочатку визначають точки, які можуть бути центрами кластерів. Далі для кожної такої точки розраховується значення потенціалу, що показує можливість формування кластера в її околиці. Чим щільніше розташовані об'єкти в околиці потенційного центра кластера, тим вище значення його потенціалу. Після цього ітераційно вибираються центри кластерів серед точок з максимальними потенціалами. Метод гірської кластеризації можна записати як послідовність таких кроків.

Крок 1. Сформувати потенційні центри кластерів, число яких Q повинно бути кінцевим. Центрами кластерів можуть бути об'єкти кластеризації – екземпляри вибірки x, тоді $Q = S$, де S – кількість екземплярів у вибірці x. Другий спосіб вибору потенційних центрів кластерів полягає в дискретизації простору вхідних ознак. Для цього діапазони зміни вхідних ознак розбивають на кілька інтервалів. Проводячи через точки розбиття прямі, паралельні координатним осям, одержуємо «ґратовий» гіперкуб. Вузли цих ґрат і будуть відповідати центрам потенційних кластерів. Позначимо через q_r – кількість значень, що можуть приймати центри кластерів за r-ою координатою, $r = 1, 2, \ldots, N$. Тоді кількість можливих кластерів буде дорівнювати: $Q = \prod_{r=1}^{N} q_r$.

Крок 2. Розрахувати потенціал центрів кластерів за формулою:

$$P(C^q) = \sum_{s=1}^{S} e^{-\alpha d(C^q, x^s)}, \; q = 1, 2, \ldots, Q,$$

де $C^q = \{C^q_j\}$ – потенційний центр q-го кластера, C^q_j – значення j-ої ознаки для центра q-го кластера; α – позитивна константа, $d(C^q, x^s)$ – відстань (наприклад, евклідова) між потенційним центром кластера C^q та об'єктом кластеризації x^s.

У випадку, коли об'єкти кластеризації задані двома ознаками ($N = 2$), графічне зображення розподілу потенціалу буде являти собою поверхню, що нагадує гірський рельєф. Звідси і назва – гірський метод кластеризации.

Крок 3. Вибрати як центри кластерів координати «гірських» вершин. Для цього, центром першого кластера призначають точку з найбільшим потенціалом. Звичайно, найвища вершина оточена декількома досить високими піками. Тому призначення центром наступного кластера точки з максимальним потенціалом серед вершин, що залишилися, призвело б до виділення великого числа близько розташованих центрів кластерів.

Щоб вибрати наступний центр кластера необхідно спочатку виключити вплив тільки що знайденого кластера. Для цього значення потенціалу для можливих центрів кластерів, що залишилися, перераховується в такий спосіб: від поточних значень потенціалу віднімають внесок центра тільки що знайденого кластера (тому кластеризацію за цим методом іноді називають суб-

трактивною). Перерахунок потенціалу відбувається за формулою: $P_2(C^q) = P_1(C^q) - P_1(C^{v_1})e^{-\beta d(C^q, C^{v_1})}$, де P_1 – потенціал на 1-й ітерації; P_2 – потенціал на 2-й ітерації; v_1 – номер першого знайденого центра кластера: $v_1 = \arg \max\limits_{q=1,2,...,Q} P_1(C^q)$, β – позитивна константа.

Номер центра другого кластера визначається за максимальним значенням оновленого потенціалу: $v_2 = \arg \max\limits_{q=1,2,...,Q} P_2(C^q)$.

Потім знову перераховується значення потенціалів:

$$P_3(C^q) = P_2(C^q) - P_2(C^{v_2})e^{-\beta d(C^q, C^{v_2})}.$$

Крок 4. Якщо максимальне значення потенціалу перевищує деякий поріг, перейти до кроку 2, у противному випадку – зупинення.

Метод гірської кластеризації є ефективним, якщо розмірність вхідного вектора не є занадто великою. У противному випадку (при великій кількості ознак) число потенційних центрів наростає лавинообразно, і процес розрахунку чергових пікових функцій стає занадто тривалим, а процедура кластеризації – малоефективною.

Метод гірської кластеризації можна використовувати для синтезу нечіткої бази знань. Нехай ми маємо $\{C^q\}$ – центри кластерів, знайдені в результаті гірської кластеризації, кожному з яких зіставлене значення цільової ознаки y^q. Тоді кожному центру кластера C^q ставиться у відповідність правило: Якщо $x^s \in C^q$, то $y^s \in y^q$, де нечіткі терми $x^s \in C^q$ та $y^s \in y^q$ характеризуються гаусівськими функціями приналежності:

$$\mu_{x^s \in C^q} = e^{-\frac{1}{2\beta}\sum\limits_{j=1}^{N}(C_j^q - x_j^s)^2}, \quad \mu_{y^s \in y^q} = e^{-\frac{1}{2\beta}(y^q - y^s)^2}.$$

Метод різницевого групування (субтрактивної кластеризації, subtractive clustering) на відміну від попереднього методу, як потенційні центри кластерів розглядає екземпляри навчаючої вибірки.

Пікова функція $P(x^s)$ задається у вигляді:

$$P(x^s) = \sum_{\substack{g=1 \\ g \neq s}}^{S} \exp\left(-\frac{\mathrm{d}(\mathrm{x}^s, \mathrm{x}^g)^{2\mathrm{b}}}{0{,}25\mathrm{r}_\mathrm{a}{}^2}\right), \; s = 1, 2, \ldots, S.$$

Значення коефіцієнта r_a визначає сферу сусідства. На значення $P(x^s)$ істотно впливають тільки ті вектори x^g, що розташовані в межах цієї сфери. При великій щільності точок навколо x^s (потенційного центра) значення функції $P(x^s)$ буде великим. Навпаки, мале її значення свідчить про те, що в околиці x^s знаходиться незначна кількість даних. Така точка вважається «невдалим» кандидатом у центри.

Після розрахунку значень пікової функції серед усіх точок відбирається вектор x^g, для якого міра щільності $P(x^g)$ виявилася найбільшою. Саме ця точка стає першим відібраним центром C^1.

Вибір наступного центра можливий після виключення попереднього центра і всіх точок, що лежать у його околиці. Подібно до методу пікового групування, перевизначається пікова функція:

$$P_2(x^s) = P_1(x^s) - P_1(C^1)\exp\left(-\frac{d(x^s, C^1)^{2b}}{0{,}25{r_b}^2}\right).$$

При визначенні $P_2(x^s)$ коефіцієнт r_b позначає нове значення константи, що задає сферу сусідства чергового центра. Звичайно дотримуються умови $r_b \geq r_a$. Пікова функція $P_2(x^s)$ приймає нульове значення при $x^s = C^1$ і близька до нуля в найближчій околиці цієї точки.

Після модифікації значень пікової функції шукається наступна точка x^g, для якої величина $P_2(x^g)$ виявляється максимальною. Ця точка стає наступним центром кластера C^2.

Процес пошуку чергового центра відновляється після виключення компонентів, що відповідають уже відібраним точкам, і завершується в момент фіксації всіх центрів, передбачених початковими умовами.

Відповідно до описаного методу відбувається самоорганізація множини векторів x, що полягає у знаходженні оптимальних значень центрів, які відповідають множині даних з мінімальною погрішністю.

Якщо ми маємо справу з множиною навчаючих даних у вигляді пар $<x^s, y^s>$, то для знаходження центрів, що відповідають множині векторів y^s, достатньо сформувати розширену версію векторів x: $x^s_{N+1} = y^s$. Процес групування, проведений із пред'явленням розширених векторів x^s, дозволяє визначити також розширені версії центрів C^v, в описі яких легко виділити частину, що відповідає векторам x (перші N компонентів), і залишок, що відповідає вектору y. У такий спосіб можна одержати центри як вхідних змінних, так очікуваних вихідних значень.

Адаптивний метод нечіткої самоорганізації сформульований для гаусівської функції і дозволяє визначати кількість центрів кластерів і їхнє розташування в частині, що відповідає умовам (множина векторів x^s) і висновкам (множина скалярних очікуваних значень y^s). Цей метод можна описати в такий спосіб.

Крок 1. При старті з першої пари даних $<x^1, y^1>$ створюється перший кластер з центром $C^1 = x^1$. Приймається, що $w_1 = y^1$ і що потужність множини $L_1 = 1$. Нехай r позначає граничну евклідову відстань між вектором x та центром, при якому дані будуть трактуватися як приналежні до створеного кластера. Для збереження загальності рішення приймається, що в момент початку навчання існують V кластерів з центрами C^v, $v = 1, 2, \ldots, V$, та відповідні ним значення w_v та L_v, $v = 1, 2, \ldots, V$.

Крок 2. Після зчитування s-ої навчаючої пари $<x^s, y^s>$ розраховуються відстані між вектором x^s та всіма існуючими центрами $d(x^s, C^v)$, $v = 1, 2, ..., V$. Допустимо, що найближчий центр – це C^q. У такому випадку в залежності від значення $d(x^s, C^q)$ може виникнути одна з двох ситуацій:

– якщо $d(x^s, C^q) > r$, тоді створюється новий кластер $C^{V+1} = x^s$, причому $w_{V+1}(s) = y^s$, $L_{V+1}(s) = 1$. Параметри створених до цього кластерів не змінюються, тобто: $w_v(s) = w_v(s–1)$, $L_v(s) = L_v(s–l)$, $v = 1, 2, ..., V$. Збільшується кількість кластерів: $V = V + 1$;

– якщо $d(x^s, C^q) \le r$, тоді дані включаються в кластер C^q, параметри якого слід уточнити відповідно до формул:

$$w_q(s) = w_q(s–1)+y^s,\ L_q(s) = L_q(s–1)+1,\ C^q(s) = (C^q(s–1)L_q(s–1)+x^s) / L_q(s).$$

В іншій версії методу фіксується положення центрів C^q після ініціалізації, і їхні координати вже не змінюються. У багатьох випадках такий прийом поліпшує результати адаптації.

Крок 3. Після уточнення параметрів нечіткої системи функція, що апроксимує вхідні дані системи, визначається як:

$$y^{s*} = \frac{\sum_{v=1}^{V} w_v(s) e^{-\sigma^{-2} d^2(x^s, C^v)}}{\sum_{v=1}^{V} L_v(s) e^{-\sigma^{-2} d^2(x^s, C^v)}},$$

де σ – певна константа, тоді як інші кластери не змінюються.

При повторенні перерахованих кроків методу до $s = S$ з уточненням щоразу значення V простір даних розподіляється на V кластерів, при цьому потужність кожного з них визначається як $L_v = L_v(s)$, центр – як $C^v = C^v(s)$, а значення приписаної йому накопиченої функції y^s – як $w_v = w_v(s)$.

Цей метод є таким, що самоорганізується, оскільки поділ простору даних на кластери відбувається самостійно і без участі людини, відповідно до заданого значення порога r. При малому значенні r кількість кластерів зростає, у результаті чого апроксимація даних стає більш точною, однак це досягається за рахунок більш складної функції і збільшення обсягу необхідних обчислень при одночасному погіршенні узагальнюючих властивостей моделі. Якщо значення r є занадто великим, то обчислювальна складність зменшується, однак зростає погрішність апроксимації. При підборі оптимальної величини порога r повинний дотримуватися компроміс між точністю відображення й обчислювальною складністю. Як правило, оптимальне значення r підбирається методом проб і помилок з використанням обчислювальних експериментів.

Метод поступово зростаючого розбиття (incremental decomposition algorithm) полягає в наступному. На першій ітерації є одне продукційне правило, що має як область свого впливу (область, у якій значення результуючої функції

приналежності передумови нечіткого правила перевищує задану величину) усю множину припустимих вхідних значень (рис. 3.2).

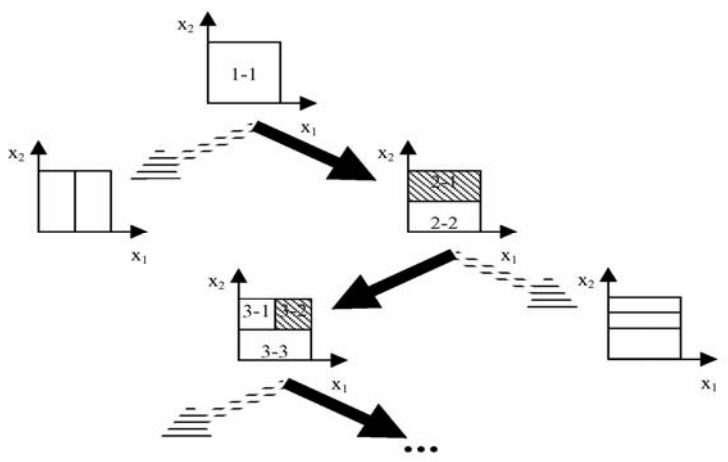


Рисунок 3.2 – Схема роботи методу IDA

На другій ітерації дане правило розбивається на два двома способами (показані стрілками). Проводиться навчання і вибирається, який зі способів розбиття дає найменшу погрішність (даний перехід відзначений чорною стрілкою). Серед наявних правил вибирається те, для якого складова погрішності в загальній погрішності є найбільшою (область його впливу заштриховано). Воно і підлягає розбиттю на два правила двома способами (ітерація 3). Описаний процес продовжується до досягнення необхідної точності або поки не буде згенеровано задане число продукційних правил.

3.13 Програмні засоби для синтезу нечітких моделей

Одним з найпотужніших засобів для побудови нечітких моделей є пакет MATLAB. Операції з нечіткою логікою у пакеті MATLAB дозволяє виконувати модуль Fuzzy Logic Toolbox. Він дозволяє створювати системи нечіткого логічного виведення і нечіткої класифікації в рамках середовища MatLab, з можливістю їхньої інтеграції в засіб Simulink пакету MATLAB.

Fuzzy Logic Toolbox містить такі категорії програмних інструментів: функції; інтерактивні модулі з графічним користувальницьким інтерфейсом (GUI) – див. рис. 3.3, блоки для пакета Simulink; демонстраційні приклади.

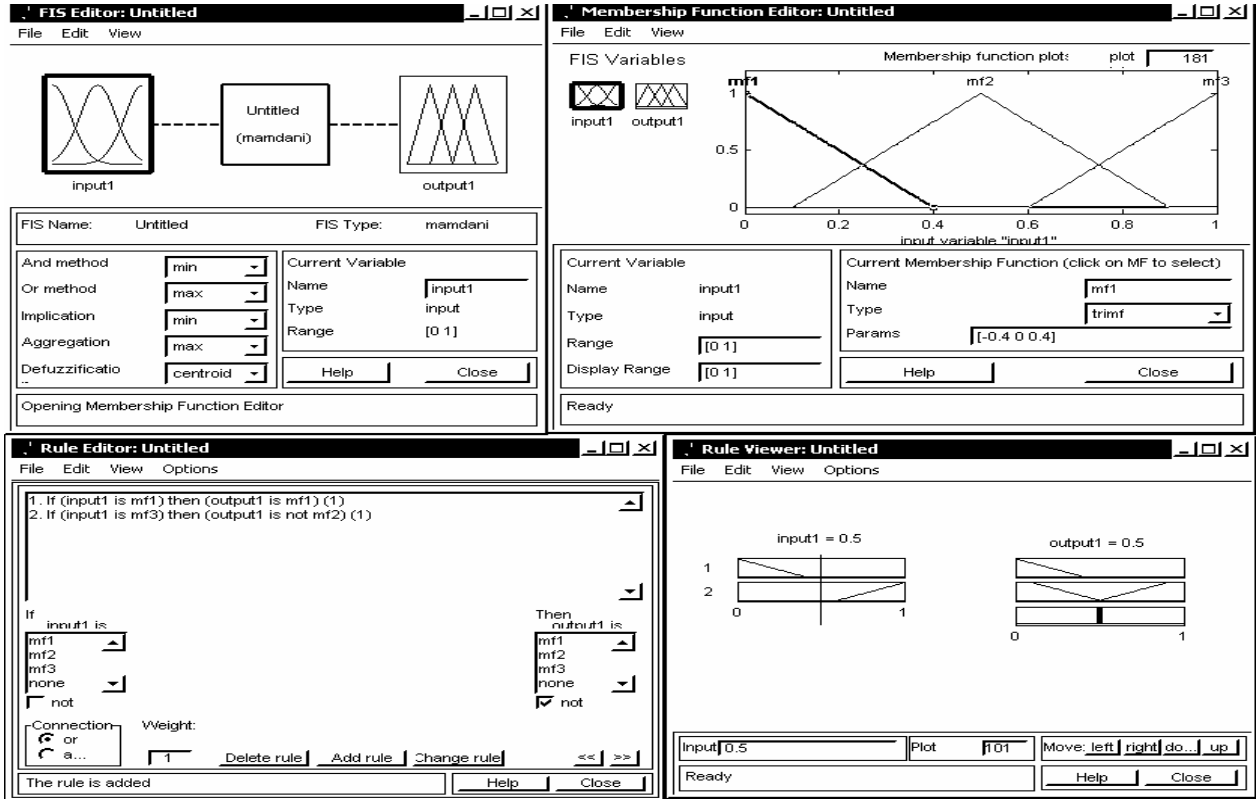


Рисунок 3.3 – Графічні форми інструментів для нечіткого виведення у пакеті MATLAB

Функції модуля Fuzzy Logic Toolbox можуть бути викликані з командного рядка. Для отримання переліку функцій слід ввести команду: `help fuzzy`. Наведемо короткий огляд функцій модуля Fuzzy Logic.

Редактори з графічним інтерфейсом користувача: findcluster – інструмент для кластеризації; fuzzy – базовий редактор FIS; mfedit – редактор функцій приналежності; ruleedit – редактор та аналізатор правил; ruleview – демонстратор правил та діаграм нечіткого виведення; surfview – демонстратор вихідної поверхні.

Функції приналежності: dsigmf, gauss2mf, gaussmf, gbellmf, pimf, psigmf, smf, sigmf, trapmf, trimf, zmf. Для зручності імена всіх убудованих функцій приналежності закінчуються на mf. Виклик функції приналежності здійснюється в такий спосіб: namemf(*x*, params), де namemf – найменування функції приналежності; *x* – вектор, для координат якого необхідно розрахувати значення функції приналежності; params – вектор параметрів функції приналежності.

У Fuzzy Logic Toolbox передбачена можливість для користувача створення власної функції приналежності. Для цього необхідно створити *m*-функцію, що містить два вхідних аргументи – вектор, для координат якого необхідно розраховувати значення функції приналежності і вектор параметрів функції приналежності. Вихідним аргументом функції повинний бути вектор ступенів приналежності.

Функції FIS: addmf – додає функцію приналежності до FIS; addrule – додає правило до FIS; addvar – додає змінну до FIS; defuzz – дефазифікує функцію приналежності; evalfis – здійснює обчислення нечіткого виведення; evalmf – обчислює функцію приналежності; gensurf – генерує поверхню виходу FIS; getfis – повертає властивості нечіткої системи; mf2mf – транслює параметри між функціями приналежності; newfis – створює нову FIS; parsrule – аналізує нечіткі правила; plotfis – показує діаграму «вхід-вихід» для FIS; plotmf – показує усі функції приналежності для однієї змінної; readfis – завантажує FIS з диску; rmmf – видаляє функцію приналежності з FIS; rmvar – видаляє змінну з FIS; setfis – встановлює властивості нечіткої системи; showfis – показує анотовану FIS; showrule – відображує правила FIS; writefis – зберігає FIS на диску.

Функції кластер-аналізу: fcm, genfis1, genfis2; subclust.

Різні функції: convertfis – перетворює нечітку матрицю структури версії 1.0 у матрицю структури версії 2.0; discfis – дискретизує FIS; evalmmf – використовується для обчислення множинних функцій приналежності; fstrvcat – поєднує матриці різного розміру; fuzarith – функція нечіткої арифметики; findrow – шукає рядки матриці, що відповідають вхідному рядку; probor – імовірнісне АБО; sugmax – максимальний вихідний діапазон для системи Сугено.

Функція `C=fuzarith(X,A,B,OPERATOR)` реалізує базові операції нечіткої логіки та повертає нечітку множину C як результат застосування оператора OPERATOR ('sum' – сума, 'sub' – вирахування, 'prod' – добуток, 'div' – ділення) до нечітких множин *A* та *B* з універсальної множини *X*. Змінні *A*, *B* та *X* мають бути векторами однакової розмірності. Нечітка множина C, яка повертається, є

вектор-стовпцем тієї є довжини, що й *A* та *B*. Зауважимо, що ця функція використовує інтервальну арифметику та передбачає, що: *A* та *B* є опуклими нечіткими множинами; функції приналежності для *A* та *B* поза *X* є нулем.

Допоміжні функції графічного користувальницького інтерфейсу: cmfdlg – додає діалог вибору функцій приналежності; cmthdlg – додає діалог вибору методу виведення; fisgui – дискрипторне посилання на інтерфейсні засоби модуля Fuzzy Logic Toolbox; gfmfdlg – генерує FIS з використанням діалогу методу ґрат; mfdlg – додає діалог функції приналежності; mfdrag – перетягування функції приналежності за допомогою миші; popundo – відновлює зі стеку останні зміни (відміняє останні дії); pushundo – передає поточні дані у стек відновлення; savedlg – діалог запису перед закриттям; statmsg – зображує повідомлення у полі статусу; updtfis – оновлює засоби графічного інтерфейсу Fuzzy Logic Toolbox; wsdlg – діалог «відкриття з» / «збереження до» робочої області.

FIS-структура (Fuzzy Inference System – система нечіткого виведення) містить усі необхідні дані для реалізації функціонального відображення «входи-виходи» на основі нечіткого логічного виведення. Поля структури даних системи нечіткого логічного виведення призначені для збереження такої інформації:

– name – найменування системи нечіткого логічного виведення;

– type – тип системи (припустимі значення 'Mamdani' та 'Sugeno');

– andMethod – реалізація логічної операції «ТА» (запрограмовані реалізації: 'min' – мінімум і 'prod' – множення);

– orMethod – реалізація логічної операції «АБО» (запрограмовані реалізації: 'max' – максимум і 'probor' – імовірнісне «АБО»);

– defuzzMethod – метод дефазифікації (запрограмовані методи для систем типу Мамдані: 'centroid' – центр ваги; 'bisector' – медіана; 'lom' – найбільший з максимумів; 'som' – найменший з максимумів; 'mom' – середнє з максимумів; запрограмовані методи для систем типу Сугено: 'wtaver' – зважене середнє і 'wtsum' – зважена сума);

– impMethod – реалізація операції імплікації (запрограмовані реалізації: 'min' – мінімум і 'prod' – множення);

– aggMethod – реалізація операції об'єднання функцій приналежності вихідної змінної (запрограмовані реалізації: 'max' – максимум; 'sum' – сума і 'probor' – імовірнісне «АБО»);

– input – масив вхідних змінних: input.name – найменування вхідної змінної; input.range – діапазон зміни вхідної змінної; input.mf – масив функцій приналежності вхідної змінної; input.mf.name – найменування функції приналежності вхідної змінної; input.mf.type – модель функції приналежності вхідної змінної; input.mf.params – масив параметрів функції приналежності вхідної змінної;

– output – масив вихідних змінних: output.name – найменування вихідної змінної; output.range – діапазон зміни вихідної змінної; output.mf – масив функцій приналежності вихідної змінної; output.mf.name – найменування фу-

нкції приналежності вихідної змінної; output.mf.type – модель функції приналежності вихідної змінної; output.mf.params – масив параметрів функції приналежності вихідної змінної;

– rule – масив правил нечіткої бази знань: rule.antecedent – посилки правила (вказуються порядкові номери термів у порядку запису вхідних змінних. Число 0 указує на те, що значення відповідної вхідний змінної не впливає на істинність правила); rule.consequent – висновок правила (вказуються порядкові номери термів у порядку запису вихідних змінних. Число 0 указує на те, що правило не поширюється на відповідну вихідну змінну); rule.weight – вага правила. Задається числом з діапазону [0, 1]; rule.connection – логічне зв'язування змінних усередині правила: 1 – логічне «ТА»; 2 – логічне «АБО».

Для доступу до властивостей системи нечіткого логічного виведення треба вказати ім'я відповідного поля. Система нечіткого виведення зберігається на диску у вигляді fis-файлу – текстового файлу спеціального формату. Функції readfis та writefis використовуються для завантажена у робочу область та збереження на диску таких файлів.

FIS-редактор (FIS Editor) призначений для створення, збереження, завантаження і виведення у друк систем нечіткого логічного виведення, а також для редагування таких властивостей: тип системи; найменування системи; кількість вхідних і вихідних змінних; найменування вхідних і вихідних змінних; параметри нечіткого логічного виведення. FIS-редактор містить три загальносистемних меню: File, Edit, View, та п'ять меню для вибору параметрів нечіткого виведення: And Method, Or Method, Implication, Aggregation та Defuzzification.

Меню File – це загальне меню для всіх GUI-модулів використовуваних із системами нечіткого виведення. За допомогою команди New FIS користувач має можливість створити нову систему нечіткого виведення. При виборі цієї команди з'являться дві альтернативи: Mamdani та Sugeno, що визначають тип створюваної системи. За допомогою команди Import користувач має можливість завантажити раніше створену систему нечіткого логічного виведення. Команда Export дозволяє зберегти систему нечіткого виведення в робочу область або на диск. Команда Print дозволяє вивести на принтер копію графічного вікна. Команда Close закриває графічне вікно.

Меню Edit. Команда Undo скасовує раніше зроблену дію. Команда Add Variable дозволяє додати в систему ще одну змінну: вхідну (input) або вихідну (output). Команда Remove Selected Variable видаляє поточну змінну із системи. Команда Membership Function відкриває редактор функцій приналежностей. Команда Rules відкриває редактор бази знань.

Меню View – це загальне меню для всіх GUI-модулів, використовуваних із системами нечіткого виведення. Це меню дозволяє відкрити вікно візуалізації нечіткого виведення (команда Rules) та вікно виведення поверхні «вхід-вихід», що відповідає системі нечіткого виведення (команда Surface).

Меню *And Method*, *Or Method*, *Implication* та *Aggregation* дозволяють установити реалізації логічних операцій «ТА», «АБО», імплікація, об'єднання, відповідно. *Меню Defuzzification* дозволяє вибрати метод дефазифікації. Користувач також має можливість установити власну реалізацію операцій. Для цього необхідно вибрати команду Custom та в графічному вікні, що з'явилося, надрукувати ім'я функції, яка реалізує цю операцію.

Панель *Current Variable* дозволяє для поточної змінної, ім'я котрої задається у полі Name, задати у полі Type тип (вхідна – input, вихідна – output), а також у полі Range – діапазон значень.

FIS-редактор також має інформаційну панель, що відображає ім'я поточної FIS (поле FIS Name) та її тип (поле FIS Type). Крім того FIS-редактор має панель із графічним зображенням системи виведення, вхідних та вихідних змінних.

Редактор функцій приналежності (Membership Function Editor) призначений для задавання такої інформації про терми-множини вхідних і вихідних змінних: кількість термів; найменування термів; тип і параметри функцій приналежності, що необхідні для подання лінгвістичних термів у вигляді нечітких множин. Редактор функцій приналежності містить меню File, Edit, View та панелі введення інформації.

Меню Edit. Команда Add MFs дозволяє додати терми в терм-множину, використовувану для лінгвістичної оцінки поточної змінної. При виборі цієї команди з'явиться діалогове вікно, у якому необхідно вибрати тип функції приналежності та кількість термів. Значення параметрів функцій приналежності будуть встановлені автоматично таким чином, щоб рівномірно покрити область визначення змінної, заданої у полі Range. При зміні області визначення у полі Range параметри функцій приналежності будуть промаштабовані. Команда Add Custom MF дозволяє додати лінгвістичний терм, функція приналежності якого відрізняється від убудованих. Після вибору цієї команди з'явиться графічне вікно, у якому необхідно надрукувати лінгвістичний терм (поле MF name), ім'я функції приналежності (поле M-File function name) і параметри функції приналежності (поле Parameter list). Команда Remove Selected MF видаляє поточний терм із терм-множини поточної змінної. Ознакою поточної змінної є червона окантовка її прямокутника. Ознакою поточного терму є червоний колір його функції приналежності. Для вибору поточного терму необхідно позиціонувати курсор миші на графіку функції приналежності і зробити щиглик лівою кнопкою миші. Команда Remove All MFs видаляє всі терми з терм-множини поточної змінної.

Панель *Current Variable* дозволяє для поточної змінної, ім'я котрої вказане у полі Name, а тип у полі Type, задати у полі Range діапазон значень, а у полі Display Range – діапазон відображення. Панель *Current Membership Function* дозволяє для поточної функції приналежності, ім'я терму котрої вказують у полі Name, задати тип (назву MATLAB-функції) у полі Type та параметри у полі

Params. Редактор також має панель із зображенням графіку поточної функції приналежності.

Редактор бази знань (Rule Editor) призначений для формування і модифікації нечітких правил. Редактор функцій приналежності містить чотири системних меню File, Edit, View, Options, меню вибору термів вхідних і вихідних змінних, поля установки логічних операцій «ТА», «АБО», «НЕ» і ваг правил, а також кнопки редагування і перегляду правил.

Для введення нового правила в базу знань необхідно за допомогою миші вибрати відповідну комбінацію лінгвістичних термів вхідних і вихідних змінних, установити тип логічного зв'язування («ТА» або «АБО») між змінними усередині правила, установити наявність чи відсутність логічної операції «НЕ» для кожної лінгвістичної змінний, увести значення вагового коефіцієнта правила і натиснути кнопку Add Rule. За замовчуванням установлені параметри: логічне зв'язування змінних усередині правила – «ТА»; логічна операція «НЕ» – відсутня; значення вагового коефіцієнта правила – 1.

Можливі випадки, коли істинність правила не змінюється при довільної значенні деякої вхідної змінної, тобто ця змінна не впливає на результат нечіткого логічного виведення в даній області факторного простору. Тоді як лінгвістичне значення цієї змінної необхідно установити none.

Для видалення правила з бази знань необхідно зробити однократний щиглик лівою кнопкою миші на цьому правилі та натиснути кнопку Delete Rule.

Для модифікації правила необхідно зробити однократний щиглик лівою кнопкою миші на цьому правилі, потім установити необхідні параметри правила і натиснути кнопку Edit Rule.

Меню Options дозволяє установити мову і формат правил бази знань. При виборі команди Language з'явиться список мов English (Англійська), Deutsch (Німецька), Francais (Французька), з якого необхідно вибрати одну. При виборі команди Format з'явиться список можливих форматів правил бази знань: Verbose – лінгвістичний; Symbolic – логічний; Indexed – індексований.

Модуль візуалізації нечіткого виведення (Rule Viewer) дозволяє проілюструвати хід логічного виведення за кожним правилом, одержання результуючої нечіткої множини й виконання процедури дефазифікації. Rule Viewer містить чотири меню – File, Edit, View, Options, два поля уведення інформації – Input і Plot points та кнопки прокручування зображення вліво – вправо (left-right), догори – униз (up-down).

Кожне правило бази знань представляється у виді послідовності горизонтально розташованих прямокутників. При цьому перші два прямокутники відображають функції приналежностей термів посилки правила (ЯКЩО-частина правила), а останній третій прямокутник відповідає функції приналежності терму-наслідку вихідної змінної (ТО-частина правила).

Порожній прямокутник у візуалізації правила означає, що в цьому правилі посилка за змінною відсутня. Жовте заливання графіків функцій приналежностей вхідних змінних указує наскільки значення входів відповідають термам даного правила. Для виведення правила у форматі Rule Editor необхідно зробити однократний щиглик лівою кнопкою миші по номеру відповідного правила. У цьому випадку зазначене правило буде виведено в нижній частині графічного вікна. Блакитне заливання графіка функції приналежності вихідної змінної являє собою результат логічного виведення у вигляді нечіткої множини за даним правилом. Результуючу нечітку множину, що відповідає логічному виведенню за всіма правилами, показано в нижньому прямокутнику останнього стовпця графічного вікна. У цьому ж прямокутнику червона вертикальна лінія відповідає чіткому значенню логічного виведення, отриманого в результаті дефазифікації.

Уведення значень вхідних змінних може здійснюватися двома способами: шляхом уведення чисельних значень у поле Input; за допомогою миші, шляхом переміщення ліній-покажчиків червоного кольору.

У полі Plot points задається кількість крапок дискретизації для побудови графіків функцій приналежності. Значення за замовчуванням – 101.

3.14 Приклади та ілюстрації

Приклад 1. Нехай $E = \{x_1, x_2, x_3, x_4, x_5\}$, $M = [0, 1]$; A – нечітка множина, для якої $\mu_A(x_1) = 0{,}3$; $\mu_A(x_2) = 0$; $\mu_A(x_3) = 1$; $\mu_A(x_4) = 0{,}5$; $\mu_A(x_5) = 0{,}9$. Тоді A можна подати у вигляді: $A = \{0{,}3/x_1;\ 0/x_2;\ 1/x_3;\ 0{,}5/x_4;\ 0{,}9/x_5\}$ або $A = \{0{,}3|x_1;\ 0|x_2;\ 1|x_3;\ 0{,}5|x_4;\ 0{,}9|x_5\}$ або $A = 0{,}3/x_1 + 0/x_2 + 1/x_3 + 0{,}5/x_4 + 0{,}9/x_5$ або $A = \{0{,}3|x_1;\ 0|x_2;\ 1|x_3;\ 0{,}5|x_4;\ 0{,}9|x_5\}$ або $A = \{\langle 0{,}3;\ x_1\rangle, \langle 0;\ x_2\rangle, \langle 1;\ x_3\rangle; \langle 0{,}5;\ x_4\rangle, \langle 0{,}9;\ x_5\rangle\}$ або

$$A = \begin{array}{|c|c|c|c|c|} \hline x_1 & x_2 & x_3 & x_4 & x_5 \\ \hline 0{,}3 & 0 & 1 & 0{,}5 & 0{,}9 \\ \hline \end{array} \text{ або } A = \begin{array}{|c|c|c|c|c|} \hline 0{,}3 & 0 & 1 & 0{,}5 & 0{,}9 \\ \hline x_1 & x_2 & x_3 & x_4 & x_5 \\ \hline \end{array}$$

Тут знак «+» не є позначенням операції додавання, а має сенс об'єднання.

Приклад 2. Нехай $E = \{1, 2, 3, ..., 100\}$ і відповідає поняттю «вік», тоді нечітка множина «молодий», може бути визначена у такий спосіб:

$$\mu_{\text{молодий}}(x) = \begin{cases} 1, x \in [1{,}25], \\ 1 + (0{,}2(x-25))^{-2}, x > 25. \end{cases}$$

Нечітка множина «молодий» на універсальній множині E' = {Іваненко, Петренко, Сидоренко, ...} задається за допомогою функції приналежності $\mu_{\text{молодий}}(x)$ на $E = \{1, 2, 3, ..., 100\}$ (вік), названої стосовно E' функцією сумісності, при цьому: $\mu_{\text{молодий}}(\text{Сидоренко}) = \mu_{\text{молодий}}(x)$, де x – вік Сидоренка.

Приклад 3. Нехай ми маємо нечіткі множини $A = \{0{,}1|1;\ 0{,}5|2;\ 1{,}0|3;\ 0{,}8|3{,}2\}$ та $B = \{0{,}6|1;\ 0{,}1|2;\ 0{,}1|3\}$. Визначимо їхні властивості та результати операцій над ними. У якості скалярного значення будемо використовувати число $V_1 = 3$, або, за потреби, $V_1^{-1} \approx 0{,}3$.

Характеристики множини A: висота: 1; нормальність: нормальна; порожність: непорожня; унімодальність: унімодальна; ядро: {3}; носій: {1; 2; 3; 3,2}; межі: {0,1|1; 0,5|2; 0,8|3,2}; точки переходу: {0,5|2}; найближча чітка множина: {2; 3; 3,2}; опуклість: опукла; міра нечіткості Ягера в метриці p = 1: 0,4; міра нечіткості Ягера в метриці p = 2: 0,5; міра нечіткості Коско: 0,348; чітка множина альфа-рівня ($\alpha = V_1^{-1} = 0{,}3$): {2; 3; 3,2}.

Характеристики множини B: висота: 0,6; нормальність: субнормальна; порожність: непорожня; унімодальність: неунімодальна; ядро: $\varnothing$; носій: {1; 2; 3}; межі: {0,6|1; 0,1|2; 0,1|3}; точки переходу: $\varnothing$; найближча чітка множина: {1}; опуклість: опукла; міра нечіткості Ягера в метриці p = 1: 0,4; міра нечіткості Ягера в метриці p = 2: 0,337; міра нечіткості Коско: 0,25; чітка множина альфа-рівня ($\alpha = V_1^{-1} = 0{,}3$): {1}.

Унарні операції з A: доповнення: {0,9|1; 0,5|2; 0,2|3,2}; зведення в ступінь (V_1 = 3): {0,001|1; 0,125|2; 1,0|3; 0,512|3,2}; CON: {0,01|1; 0,25|2; 1,0|3; 0,64|3,2}; DIL: {0,316|1; 0,707|2; 1,0|3; 0,894|3,2}; множення на число ($a = V_1^{-1} = 0{,}3$): {0,3|1; 0,15|2; 0,3|3; 0,24|3,2}; нормалізація: {0,1|1; 0,5|2; 1,0|3; 0,8|3,2}.

Унарні операції з B: доповнення: {0,4|1; 0,9|2; 0,9|3}; зведення в ступінь (V_1 = 3): {0,216|1; 0,001|2; 0,001|3}; CON: {0,36|1; 0,01|2; 0,01|3}; DIL: {0,775|1; 0,316|2; 0,316|3}; множення на число ($a = V_1^{-1} = 0{,}3$): {0,18|1; 0,03|2; 0,03|3}; нормалізація: {1,0|1; 0,167|2; 0,167|3}.

Бінарні операції з A та B: включення: $A \not\subseteq B$; рівність: $A \neq B$; об'єднання: {0,6|1; 0,5|2; 1,0|3; 0,8|3,2}; перетинання: {0,1|1; 0,1|2; 0,1|3}; різниця: {0,4|2; 0,9|3; 0,8|3,2}; симетрична різниця: {0,5|1; 0,4|2; 0,9|3; 0,8|3,2}; диз'юнктивна сума: {0,6|1; 0,5|2; 0,9|3}; алгебраїчний добуток: {0,06|1; 0,05|2; 0,1|3}; алгебраїчна сума: {0,694|1; 0,595|2; 1,0|3; 0,8|3,2}; обмежена сума: {0,7|1; 0,6|2; 1,0|3; 0,8|3,2}; обмежена різниця: {0,4|2; 0,9|3; 0,8|3,2}; обмежений добуток: {0,1|3}; драстичне перетинання: {0,1|3}; драстичне об'єднання: {1,0|1; 1,0|2; 1,0|3; 0,8|3,2}; λ-сума ($\lambda = V_1^{-1} = 0{,}3$): {0,45|1; 0,22|2; 0,37|3; 0,24|3,2}; опукла комбінація ($\lambda_1 = V_1^{-1} = 0{,}3$, $\lambda_2 = 1 - \lambda_1 = 0{,}7$): {0,72|1; 0,22|2; 0,37|3; 0,24|3,2}; нечітке включення за Лукасевичем: {0,1|3}; нечітке включення за Заде: {0,1|3}.

Приклад 4. Нехай E = {1, 2, 3, 4}; A = 0,8/1 + 0,6/2 + 0/3 + 0/4; $K(1)$ = 1/1 + 0,4/2; $K(2)$ = 1/2 + 0,4/1 + 0,4/3; $K(3)$ = 1/3 + 0,5/4; $K(4)$ = 1/4. Тоді $H(A, K) = \mu_A(1)K(1) \cup \mu_A(2)K(2) \cup \mu_A(3)K(3) \cup \mu_A(4)K(4) = 0{,}8(1/1 + + 0{,}4/2) \cup 0{,}6(1/2 + 0{,}4/1 + 0{,}4/3) = 0{,}8/1 + 0{,}6/2 + 0{,}24/3$.

Приклад 5. Нехай x_1 = {3, 5, 7}, x_2, = {3, 5, 7}, A_1 = {0,5|3 + 1|5 + 0,6|7}, A_2 = {1|3+0,6|5}.

Тоді декартовий (прямий) добуток $A = A_1 \times A_2$ ={<0,5|3, 3>+ +<0,5|3, 5> + <1|5, 3> + <0,6|5, 5> +<0,6|7, 3> + <0,6|7, 5>}.

Приклад 6. Нехай $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2, y_3, y_4\}$, M = [0, 1]. Нечітке відношення $R = X\,R\,Y$ може бути задане таблицею:

R	y_1	y_2	y_3	y_4
x_1	0	0	0,1	0,3
x_2	0	0,8	1	0,7
x_3	1	0,5	0,6	1

Приклад 7. Нехай $X = Y = (-\infty,\infty)$, тобто множина усіх дійсних чисел. Відношення *x*»*y* (*x* забагато більший ніж *y*) можна задати функцією приналежності:

$$\mu_R = \begin{cases} 0, \text{ якщо } x \le y, \\ \left(1 + (x-y)^{-2}\right)^{-1}, \text{ якщо } y > x. \end{cases}$$

Приклад 8. Відношення *R*, для котрого $\mu_R(x,y) = e^{-k(x-y)^2}$ при достатньо великих *k* можна інтерпретувати так: «*x* та *y* близькі одне до одного числа».

Приклад 9. Нехай нечіткі відношення R_1 та R_2 задаються такими таблицями:

R_1	y_1	y_2	y_3
x_1	0,1	0,7	0,4
x_2	1	0,5	0

R_2	z_1	z_2	z_3	z_4
y_1	0,9	0	1	0,2
y_2	0,3	0,6	0	0,9
y_3	0,1	1	0	0,5

Тоді нечітке відношення $R_1 \bullet R_2$ може бути задане таблицею:

$R_1 \bullet R_2$	z_1	z_2	z_3	z_4
x_1	0,3	0,6	0,1	0,7
x_2	0,9	0,5	1	0,5

При цьому:

$$\mu_{R_1 \bullet R_2}(x_1, z_1) = [\mu_{R_1}(x_1, y_1) \cap \mu_{R_2}(y_1, z_1)] \cup [\mu_{R_1}(x_1, y_2) \cap \mu_{R_2}(y_2, z_1)] \cup$$

$$\cup [\mu_{R_1}(x_1, y_3) \cap \mu_{R_2}(y_3, z_1)] = (0{,}1 \cap 0{,}9) \cup (0{,}7 \cap 0{,}3) \cup (0{,}4 \cap 0{,}1) =$$

$$= 0{,}1 \cup 0{,}3 \cup 0{,}1 = 0{,}3;$$

$$\mu_{R_1 \bullet R_2}(x_1, z_2) = (0{,}1 \cap 0) \cup (0{,}7 \cap 0{,}6) \cup (0{,}4 \cap 1) = 0 \cup 0{,}6 \cup 0{,}4 = 0{,}6.$$

У цьому прикладі спочатку використано «аналітичний» спосіб композиції відношень R_1 та R_2, тобто *i*-й рядок R_1 «збільшується» на *j*-й стовпець R_2 з використанням операції $\cap$, потім отриманий результат «згортається» з використанням операції $\cup$ в $\mu(x_i, z_j)$.

Приклад 10. Нехай ми маємо нечітку множину для поняття «чоловік середнього росту»: *A*={0|155, 0,1|160, 0,3|165, 0,8|170, 1|175, 1|180, 0,5|185, 0|190}. Проведемо дефазифікацію нечіткої множини «чоловік середнього росту» за методом центра тяжіння: *a* = (0·155 + 0,1·160 + 0,3·165 + 0,8·170 + +1·175 + 1·180 + 0,5·185 + 0·190) / (0 + 0,1 + 0,3 + 0,8 + 1 + 1 + 0,5 + 0) = 175,4.

Приклад 11. Спадне нечітке логічне виведення. Нехай задана модель діагностики системи, що складається з двох передумов і трьох висновків: $X = \{x_1, x_2\}$, $Y = \{y_1, y_2, y_3\}$, а, матриця нечітких відношень має вигляд:

$$R = \begin{bmatrix} 0{,}8 & 0{,}2 & 0{,}3 \\ 0{,}7 & 0{,}4 & 0{,}5 \end{bmatrix}.$$

Припустимо, що в результаті діагностики системи були отримані такі висновки: $Y = 0{,}8/y_1 + 0{,}2/y_2 + 0{,}3/y_3$. Необхідно знайти передумови, що до цього призвели: $A = a_1/x_1 + a_2/x_2$.

З урахуванням конкретних даних відношення між передумовами та висновками будуть подані в транспонованому вигляді в такий спосіб:

$$\begin{bmatrix} 0{,}8 \\ 0{,}2 \\ 0{,}3 \end{bmatrix} = \begin{bmatrix} 0{,}8 & 0{,}7 \\ 0{,}2 & 0{,}4 \\ 0{,}3 & 0{,}5 \end{bmatrix} \bullet \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}.$$

При використанні max-min-композиції останнє співвідношення перетвориться до вигляду: $0{,}8 = (0{,}8 \cap a_1) \cup (0{,}7 \cap a_2)$, $0{,}2 = (0{,}2 \cap a_1) \cup (0{,}4 \cap a_2)$, $0{,}3 = (0{,}3 \cap a_1) \cup (0{,}5 \cap a_2)$. При вирішенні даної системи помітимо, що в першому рівнянні другий член правої частини не впливає на ліву частину: $0{,}8 = 0{,}8 \cap a_1$, $a_1 \geq 0{,}8$. З другого рівняння одержимо: $0{,}2 = 0{,}4 \cap a_2$, $a_2 \leq 0{,}2$. Отримане рішення задовольняє третьому рівнянню. У такий спосіб: $0{,}8 < a_1 < 1$, $0 < a_2 < 0{,}2$.

При вирішенні практичних задач можуть одночасно використовуватися різні правила композиції нечітких виведень. Сама схема виведень може бути багатокаскадною. В даний час загальних методів вирішення подібних задач, очевидно, не існує.

Приклад 12. Створити на мові пакету MATLAB *m*-функцію, що реалізує колоколообразну функцію приналежності $\mu(x) = 1/\left(1 + ((x-b)/a)^2\right)$.

```
function mu=bellmf(x, params)
%bellmf - колоколообразна функція приналежності;
%x - вхідний вектор;
%params(1) - коефіцієнт концентрації (>0);
%params(2) - координата максимуму.
a=params(1); b=params(2);
mu=1./(1 + ((x - b)/a).^2);
```

Приклад 13. Використання функції fuzarith.

```
% визначення розподіляючої здатності функцій приналежності
point_n = 101;
min_x = -20; max_x = 20;% універсальна множина є [min_x, max_x]
x = linspace(min_x, max_x, point_n)';
A = trapmf(x, [-10 -2 1 3]);% трапеціедальна нечітка множина A
B = gaussmf(x, [2 5]);% гаусівська нечітка множина B
C1 = fuzarith(x, A, B, 'sum');
subplot(2,2,1); plot(x,A,'y--',x,B,'m:',x,C1,'c');
title('нечітке додавання A+B');
C2 = fuzarith(x, A, B, 'sub');
subplot(2,2,2); plot(x,A,'y--',x,B,'m:',x,C2,'c');
title('нечітке вирахування A-B');
```

```
C3 = fuzarith(x, A, B, 'prod');
subplot(2,2,3); plot(x,A,'y--',x,B,'m:',x,C3,'c');
title('нечітке множення A*B');
C4 = fuzarith(x, A, B, 'div');
subplot(2,2,4); plot(x,A,'y--',x,B,'m:',x,C4,'c');
title('нечітке ділення A/B');
```

Зауваження. У програмах на мові MATLAB небажано використовувати кириличні літери. У наведених прикладах україномовні фрази наведено лише для пояснення тексту програм.

Приклад 14. Нехай ми вирішуємо задачу визначення діагнозу пацієнта за значеннями симптомів, що характеризують його стан. Вхідними параметрами є симптоми пацієнта: температура тіла (нормальна – близько 36,6 ˚С, підвищена – більше 37 ˚С), місце болю (у спині, у попереку), вид болю (сильний, слабкий). Вихідним параметром є діагноз пацієнта (хронічний апендицит, гострий апендицит, хронічний пієлонефрит, гострий пієлонефрит). Ми маємо визначені експертами правила (ми будемо вважати, що якщо апендицит або пієлонефрит не є гострими, то вони є хронічними):

ЯКЩО			ТО		
температура	місце болю	вид болю	апендицит	пієлонефрит	гострий
підвищена	у спині	сильний	немає	є	є
підвищена	у животі	сильний	є	немає	є
нормальна	у спині	слабкий	немає	є	немає
нормальна	у животі	слабкий	є	немає	немає

Необхідно побудувати систему нечіткого виведення.

1. Завантажимо пакет MATLAB та запустимо редактор FIS: `fuzzy`.

2. Створимо три змінні (одна вже є за замовчанням, тому додамо ще дві змінні): у головному меню редактора у підменю Edit оберемо опцію Add variables: Input. Це потрібно зробити 2 рази. Після чого створимо три вихідні змінні (подібним чином, але обираючи Add variables: Output).

3. Переміщуючи курсор миші по областях змінних можна обрати потрібну змінну та задати її назву в області Current Variable: Name. Назвемо наші змінні: temperature – температура, painlocation – місце болю, paintype – вид болю. Також задамо назви вихідних змінних: apendicite – апендицит, pielonefrit – пієлонефрит, acute – гострий, chronic – хронічний.

4. Подвійним щигликом на області якої-небудь змінної можна активізувати редактор функцій приналежності. За його допомогою визначимо діапазон кожної змінної та необхідні функції приналежності.

5. У головному меню редактора FIS оберемо в підменю Edit пункт Rules. Тепер ми маємо можливість задавати наші правила.

6. У головному меню редактора FIS оберемо в підменю View пункт Rules. Переміщуючи курсором миші лінії, що регулюють значення змінних,

будемо отримувати значення вихідних змінних при відповідному наборі значень вхідних змінних.

Завантаживши файл Sample.fis із наведеним нижче текстом до редактора FIS можна переглянути описане у прикладі вирішення задачі.

Вміст файлу Sample.fis:

```
[System]
Name='Sample'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=4
NumRules=4
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'
[Input1]
Name='temperature'
Range=[35 42]
NumMFs=2
MF1='normal':'trimf',[35.9 36.6 37]
MF2='high':'smf',[36.8 37.2]
[Input2]
Name='painlocation'
Range=[0 1]
NumMFs=2
MF1='stomach':'trimf',[0 0 0.5]
MF2='spine':'trimf',[0.5 1 1]
[Input3]
Name='paintype'
Range=[0 1]
NumMFs=2
MF1='acute':'trimf',[0.5 1 1]
MF2='weak':'trimf',[0 0 0.5]
[Output1]
Name='appendicite'
Range=[0 1]
NumMFs=1
MF1='is':'trimf',[0.5 1 1]
[Output2]
Name='pielonefrit'
Range=[0 1]
NumMFs=1
MF1='is':'trimf',[0.5 1 1.4]
[Output3]
Name='acute'
Range=[0 1]
NumMFs=1
MF1='is':'trimf',[0 0 0.5]
[Output4]
Name='chronic'
Range=[0 1]
NumMFs=1
MF1='is':'trimf',[0.5 1 1.4]
[Rules]
1 1 2, 1 0 0 1 (1) : 1
1 2 2, 0 1 0 1 (1) : 1
2 1 1, 1 0 1 0 (1) : 1
2 2 1, 0 1 1 0 (1) : 1
```

Наведена система нечіткого виведення буде на вході мати три вхідні змінні, для яких будуть визначатися ступені приналежності до нечітких термів: 'normal' (нормальна) та 'high' (підвищена) для 'temperature', 'stomach' (у животі) та 'spine' (у спині) для 'painlocation', 'acute' (сильний) та 'weak' (слабкий) для 'paintype', відповідно. Нечітка модель на виході буде мати чотири вихідні змінні, кожна з яких буде характеризуватися одним нечітким термом 'is' (є).

Пропонована система нечіткого виведення також може мати альтернативне подання, коли вхідні змінні та вхідні терми є тими самими, як і в попередньому випадку, а вихідна змінна – одна 'diagnosis' (діагноз), яка характеризується приналежностями до чотирьох нечітких термів 'AcuteAppendicit' (гострий аппендицит), 'ChonicAppendicit' (хронічний аппендицит), 'AcutePielonefrit' (гострий пієлонефрит), 'ChronicPielonefrit' (хронічний пієлонефрит), відповідно.

Вміст файлу Sample1.fis з альтернативним поданням нечіткої моделі наведено нижче.

```
[System]
Name='Sample1'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=4
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'
[Input1]
Name='temperature'
Range=[35 42]
NumMFs=2
MF1='normal':'trimf',[35.9 36.6 37]
MF2='high':'smf',[36.8 37.2]
[Input2]
Name='painlocation'
Range=[0 1]
NumMFs=2
MF1='stomach':'trimf',[0 0 0.5]
MF2='spine':'trimf',[0.5 1 1]
[Input3]
Name='paintype'
Range=[0 1]
NumMFs=2
MF1='acute':'trimf',[0.5 1 1]
MF2='weak':'trimf',[0 0 0.5]
[Output1]
Name='diagnosis'
Range=[0 1]
NumMFs=4
MF1='AcuteAppendicit':'trimf',[0 0.125 0.25]
MF2='ChonicAppendicit':'trimf',[0.25 0.375 0.5]
MF3='AcutePielonefrit':'trimf',[0.5 0.625 0.75]
MF4='ChronicPielonefrit':'trimf',[0.75 0.875 1]
[Rules]
1 1 2, 2 (1) : 1
1 2 2, 4 (1) : 1
2 1 1, 1 (1) : 1
2 2 1, 3 (1) : 1
```

? 3.15 Контрольні питання

1. Що таке та у чому проявляється нечіткість знань?
2. Де і для яких задач доцільно та недоцільно застосовувати нечітку логіку? Які теореми це підтверджують?
3. Що таке: нечітка множина, чітка підмножина, функція приналежності, нечітка підмножина, множина приналежностей, нечітка змінна, лінгвістична змінна, терм-множина, терм, нечіткий терм, семантична процедура? Наведіть приклади цих понять.
4. Які існують методи побудови та запису функцій приналежності?
5. Як візуалізувати функції приналежності нечітких множин?
6. Які основні типи функцій приналежності реалізовано у пакеті MATLAB? Наведіть їх аналітичний запис, формат виклику у пакеті MATLAB та побудуйте графіки.
7. Дайте визначення характеристик та властивостей нечітких множин: висота, нормальність, субнормальність, порожність, унімодальність, ядро, носій, скінченність, нескінченність, межі, точки переходу, найближча чітка множина, опуклість, міра нечіткості Ягера, лінійний індекс нечіткості, квадратичний індекс нечіткості, ентропія, міра нечіткості Коско, кардинальне число, чітка множина альфа-рівня.
8. Дайте визначення операцій над нечіткими множинами: доповнення, інволюція, включення, непорівненість, рівність, власна нечітка підмножина,

об'єднання, перетинання, різниця, симетрична різниця, диз'юнктивна сума, алгебраїчний добуток, алгебраїчна сума, обмежена сума, обмежена різниця, обмежений добуток, драстичне перетинання, драстичне об'єднання, λ-сума, зведення в ступінь, CON, DIL, множення на число, опукла комбінація, декартовий добуток, нормалізація, імплікація, включення, еквівалентність, оператор збільшення нечіткості. Які нечіткі операції реалізовано у пакеті MATLAB?

9. Які мають властивості нечіткі операції та які існують закони нечіткої логіки?

10. Які існують узагальнення нечітких операцій?

11. Які існують визначення *T*-норми та *S*-конорми?

12. Що таке: нечітка величина, нечіткий інтервал, нечітке число, нечіткий нуль, позитивне і негативне нечіткі числа, нечітке число (*L*-*R*)-типу, функція приналежності нечітких чисел (*L*-*R*)-типу, нечітке n-арне відношення, нечітке відношення на множині?

13. У чому відмінності звичайних та нечітких чисел?

14. Як визначаються операції над нечіткими числами?

15. Що таке: n-арне відношення, повне та пусте нечіткі відношення, нечіткий предикат, предметні змінні та предметна область нечіткого предикату?

16. Які існують способи задавання нечітких відношень?

17. За допомогою яких показників характеризують нечіткі відношення?

18. Що таке: мода нечіткого відношення, скінчене й нескінчене, рефлексивне й антирефлексивне, симетричне й асиметричне, зворотне й транзитивне нечіткі відношення, транзитивне замикання нечіткого відношення?

19. Як визначаються операції над нечіткими відношеннями? Що таке звичайне відношення, найближче до нечіткого?

20. Як визначаються згортки: (max-*)-композиція, (min-*)-композиція та (sum-prod)-композиція? Які властивості вони мають?

21. Що таке нечітка база знань, правила Мамдані, правила Такагі-Сугено, антецедент і консеквент правила, імплікація, нечітке логічне виведення, нечіткий «modus ponens»?

22. Які існують етапи та з яких компонентів складаються системи нечіткого виведення? На які типи класифікують системи нечіткого виведення?

23. У чому полягають висхідний та низхідний методи нечіткого виведення? Опишіть основні етапи методів Мамдані, Цукамото, Сугено та Ларсена.

24. Що таке фазифікація та дефазифікація? Дайте визначення основних методів дефазифікації: середній з максимальних, найбільший з максимальних, найменший з максимальних, центр тяжіння, метод медіани, висотна дефазифікація.

25. Що таке: клас, кластер, кластерний аналіз, задача кластеризації, норма, евклідова норма, діагональна норма, норма Махаланобіса, індекс Хіє-Бені?

26. На які різновиди класифікують методи кластер-аналізу?

27. У чому полягають методи нечітких с-середніх, Густавсона-Кесселя, гірської та субтрактивної кластеризації?

28. Які існують засоби для створення нечітких моделей у пакеті MATLAB? Що таке структура FIS?

3.16 Практичні завдання

✍ *Завдання 1.* Нехай V – номер варіанта студента. Визначимо:

$$V_1 = \begin{cases} 4V, \text{якщо } V < 5; \\ V, \text{якщо } 5 \le V \le 10; \\ \text{round}(0{,}3V), \text{якщо } V > 10, \end{cases}$$

де round(x) – функція округлення.

Сформувати нечіткі множини:

$$A = \{< \frac{k}{V_1} \mid k >\}, \; B = \{< \frac{2k}{V_1} \mid 0{,}5k >\}, \; k = 0, 1, ..., V.$$

✍ *Завдання 2.* Для кожної з нечітких множин A та B визначити: висоту, чи є множина нормальною, субнормальною або порожньою; чи є множина унімодальною, ядро, носій, межі, точки переходу, найближчу чітку множину, чи є множина опуклою, міру нечіткості Ягера (у метриках: $p = 1$ та $p = 2$), міру нечіткості Коско, чітку множину альфа-рівня.

✍ *Завдання 3.* Визначити результати виконання над нечіткими множинами A та B операцій: доповнення, включення, рівність, об'єднання, перетинання, різниця, симетрична різниця, диз'юнктивна сума, алгебраїчний добуток, алгебраїчна сума, обмежена сума, обмежена різниця, обмежений добуток, драстичне перетинання, драстичне об'єднання, λ-сума, зведення в ступінь, CON, DIL, множення на число, опукла комбінація, нормалізація, нечітке включення. У якості скалярного числа де потрібно використовувати число V_1 або V_1^{-1}.

✍ *Завдання 4.* Придумати приклад лінгвістичної змінної та варіантів її значень. Для лінгвістичної змінної запропонувати нечіткі змінні. Навести приклади можливих значень нечітких змінних. Визначити терм-множину лінгвістичної змінної.

💻 *Завдання 5.* Побудувати декілька разів графіки функцій приналежності у залежності від номеру варіанту із різними значеннями параметрів. Параметри функцій приналежності задати самостійно. Дослідити, як впливають значення параметрів на зміну значення функцій приналежності. Завдання виконати у пакеті MATLAB із використанням функцій модуля Fuzzy Logic Toolbox.

V	Функції		*V*	Функції	
1.	dsigmf	gbellmf	14.	psigmf	gaussmf
2.	dsigmf	zmf	15.	psigmf	pimf
3.	dsigmf	trimf	16.	sigmf	dsigmf
4.	gauss2mf	gbellmf	17.	sigmf	pimf
5.	gauss2mf	trimf	18.	sigmf	trimf
6.	gauss2mf	trapmf	19.	smf	gauss2mf
7.	gaussmf	gauss2mf	20.	smf	gbellmf
8.	gaussmf	pimf	21.	trapmf	gaussmf
9.	gaussmf	trapmf	22.	trimf	gauss2mf
10.	gbellmf	pimf	23.	trimf	gbellmf
11.	gbellmf	smf	24.	zmf	smf
12.	pimf	gaussmf	25.	zmf	dsigmf
13.	pimf	sigmf	26.	zmf	pimf

Завдання 6. Для множин *A* та *B* за допомогою пакету MATLAB визначити результати нечітких операцій: 'sum' – сума, 'sub' – вирахування, 'prod' – добуток, 'div' – ділення. Результати зобразити графічно у пакеті MATLAB. При виконанні завдання використовувати функції приналежності з завдання 5.

Завдання 8. Придумати приклад нечіткого відношення *A R B*, а також його доповнення $\overline{R}$. Визначити результати операцій:

$$C = R \cup \overline{R}\,,\ D = R \cap \overline{R}\,,$$
$$E = (C \cap D) \cup R\,.$$

Завдання 9. Для нечітких множин *A* та *B* відповідно до *V* знайти чіткі значення A^* та B^*, використовуючи методи дефазифікації: середній з максимальних, найбільший з максимальних, найменший з максимальних, центр тяжіння, метод медіани, висотна дефазифікація. Порівняти результати методів для кожної нечіткої множини.

Завдання 10. Використовуючи продукційну модель бази знань експертної системи побудувати її спрощений варіант у вигляді правил «Якщо-то». Використовувати не більше чотирьох вхідних змінних та п’яти значень вихідної змінної. Задати не більше восьми правил. У редакторі FIS пакету MATLAB побудувати систему нечіткого виведення для створених змінних і правил. Переглянути результати роботи FIS для позитивного та негативного сценаріїв. Зберегти створену FIS у файлі на диску.

Завдання 11. Підготувати реферат на одну з таких тем.

1. Методи визначення функцій приналежності нечітких множин.

2. Операції над нечіткими множинами та відношеннями.
3. Закони чіткої та нечіткої логік.
4. Узагальнення нечітких операцій.
5. Нечіткі числа та операції з ними.
6. Методи нечіткого виведення.
7. Методи дефазифікації.
8. Нечіткий кластерний аналіз.
9. Програми для нечіткого моделювання.

3.17 Література до розділу

Основні визначення і поняття нечіткої логіки, а також опис операцій та властивостей нечітких множин наведено у [2, 9–11, 13, 14, 20, 22, 24–28, 33, 41, 44–46, 48, 49, 60, 61, 75–78, 80, 83, 94, 96, 97, 100, 102, 105, 107–110, 113, 115, 117, 121, 123, 124, 127, 130, 131, 134, 138, 144, 148, 149, 151–155, 157, 158, 160–162, 189, 198, 204–206], моделі та методи нечіткого виведення описано в [2, 13, 14, 20, 34, 40, 42, 49, 97, 105, 108, 110, 113, 128, 131, 144, 160, 163, 164, 167, 186, 197, 200, 205], кластер-аналізу присвячено [8, 22, 53, 54, 64, 86, 105, 107, 164, 169, 170, 199], програмні засоби для побудови нечітких моделей розглянуто в [2, 49, 62, 105, 205].

Навчально-методичні матеріали до розділу доступні на сайті автора в мережі Інтернет за адресою: `http://csit.narod.ru`.

4. НЕЙРО-НЕЧІТКІ МЕРЕЖІ ДЛЯ ПОДАННЯ Й ОБРОБКИ ЗНАНЬ

4.1 Загальна характеристика та властивості нейро-нечітких мереж

Різні типи інтелектуальних систем мають свої особливості, наприклад, за можливостями навчання, узагальнення і отримання результатів, що робить їх найбільш придатними для вирішення одних класів задач і менш придатними – для інших.

Нейронні мережі, наприклад, є зручними для задач розпізнавання образів, але дуже незручні для пояснення, як вони таке розпізнавання здійснюють. Вони можуть автоматично здобувати знання, але процес їхнього навчання найчастіше відбувається досить повільно, а аналіз навченої мережі є дуже складним (навчена мережа є звичайно «чорною скринею» для користувача). При цьому яку-небудь апріорну інформацію (знання експерта) для прискорення процесу навчання в нейронну мережу ввести складно.

Системи з нечіткою логікою, напроти, є зручними для пояснення одержуваних за їхньою допомогою висновків, але вони не можуть автоматично здобувати знання для використання їх у механізмах виведень. Необхідність розбиття універсальних множин на окремі області, як правило, обмежує кількість вхідних змінних у таких системах невеликим значенням.

Хаяши (Y. Hayashi) та Імура (A. Imura) показали, що нейромережа прямого поширення може апроксимувати будь-яку систему, що заснована на нечітких правилах, та будь-яка нейромережа прямого поширення може бути апроксимована системою, що заснована на нечітких правилах.

Узагалі говорячи, теоретично, системи з нечіткою логікою і штучні нейронні мережі подібні одні одним, однак, відповідно до викладеного вище, на практиці в них є власні переваги і недоліки.

Дане розуміння лягло в основу створення апарату *нечітких нейронних мереж*, у яких виведення робляться на основі апарату нечіткої логіки, але відповідні функції приналежності підбудовуються з використанням методів навчання нейронних мереж, наприклад, методу зворотного поширення помилки. Такі системи не тільки використовують апріорну інформацію, але можуть здобувати нові знання, будучи *логічно прозорими*.

Нейро-нечітка мережа – це подання системи нечіткого виведення у вигляді нейронної мережі, зручної для навчання, поповнення, аналізу та використання. Структура нейро-нечіткої мережі відповідає основним блокам систем нечіткого виведення.

Основними *властивостями нейро-нечітких мереж* є те, що:

– нейро-нечіткі мережі засновані на нечітких системах, які навчаються за допомогою методів, використовуваних у нейромережах;

– нейро-нечітка мережа зазвичай є багатошаровою (частіше – тришаровою) нейронною мережею. Перший шар становить вхідні змінні, середній становить нечіткі правила, а третій – вихідні змінні. Ваги підключення відповідають нечітким множинам вхідних і вихідних змінних. Іноді використовується п'ятишарова архітектура. В загальному випадку нечітка система необов'язково повинна бути подана в такому вигляді, однак це є зручною моделлю для застосування навчаючих методів;

– нейро-нечітка мережа завжди (до, під час і після навчання) може бути інтерпретована як система нечітких правил;

– процедура навчання враховує семантичні властивості нечіткої системи. Це виражається в обмеженні можливих модифікацій, які застосовуються до параметрів, що налагоджуються. Потрібно, однак, сказати, що не всі методи навчання нейро-нечітких мереж враховують семантику системи;

– нейро-нечітка система апроксимує $N \times M$ – розмірну невідому функцію, що частково описана навчаючими даними.

Типи поєднання нечіткої логіки і нейронних мереж за способом взаємодії виділяють такі:

– *нечіткі нейронні системи* (fuzzy neural systems). В цьому випадку в нейронних мережах застосовуються принципи нечіткої логіки для прискорення процесу налагодження або поліпшення інших параметрів. При такому підході нечітка логіка є лише інструментом нейронних мереж і така система не може бути інтерпретована в нечіткі правила, оскільки являє собою «чорну скриню»;

– *конкуруючі нейро-нечіткі системи* (concurrent neuro-fuzzy systems). У таких моделях нечітка система і нейронна мережа працюють над однією задачею, не впливаючи на параметри одна одної. Можлива послідовна обробка даних спочатку однією системою, потім іншою;

– *паралельні нейро-нечіткі системи* (cooperative neuro-fuzzy systems). У таких системах налагодження параметрів виконується за допомогою нейронних мереж. Далі нечітка система функціонує самостійно. Виділяють такі *типи паралельних нейро-нечітких моделей*: 1) нечітка асоціативна пам'ять (fuzzy associative memory); 2) системи із виділенням нечітких правил шляхом використання карт, що самоорганізуються (fuzzy rule extraction using self-organizing maps); системи, здатні навчати параметри нечітких множин (systems capable of learning fuzzy set parameters);

– *інтегровані* (*гібридні*) *нейро-нечіткі системи* (integrated neuro-fuzzy systems) – системи з тісною взаємодією нечіткої логіки і нейронних мереж. Під терміном «нейро-нечіткі мережі» частіше за все мають на увазі системи саме цього типу. Як правило інтегровані системи є системами типу Мамдані або Такагі-Сугено.

За способом відображення нечітких множин в структурі мережі нейро-нечіткі мережі бувають трьох основних типів:

– *системи, побудовані на вибіркових нечітких множинах*. В таких системах ступені приналежності описані лише для деяких значень з області визначення і функція приналежності подана у вигляді вектору. Кожному ступеню приналежності відповідає лише один вхідний або вихідний нейрон. Існує два підходи до реалізації таких систем. В першому система просто апроксимує відповідність виходів входам, така система є «чорною скринею». В другому створюється система із спеціальною архітектурою, в якій втілюються нечіткі правила;

– *системи, параметризовані функції* приналежності яких зберігаються в нейронах. Прикладом таких систем є ANFIS;

– *системи, в яких параметризовані функції приналежності використовуються як ваги зв'язків між нейронами*. Таку систему інакше можна назвати персептроном з нечіткими зв'язками або *нечітким персептроном*. Прикладами таких систем є NEFCON, NEFCLASS, NEFPROX.

За характером навчання виділяють такі *типи нейро-нечітких мереж* :

– *самоналагоджувані нейро-нечіткі мережі* – з адаптацією структури та параметрів;

– *адаптивні нейро-нечіткі мережі* – із жорсткою структурою та адаптацією параметрів мережі.

Адаптивні нейро-нечіткі мережі *за видом методу оптимізації* поділяють на такі, що використовують детерміновані методи типу градієнтного пошуку, та такі, що використовують стохастичні методи, зокрема еволюційні.

Адаптивні нейро-нечіткі мережі *за типом параметрів адаптації* поділяють на мережі з адаптацією параметрів функцій приналежності, мережі з адаптацією ваг правил та мережі з адаптацією параметрів оператора агрегації.

4.2 Формування бази знань нейро-нечіткої мережі

Розглянемо об'єкт виду $y = f(x_1, x_2, ..., x_n)$ для якого зв'язок «входи x_i – вихід y» можна подати у вигляді *експертної матриці знань* (табл. 4.1).

Цій матриці відповідає нечітка база знань:

$$\text{Якщо } [(x_1 = a_1^{j1}) \text{ та } (x_2 = a_2^{j1}) \text{ та } \dots (x_n = a_n^{j1})] \text{ (з вагою } w_{j1}) \dots$$

$$\dots \text{ або } [(x_1 = a_1^{jk_j}) \text{ та } (x_2 = a_2^{jk_j}) \text{ та } \dots (x_n = a_n^{jk_j})] \text{ (з вагою } w_{jk_j}),$$

$$\text{то } \quad y = d_j, \quad j = 1, 2, ..., m, \text{p} = 1, 2, \dots, k_j,$$

де a_i^{jp} – лінгвістичний терм, що оцінює змінну x_i у рядку p; p = 1, 2,…,k_j; k_j – кількість рядків-кон'юнкцій, що відповідають класу d_j вихідної змінної y, w_{jp} – число в діапазоні [0, 1], що характеризує суб'єктивну міру впевненості експерта щодо висловлення з номером p.

Якщо вихідна змінна (консеквент) є дискретною, то класи d_j, j = 1, 2, ..., m, формуються як номери дискретних значень вихідної змінної.

Таблиця 4.1 – Експертна матриця знань

Номер правила	Якщо (входи)				То (вихід)	Вага правила
	x_1	x_2	...	x_n		
11	a_1^{11}	a_2^{11}	...	a_n^{11}	d_1	w_{11}
12	a_1^{12}	a_2^{12}	...	a_n^{12}		w_{12}
...	...	...	...	...		...
$1k_1$	$a_1^{1k_1}$	$a_2^{1k_1}$	...	$a_n^{1k_1}$		w_{1k_1}
...	...	...	...	...	...	...
$m1$	a_1^{m1}	a_2^{m1}	...	a_n^{m1}	d_m	w_{m1}
$m2$	a_1^{m2}	a_2^{m2}	...	a_n^{m2}		w_{m2}
...	...	...	...	...		...
mk_m	$a_1^{mk_m}$	$a_2^{mk_m}$	...	$a_n^{mk_m}$		w_{mk_m}

Якщо вихідна змінна є дійсною, то класи d_j, $j = 1, 2, ..., m$, формуються шляхом квантування діапазону $[\underline{y}, \overline{y}]$ вихідної змінної y на m рівнів:

$$[\underline{y}, \overline{y}] = \underbrace{[\underline{y}, y_1]}_{d_1} \cup \ldots \cup \underbrace{[y_{j-1}, y_j]}_{d_j} \cup \ldots \cup \underbrace{[y_{m-1}, \overline{y}]}_{d_m}.$$

Основні принципи формування бази знань нейро-нечітких систем полягають у наступному.

Копіювання навчаючої вибірки в базу знань – для кожного екземпляра навчаючої вибірки формується окреме правило. Перевагою даного методу є простота та висока швидкість роботи, недоліком – відсутність узагальнюючих властивостей і громіздкість одержуваної мережі.

Оптимізація кількості продукційних правил – знаходження такого значення кількості продукційних правил S, при якому значення помилки $E(S)$ є мінімальним, для чого при різних значеннях S навчають мережу і вимірюють значення помилки, після чого оптимізують функцію $E(S)$ за параметром S. Недоліком даного методу є дуже високі вимоги до обчислювальних ресурсів, обумовлені необхідністю заново навчати мережу на кожному кроці.

Спільна оптимізація ваг мережі та кількості продукційних правил шляхом вирішення багатоекстремальної оптимізаційної задачі або автоматичне визначення числа кластерів у навчаючій вибірці та встановлення центрів функцій приналежності в їхні центри на основі кластер-аналізу.

Скорочення (*редукція*) *правил.* В методах скорочення при ініціалізації формується нечітка система, що містить свідомо надлишкове число продукційних правил. У процесі роботи методу зайві продукційні правила виключаються.

Основні принципи редукції правил:

– скорочення нечітких правил відповідно до їхніх логічних функцій: 1) виключення правил, для яких результуюча функція приналежності менше визначеного порога, як таких, що мало впливають на остаточний результат; 2) виключення суперечливих правил, які взаємно компенсуються; 3) виключення одного з двох співпадаючих правил, як таких, що не несуть нової інформації;

– ортогоналізація: видалення тих продукційних правил, вплив яких на точність виявляється мінімальним після оцінки індивідуального внеску кожного продукційного правила у вихідний сигнал мережі, одержуваної шляхом використання ортогонального методу найменших квадратів.

Істотним недоліком методів скорочення є необхідність спочатку працювати зі свідомо надлишковою за розміром базою знань, що обумовлює в ряді випадків повільну роботу методів.

Нарощування (*конструювання*) *правил*: формується початкова база продукційних правил (вона може бути і порожньою), що потім послідовно поповнюється нечіткими правилами.

Виділяють два різновиди методів нарощування:

1) При надходженні чергової навчаючої пари $<x^s,y^s>$ визначають відстані $d(x^s, C^v)$ між центрами передумов наявних правил і точкою x^s. Якщо $\min(d(x^s, C^v))> d$, $v = 1, 2, \ldots, V$, де d – деякий параметр, що можливо змінюється в процесі роботи методу, то додається ще одне правило з центрами функцій приналежності у точці $C^{V+1} = x^s$ та висновком y^s. У випадку невиконання зазначеної нерівності додавання продукційного правила не відбувається.

Недоліком даного методу є відсутність явного зв'язку між процедурою додавання продукційних правил і точністю апроксимації, що повинна визначатися окремо.

2) Нове правило додається при виконанні нерівності $|y^{s*} – y^s|>\varepsilon$, де y^{s*} – значення виходу мережі при подачі на вхід x^s, розраховане за наявними продукційними правилами, ε – постійний параметр, що характеризує точність апроксимації.

Прикладом цього підходу є метод формування бази знань шляхом генерації нових продукційних правил, що не суперечать правилам з бази знань системи, виходячи з аналізу експериментальних даних про об'єкт.

Нехай ми маємо вибірку S пар значень $<x^s, y^s>$, $s = 1, 2, ..., S$; значення S при необхідності допускає модифікацію. Тоді метод формування бази знань може бути описаний у такий спосіб.

Крок 1. З m ($m < S$) довільних значень $<x^s, y^s>$ складається початкова база знань моделі, відображувана матрицею $U_{m\times(N+1)}$ з рядками $<x^s,y^s> = <x^s_1, x^s_2, ... x^s_N, y^s>$. Таке подання еквівалентно набору продукційних правил:

$П_s$: якщо $x_1 \in A^s_1$ та $x_2 \in A^s_2$ та ... та $x_N \in A^s_N$, то $y = y^s$, $s = 1, ..., m$.

Крок 2. Для кожної нової експериментальної точки $<x^*,y^*>$ розраховується прогнозоване значення за формулою, що відповідає центроїдному методу:

$$y^*_{\text{р.}} = \frac{\sum\limits_{s=1}^{m} y^s \mu(\left\| x^s - x^* \right\|)}{\sum\limits_{s=1}^{m} \mu(\left\| x^s - x^* \right\|)},$$

де μ – функція колоколообразної або експонентної форми:

$$\mu(\left\| x^s - x^* \right\|) = \exp(-\lambda \sum_{j=1}^{N} \left| x^s_j - x^*_j \right|),$$

де λ – параметр функції.

Крок 3. Якщо $| y^* - y^*_{\text{р.}} | > \varepsilon$, де ε – задана константа, що визначає погрішність апроксимації, тоді база знань поповнюється шляхом розширення матриці U (додаванням рядка $<x^*, y^*>$), у протилежному випадку – матриця U залишається без змін.

Крок 4. Перевіряється правило зупинення. У даному варіанті методу побудова моделі вважається закінченою, якщо відповідно до кроків 2 та 3 перебрані всі S експериментальних точок (без урахування значень початкової бази знань). Якщо не всі експериментальні точки використані, то здійснюється перехід до кроку 2, у протилежному випадку – зупинення.

У процесі реалізації методу параметри λ та ε вважаються наперед заданими. При використанні системи заданими вважаються матриця U (на етапі використання моделі вона не змінюється) та параметри λ й ε, а розрахунок $y^*_{p.}$ здійснюється відповідно до кроку 2 наведеного методу.

Легко бачити, що описаний метод, по суті, відповідає спрощеному методу нечіткого логічного виведення, але відрізняється від останнього тім, що база знань не залишається фіксованою, а модернізується в міру надходження експериментальних даних. Причому несуперечність нового продукційного правила щодо набору правил з бази знань гарантується процедурою її поповнення.

4.3 Елементи нейро-нечітких мереж

Розглянемо просту нейронну мережу, що складається з одного нейрона з двома входами (рис. 4.1).

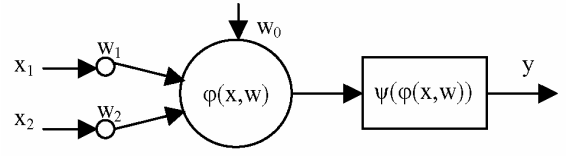


Рисунок 4.1 – Схема штучного нейрона

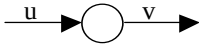
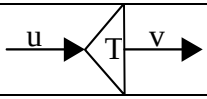
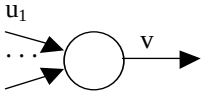
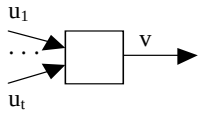
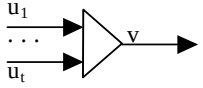
Вхідні сигнали x_j «взаємодіють» із *синаптичними вагами* w_j, утворюючи значення *рівня збудження нейрона* $\varphi(x,w) = w_1x_1+w_2x_2+w_0$, де $\varphi(x,w)$ – *функція постсинаптичного потенціалу* (дискримінантна, вагова функція) нейрона, w_0 – *поріг нейрона*. Вихід нейрона у утворюється в результаті перетворення значення $\varphi(x,w)$ *функцією активації* ψ:

$$y = \psi(\varphi(x,w)) = \psi(w_1 x_1 + w_2 x_2 + w_0).$$

Розглянута однонейронна мережа, у якій використовуються операції множення, підсумовування і функція активації є *стандартною нейронною мережею*. У випадку застосування замість операцій множення, підсумовування й функцій активації таких операцій, як *t*-норма і *t*-конорма дану нейронну мережу будемо називати *нечіткою*. Входи, виходи і ваги нечіткої нейронної мережі – дійсні числа, що належать відрізку [0, 1].

У табл. 4.2 наведено опис основних елементів нечітких нейромереж. Тут використовуються такі позначення: $\mu_T(x)$ – функція приналежності змінної x терму T; d_j – центр j-го класу правил, $d_j \in [\underline{y}, \overline{y}]$, де $\underline{y}$ та $\overline{y}$ – нижня і верхня межі значень вихідної змінної.

Таблиця 4.2 – Елементи нечітких нейромереж

Назва елемента	Зображення елемента	Функція елемента
Вхід		$v = u$
Нечіткий терм		$v = \mu_T(u)$
Нечітке правило		$v = \min\limits_{i=1,2...,t} u_i$ $v = \prod\limits_{i=1}^{t} u_i$
Клас правил		$v = \max\limits_{i=1,2...,t} u_i$ або $v = \sum\limits_{i=1}^{t} u_i$
Дефазифікація		$v = \sum\limits_{j=1}^{m} u_j d_j \Big/ \sum\limits_{j=1}^{m} u_j$

Розглянемо приклади *елементів нечітких нейронних мереж*.

Нечіткий нейрон «ТА». Сигнали x_i та ваги w_i у даному випадку поєднуються за допомогою *t*-конорми: $p_j = S(w_j, x_j)$, $j = 1, 2$, а вихід утворюється

з застосуванням t-норми: $y = \text{AND}(p_1, p_2) = T(p_1, p_2) = T(S(w_1, x_1), S(w_2, x_2))$. Якщо прийняти $T = \min$, $S = \max$, то нечіткий нейрон «ТА» реалізує композицію min-max: $y = \min(\max(w_1, x_1), \max(w_2, x_2))$.

Нечіткий нейрон «АБО». Сигнали x_j і ваги w_j тут поєднуються за допомогою t-норми: $p_j = T(w_j, x_j)$, $j = 1, 2$, а вихід утворюється з застосуванням t-конорми: $y = \text{OR}(p_1, p_2) = S(p_1, p_2) = S(T(w_1, x_1), T(w_2, x_2))$. Якщо прийняти $T = \min$, $S = \max$, то нечіткий нейрон «АБО» реалізує композицію max-min: $y = \max(\min(w_1, x_1), \min(w_2, x_2))$.

4.4 Паралельні нейро-нечіткі системи

У найпростішому випадку паралельна модель (рис. 4.2) може розглядатися як передоброблювач, де механізм навчання штучної нейронної мережі визначає функції приналежності або нечіткі правила системи нечіткого виведення за навчаючими даними. Як тільки параметри системи нечіткого виведення визначено, нейронна мережа припиняє використовуватися.

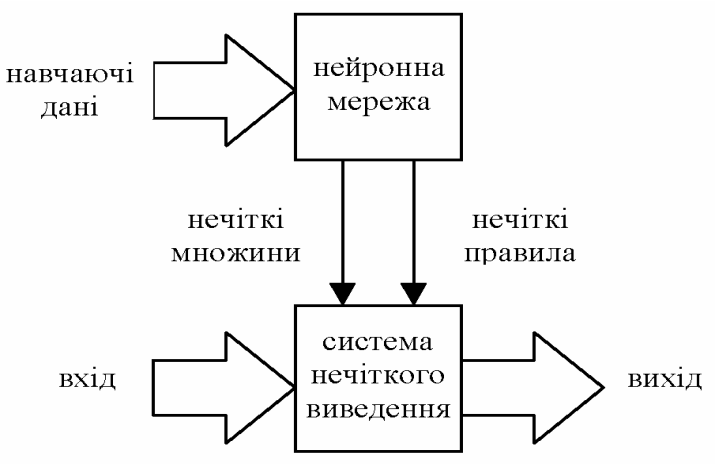


Рисунок 4.2 – Паралельна нейро-нечітка модель

База правил нечіткої системи звичайно визначається на основі карт, що самоорганізуються, або методів нечіткої кластеризації. Функції приналежності звичайно апроксимуються нейронною мережею за навчаючими даними.

Найбільш відомими прикладами паралельних нейро-нечітких систем є: нечітка асоціативна пам’ять Коско (B. Kosko), виділення нечітких правил на

основі карт Педрича (W. Pedrycz), що самоорганізуються, та системи Номури (H. Nomura), здатні до навчання параметрів нечітких множин.

Нейро-нечітка мережа FAM (Kosko Fuzzy Associative Memory – нечітка асоціативна пам'ять Коско) заснована на інтерпретації нечіткого правила як асоціації між антецедентом і консеквентом. Якщо нечітка множина подається як точка в одиничному гіперкубі, а правила є асоціаціями, то можливо використовувати нейронну асоціативну пам'ять для збереження нечітких правил. Нейронна асоціативна пам'ять може бути подана її матрицею зв'язків.

Асоціативний пошук еквівалентний множенню ключового фактора на цю матрицю. Ваги зберігають кореляції між ознаками ключа й інформаційною частиною. Через обмежений обсяг асоціативної пам'яті і через небажаність об'єднання багатьох матриць зв'язків в одну матрицю через серйозну втрату інформації необхідно зберігати кожне нечітке правило в окремій мережі FAM. Правила з N змінними, з'єднаними за допомогою кон'юнкцій в антецедентах, можуть бути подані за допомогою N мереж FAM, де кожна мережа зберігає одне правило. Побудова мережі FAM для всього набору правил завершується об'єднанням усіх виходів операцією максимуму і дефазифікацією результату.

Навчання може бути включене в FAM як навчання ваг, зв'язаних з виходом FAM, або шляхом створення FAM цілком за допомогою навчання. Метод навчання нейронної мережі визначає ваги правил для нечітких правил. Такі показники часто інтерпретуються як вплив правил та перемножуються з виходами правил.

Ваги правил можуть бути замінені еквівалентною модифікацією функцій приналежності. Однак це може привести до невірної інтерпретації нечітких множин та того, що ідентичні лінгвістичні значення могли б бути по різному подані в різних правилах. Коско запропонував метод адаптивного квантування вектора для навчання FAM. Цей метод названий диференціальним конкурентним навчанням і подібний навчанню карт самоорганізації.

Адаптивна пам'ять Коско є паралельною нейро-нечіткою моделлю, оскільки вона використовує навчаючий метод для визначення правил і їхніх ваг. Головною незручністю FAM є зважування правил. Те, що деякі правила не мають сильного впливу, не означає, що вони є малозначимими. Отже, надійність FAM для деяких застосуваньдодатків є сумнівною. Однак через простоту реалізації FAM широко використовується в багатьох застосуваннях.

Виділення нечітких правил з використанням карт, що самоорганізуються (Fuzzy Rule Extraction Using Self Organizing Maps). Педрич запропонував використання карт самоорганізації із змагаючимся шаром для кластеризації навчаючих даних і розробив засоби для інтерпретації результатів навчання.

Результати навчання можуть показувати, чи є два вхідних вектори подібними один одному або чи належать вони до одного й того ж самого класу. Однак, у випадку багатомірних вхідних векторів, структура задачі навчання

може рідко виявлятися в двомірній карті. Передбачено процедуру інтерпретації результатів навчання з використанням лінгвістичних змінних.

Після процесу навчання матриця ваг *w* подає вагу кожної вхідної ознаки стосовно до виходу. Така матриця визначає карту тільки для однієї ознаки. Для кожної вхідної ознаки відповідно до лінгвістичного опису B визначаються нечіткі множини (одна нечітка множина для кожної змінної). Вони застосовуються до матриці ваг *w* для одержання множини перетворених матриць.

Кожна комбінація лінгвістичних термів – це можливий опис підмножини або екземплярів кластера. Щоб перевірити лінгвістичний опис *B* на придатність перетворені карти схрещуються і виходить матриця *D*, що визначає сумісність результату навчання з лінгвістичним описом *B*. *D*(*B*) – це нечітке відношення, яке інтерпретується як ступінь підтримки *B*.

Описуючи відношення *D*(*B*) його *a*-розтинами $D_a(B)$ одержують підмножини вихідних вузлів, ступінь приналежності яких є принаймні *a*, так, що достовірність приналежності всіх екземплярів x_a до класу, описаному *B*, скорочується зі зменшенням *a*. Кожний опис *B* є придатним описом кластера, якщо *D*(*B*) має непорожній *a*-розтин $D_a(B)$.

Якщо ознаки розподілені на вхідні і вихідні, то кожний опис *B* подає лінгвістичне правило, і, досліджуючи кожну комбінацію лінгвістичних значень, можна створити повну базу нечітких правил. Цей метод також показує, які зразки належать нечіткому правилу, оскільки вони не містяться в жодній підмножині x_a.

Важливою перевагою даного методу порівняно з FAM є те, що правила не зважуються. Проблемою є визначення кількості вихідних нейронів і значень *a* для кожної задачі навчання. Оскільки форма функції приналежності сильно впливає на якість роботи мережі дані можуть бути краще використані в порівнянні з FAM. Оскільки процедура навчання Коско не бере до уваги відношення сусідства між вихідними нейронами, гарне топологічне відображення вхідних даних до вихідного не завжди може бути отримано. Таким чином, навчаюча процедура FAM більше залежить від послідовності навчаючих даних ніж навчаюча процедура Педрича. Спочатку визначається структура простору ознак і потім, використовуючи доступні нечіткі розбиття, отримують лінгвістичні описи, що найкраще відповідають результатам навчання. Якщо велика кількість екземплярів не відповідає жодному з описів, це може пояснюватися невірним вибором функцій приналежності, і вони можуть бути перевизначені. Отже для навчання нечітких правил цей підхід є кращим у порівнянні з FAM.

Ефективність даного методу усе ще залежить від кроку навчання і розміру околиці для модифікації ваг, що у залежності від задачі визначаються евристично.

Системи, здатні навчати нечіткі множини. Номура (H. Nomura), Хаяши (I. Hayashi) та Вакамі (N. Wakami) запропонували метод контрольованого навчання для настроювання параметрів нечітких множин нечіткої системи Такагі-Сугено.

Метод навчання використовує процедуру градієнтного спуску, що використовує міру помилки *E* (різниця між фактичним і цільовим значенням виходу) для настроювання параметрів функцій приналежності. Дана процедура подібна дельта-правилу багатошарових персептронів. Навчання здійснюється в режимі реального часу. Для вхідного вектора обчислюється результуюча помилка *E*, заснована на зміні консеквентів правил. Потім для тих же екземплярів налагоджуються параметри функцій приналежності. Це робиться для того, щоб врахувати зміни в консеквентах, коли модифікуються антецеденти.

Серйозним недоліком даного методу є те, що подання лінгвістичних значень вхідних змінних залежить від правил, у яких вони з'являються. Спочатку ідентичні лінгвістичні терми подаються ідентичними функціями приналежності. Протягом процесу навчання вони можуть по різному змінюватися так, щоб ідентичні лінгвістичні терми були подані різними нечіткими множинами. Даний метод застосовується тільки до систем нечіткого виведення типу Такагі-Сугено.

Використовуючи подібний підхід, Мійоши (T. Miyoshi), Тано (S.Tano), Като (Y. Kato) та Арнольд (T. Arnould), налагоджували оператори *t*-норми і *t*-конорми, у той час, як Ягер (R. Yager) налагоджував оператор дефазифікації, використовуючи метод контрольованого навчання.

4.5 Конкурентні нейро-нечіткі системи

У конкурентній моделі нейронна мережа допомагає системі нечіткого виведення безупинно визначати необхідні параметри, особливо, якщо вхідні змінні не можуть бути безпосередньо вимірювані. Така комбінація мереж не оптимізує нечітку систему, але допомагає поліпшити експлуатаційні якості всієї конкуруючої системи. Навчання застосовується тільки для нейронної мережі, а нечітка система залишається незмінною протягом цієї стадії.

У деяких випадках виходи нечіткої системи безпосередньо не можуть бути застосовні до керованого процесу. У цьому випадку нейронна мережа може діяти як постпроцесор виходів нечіткої системи.

На рис. 4.3 зображено конкурентну нейро-нечітку систему, де вхідні дані подаються на входи нейронної мережі, а вихід нейронної мережі далі оброблюється нечіткою системою.

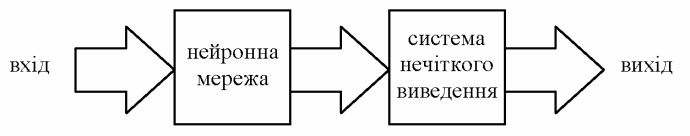


Рисунок 4.3 – Конкурентна нейро-нечітка система

4.6 Інтегровані нейро-нечіткі системи

В інтегрованій моделі для визначення параметрів системи нечіткого виведення використовуються методи навчання нейронних мереж. Інтегровані нейро-нечіткі системи розподіляють структури даних і подання знань.

Система нечіткого виведення може використовувати людські експертні знання, зберігаючи їхні істотні компоненти в базі правил і базі даних, та виконувати нечітке виведення для одержання вихідного значення. Формування правил і відповідних функцій приналежності сильно залежить від апріорного знання щодо розглянутої системи. Однак універсальний спосіб перетворення дослідних знань людських експертів у базу знань системи нечіткого виведення є невідомим. Є також потреба в адаптованості або деяких методах навчання для одержання виходів з необхідним рівнем точності.

З іншого боку, механізм навчання нейронних мереж не покладається на людську експертизу. Через однорідну структуру нейронних мереж, з них складно витягти структуроване знання. Ваги нейронної мережі являють собою коефіцієнти гіперплощини, що розподіляє вхідний простір на області з різними значеннями виходу. Якщо ми можемо візуалізувати цю структуру гіперплощини за навчаючими даними, тоді наступні процедури навчання в нейронній мережі можуть бути виключені. Однак, у дійсності, апріорне знання звичайно виходить від людських експертів, що більш відповідає уявленню знань у вигляді набору нечітких правил «якщо, то», і це складно закодувати в нейронну мережу.

У значній мірі недоліки цих двох підходів компенсуються. Тому, природно розглянути створення інтегрованої системи, що сполучить концепції нейромереж і систем нечіткого виведення.

Найбільш загальний спосіб застосування методу навчання до нечіткої системи полягає в тому, щоб подати її у вигляді архітектури, подібної нейронній мережі. Однак звичайні (градієнтні) методи навчання нейронних мереж не можуть безпосередньо застосовуватися до такої системи, оскільки функції, використовувані в процесі виведення звичайно є недиференційовані. Ця проблема може бути вирішена шляхом використання диференційованих функцій у системі виведення або не використання стандартних методів навчання нейромереж.

Очевидно, що паралельні та конкуруючі моделі не є цілком інтерпретованими через присутність нейронної мережі («чорна скриня»). У той же час інтегрована нейро-нечітка модель є інтерпретованою і здатна до контрольованого навчання (чи навіть до навчання з підкріпленням, подібно NEFCON). У FALCON, GARIC, ANFIS, NEFCON, SONFIN, FINEST і FUN процес навчання сконцентрований тільки на настроюванні значень параметрів у межах установлених структур. Для багатомірних задач буде складним визначити оптимальні структури «передумова-наслідок», кількість правил і т. д. Користувач повинний визначити деталі архітектури: тип і кількість функцій прина-

лежності для вхідної і вихідної змінних, тип нечітких операторів і т. д. FINEST містить механізм, заснований на поліпшеному узагальненому modus ponens для настроювання нечітких предикатів і комбінуючих функцій, а також настроювання функції імплікації. Важливою особливістю EFUNN і dmEFuNN є однопрохідне навчання в реальному режимі часу.

Серед інтегрованих нейро-нечітких моделей ANFIS має найбільшу точність, а NEFPROX – найменшу. Це пояснюється тим, що в ANFIS реалізовані правила Такагі-Сугено, а NEFPROX є нечіткою системою типу Мамдані. Однак, NEFPROX швидше працює в порівнянні з ANFIS. Через меншу кількість правил SONFIN, EFUNN і dmEFuNN також здатні працювати швидше ніж ANFIS. Отже, є протиріччя між інтерпретабельністю і точністю. Системи виведення типу Такагі-Сугено є більш точними, але вимагають більше обчислювальних витрат. У той час як системи виведення типу Мамдані є більш інтерпретабельними і вимагають менше обчислювальних витрат, але часто з компромісом за точністю.

Нейро-нечіткий апроксиматор Мамдані (Mamdani neuro-fuzzy approximator) є найбільш узагальненою моделлю нейро-нечіткої мережі, побудованої на правилах Мамдані. Структуру мережі Мамдані зображено на рис. 4.4.

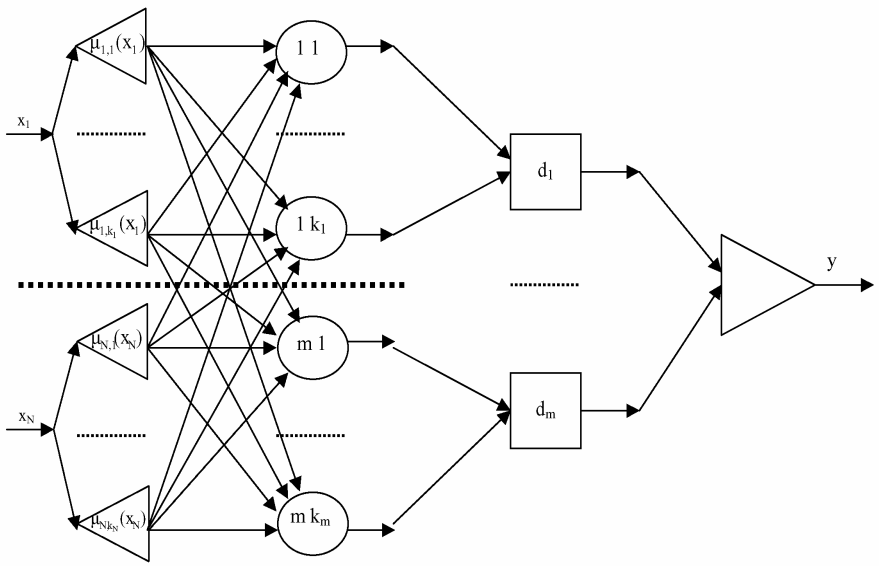


Рисунок 4.4 – Структура нейро-нечіткої мережі Мамдані

Нейро-нечітка мережа Мамдані має п’ять шарів.

Перший (вхідний) шар утворюють елементи вхідного вектора.

Другий шар містить нечіткі терми, які відповідають вхідним змінним, та обчислює приналежності вхідного вектора до кожного з нечітких термів:

$$\mu_{jp}(x_i) = \frac{1}{1 + \left(\frac{x_i - b_i^{jp}}{c_i^{jp}} \right)^2}, \quad i = \overline{1, N},\ p = \overline{1, k_j},\ j = \overline{1, m},$$

де $\mu_{ip}(x_i)$ – функція приналежності змінної x_i терму p-го рядка j-го правила, a_i^{jp}, b_i^{jp}, c_i^{jp} – параметри настроювання функцій приналежності.

Третій шар утворюють рядки-кон'юнкції антецедентів правил нечіткої бази знань: кожний вузол цього шару реалізує оператор T-норми та видає ступінь відповідності вхідного вектора до співставленого вузлу правила.

Четвертий шар поєднує правила в класи d_j, j = l, 2, …, m, та обчислює ступені приналежності вхідного вектора до відповідних термів вихідної змінної; кожний вузол цього шару реалізує оператор T-конорми.

Функція приналежності вхідного вектора до j-го терму d_j вихідної змінної у визначається як: $\mu_{d_j}(y) = \max\limits_{p=1,k_j} \{ w_{jp} \min\limits_{i=1,N} [\mu_{jp}(x_i)] \}$.

П'ятий шар містить один нейрон, що поєднує приналежності до нечітких термів вихідної змінної та виконує операцію дефазифікації:

– для дискретного виходу: $y = \dfrac{\sum\limits_{j=1}^{m} d_j \mu_{d_j}(y)}{\sum\limits_{j=1}^{m} \mu_{d_j}(y)}$;

– для неперервного виходу: $y = \dfrac{y\mu_{d_1}(y) + y_1\mu_{d_2}(y) + \ldots + y_{m-1}\mu_{d_m}(y)}{\sum\limits_{j=1}^{m} \mu_{d_j}(y)}$.

Число вузлів у нейро-нечіткій мережі визначається так: шар 1 – за кількістю елементів вхідного вектора; шар 2 – за кількістю нечітких термів у базі знань (n); шар 3 – за кількістю рядків-кон'юнкцій у базі знань; шар 4 – за кількістю класів, на які розбивається діапазон вихідної змінної (m).

Дугам графа присвоюються такі ваги: одиниця (дуги між першим і другим шарами); функції приналежності входу до нечіткого терму (дуги між другим і третім шарами); ваги правил (дуги між третім і четвертим шарами); одиниця (дуги між четвертим і п'ятим шарами). Пороги нейронів вважають відсутніми або такими, що дорівнюють нулю.

Навчання нейро-нечіткої мережі Мамдані здійснюють *методом зворотного поширення помилки* з метою мінімізації *критерію помилки*

$$E = \frac{1}{V} \sum_{s=1}^{S} (y^{s*} - y^s)^2,$$

що застосовується в теорії нейронних мереж, де y^{s*} та y^s – бажане та розрахункове значення виходу мережі для *s*-го екземпляра, *V* – певна константа, як правило *V* = 2 або *S*.

Доти, поки помилка *E* не досягне максимального прийнятного рівня, послідовно для кожного *s*-го екземпляра x^s, *s* = 1, 2, …, *S*, виконують два етапи. На першому етапі (прямий хід) для *s*-го екземпляра обчислюють розрахункове значення виходу мережі y^s. На другому етапі (зворотний хід) обчислюють значення миттєвої помилки E_t та перераховують ваги зв'язків:

$$E_t = \frac{1}{2}(y^{s*}(t) - y^s(t))^2, \quad w_{jp}(t+1) = w_{jp}(t) - \alpha \frac{\partial E_t}{\partial w_{jp}(t)},$$

$$c_i^{jp}(t+1) = c_i^{jp}(t) - \alpha \frac{\partial E_t}{\partial c_i^{jp}(t)}, \; b_i^{jp}(t+1) = b_i^{jp}(t) - \alpha \frac{\partial E_t}{\partial b_i^{jp}(t)}, \quad j = \overline{1,m}, \quad i = \overline{1,N}, \quad p = k_j,$$

де $y^{s*}(t)$ та $y^s(t)$ – бажане та розрахункове значення виходу мережі для *s*-го екземпляра на *t*-му кроці навчання, $w_{jp}(t)$, $c_i^{jp}(t)$, $b_i^{jp}(t)$ – ваги правил і параметри функцій приналежності на *t*-му кроці навчання; α – величина кроку навчання.

Часткові похідні, що використовуються у правилах корегування параметрів мережі, характеризують чуттєвість помилки E_t до зміни параметрів та обчислюються таким чином:

$$\frac{\partial E_t}{\partial w_{jp}} = \varepsilon_1 \varepsilon_2 \varepsilon_3 \frac{\partial \mu_{d_j}}{\partial w_{jp}}, \; \frac{\partial E_t}{\partial c_i^{jp}} = \varepsilon_1 \varepsilon_2 \varepsilon_3 \varepsilon_4 \frac{\partial \mu_{jp}(x_i)}{\partial c_i^{jp}}, \; \frac{\partial E_t}{\partial b_i^{jp}} = \varepsilon_1 \varepsilon_2 \varepsilon_3 \varepsilon_4 \frac{\partial \mu_{jp}(x_i)}{\partial b_i^{jp}},$$

де $\varepsilon_1 = \frac{\partial E_t}{\partial y} = y_t^{s*} - y_t^s$, $\varepsilon_2 = \frac{\partial y}{\partial \mu_{d_j}} = \frac{d_j \sum\limits_{q=1}^{m} \mu_{d_q} - \sum\limits_{q=1}^{m} d_q \mu_{d_q}}{\left(\sum\limits_{q=1}^{m} \mu_{d_q} \right)^2}$, $\varepsilon_3 = \frac{\partial \mu_{d_j}}{\partial \left(\prod\limits_{i=1}^{N} \mu_{jp}(x_i) \right)} = w_{jp}$,

$$\varepsilon_4 = \frac{\partial \left(\prod\limits_{i=1}^{N} \mu_{jp}(x_i) \right)}{\partial \mu_{jp}(x_i)} = \frac{1}{\mu_{jp}(x_i)} \prod_{i=1}^{N} \mu_{jp}(x_i), \; \frac{\partial \mu_{d_j}}{\partial w_{jp}} = \prod_{i=1}^{N} \mu_{jp}(x_i),$$

$$\frac{\partial \mu_{jp}(x_i)}{\partial c_i^{jp}} = \frac{2 c_i^{jp} \left(x_i - b_i^{jp} \right)^2}{\left(\left(c_i^{jp} \right)^2 + \left(x_i - b_i^{jp} \right)^2 \right)^2}, \; \frac{\partial \mu_{jp}(x_i)}{\partial b_i^{jp}} = \frac{2 \left(c_i^{jp} \right)^2 \left(x_i - b_i^{jp} \right)}{\left(\left(c_i^{jp} \right)^2 + \left(x_i - b_i^{jp} \right)^2 \right)^2}.$$

Нейро-нечітка мережа FALCON (Fuzzy Adaptive Learning Control Network – нечітка мережа керування з адаптивним навчанням) має п'ятишарову архітектуру, як показано на рис. 4.5, і реалізує систему нечіткого виведення Мамдані. Для кожної вихідної змінної в мережі є два лінгвісти-

чних вузли. Один вузол – для навчаючих даних (бажаний вихід), а інший – для фактичного виходу FALCON. Перший схований шар виконує фазифікацію вхідних змінних. Кожен вузол може бути єдиним вузлом, що реалізує просту функцію приналежності, або складатися з декількох шарів вузлів, що обчислюють складну функцію приналежності. Другий схований шар визначає антецеденти правил, що супроводжуються консеквентами правил, які реалізує третій схований шар.

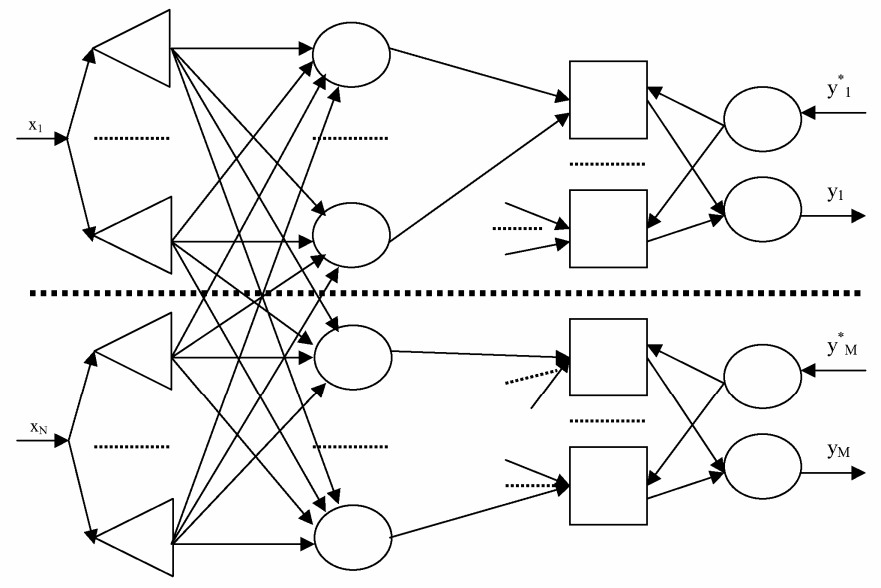


Рисунок 4.5 – Мережа FALCON

FALCON використовує гібридний метод навчання на основі неконтрольованого навчання і градієнтного спуску для оптимального настроювання параметрів з метою одержання бажаних виходів. Гібридне навчання здійснюється на двох стадіях.

У початковій стадії центри і ширина функцій приналежності визначаються методами навчання, що самоорганізуються, аналогічними статистичним методам кластеризації. Як тільки початкові параметри визначені формулюються антецеденти правил. Метод конкурентного навчання використовується для визначення зв'язків консеквентів правил для кожного вузла правила. Після того, як база нечітких правил отримана, уся структура мережі є відомою.

Потім мережа входить у другу стадію навчання для оптимального настроювання параметрів (вхідних і вихідних) функцій приналежності. Метод зворотного поширення помилки використовується для контрольованого навчання.

Таким чином, метод FALCON забезпечує основу для структурної і параметричної адаптації при проектуванні нейро-нечітких систем.

Нейро-нечітка мережа Такагі-Сугено-Канга (Takagi-Sugeno-Kang Neuro-fuzzy network, TSK) виконує нечітке виведення із використанням N змінних x_j та m правил:

$$\Pi_k\text{: Якщо } x_1 \in A^{(k)}{}_1 \text{ та } x_2 \in A^{(k)}{}_2 \text{ та } \ldots \text{ та } x \in A^{(k)}{}_N, \text{ тоді } y_k = w_0^k + \sum_{j=1}^{N} w_j^k x_j,$$

де $A^{(k)}{}_j$ – нечіткий терм, до якого має належати j-та вхідна змінна, щоб активізувати k-те правило, w^k_j – ваговий коефіцієнт, k = 1, 2, …, m.

Мережа Такагі-Сугено-Канга (рис. 4.6) складається з п’яти шарів.

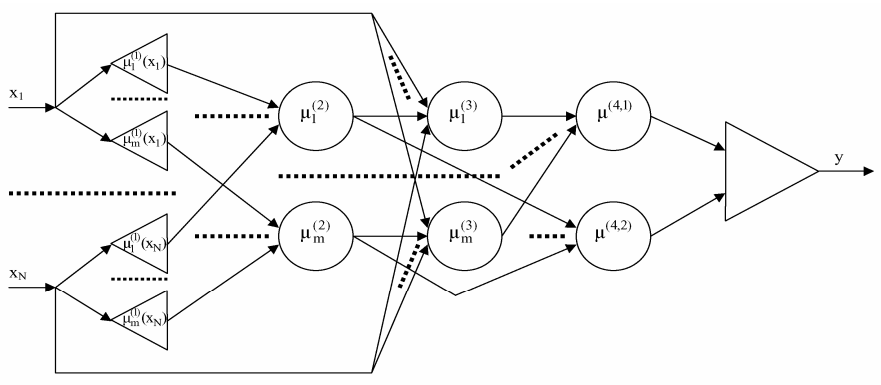


Рисунок 4.6 – Мережа Такагі-Сугено-Канга

Перший шар виконує окрему фазифікацію кожної вхідної змінної x_j, j = 1, 2, ..., N, визначаючи для кожного k-го правила значення функції приналежності

$$\mu_k^{(1)}(x_j) = \frac{1}{1 + \left(\dfrac{x_j - c_{jk}}{\sigma_{jk}} \right)^{2b_{jk}}},$$

параметри якої $c_{jk}, \sigma_{jk}, b_{jk}$ підлягають адаптації в процесі навчання.

Другий шар виконує агрегування функцій приналежності вхідних змінних до термів антецедентів нечітких правил, визначаючи приналежність вхідного вектора до *k*-го правила (рівень активації правила):

$$\mu_k^{(2)} = \prod_{j=1}^{N} \mu_k^{(1)}(x_j, c_{jk}, \sigma_{jk}, b_{jk}),\ k = 1, 2, \ldots, m.$$

Третій шар генерує значення функцій консеквентів нечітких правил з урахуванням рівнів активації правил:

$$\mu_k^{(3)} = \mu_k^{(2)} \left(w_0^k + \sum_{j=1}^{N} w_j^k x_j \right).$$

Це параметричний шар, у якому адаптації підлягають лінійні ваги w^k_j, $k = 1, 2, ..., m$, $j = 1, 2, ..., N$.

Четвертий шар агрегує *m* правил виведення (перший нейрон) і генерує нормалізуючий сигнал (другий нейрон):

$$\mu^{(4,1)} = \sum_{k=1}^{m} w_k^{(4)} \mu_k^{(3)},\ \mu^{(4,2)} = \sum_{k=1}^{m} \mu_k^{(3)}.$$

П’ятий (вихідний) шар містить один нейрон і здійснює нормалізацію, формуючи вихідний сигнал *y*: $y(x_1, x_2, ..., x_N) = \mu^{(4,1)} \Big/ \mu^{(4,2)}$.

Мережа Такагі-Сугено-Канга містить тільки два параметричних шари (перший і третій), параметри яких уточнюються в процесі навчання. Параметри першого шару будемо називати нелінійними параметрами, а параметри третього шару – лінійними вагами.

Якщо прийняти, що в конкретний момент часу параметри умов зафіксовані, то функція $y(x_1, x_2, \ldots, x_N)$ є лінійною щодо змінних $x_1, x_2, \ldots, x_N$.

При наявності *N* вхідних змінних кожне правило формує *N*+1 змінних w^k_j лінійної залежності $y_k(x_1, x_2, \ldots, x_N)$. При m правилах виведення це дає $m(N+1)$ лінійних параметрів мережі. У свою чергу, кожна функція приналежності використовує три параметри $c_{jk}, \sigma_{jk}, b_{jk}$, що підлягають адаптації. Якщо прийняти, що кожна змінна x_j, характеризується власною функцією приналежності, то при *m* правилах виведення ми одержимо 3*m* нелінійних параметрів. У сумі це дає $m(4N+1)$ лінійних і нелінійних параметрів, значення яких повинні підбиратися в процесі навчання мережі.

На практиці для зменшення кількості параметрів, що адаптуються, оперують меншою кількістю незалежних функцій приналежності для окремих змінних, керуючись правилами, у яких комбінуються функції приналежності різних змінних. Якщо прийняти, що кожна змінна x_j має k_j різних функцій приналежності, то максимальна кількість правил, яку можна створити при їхньому комбінуванні, складе: $m = k_j$. У такий спосіб сумарна кількість нелінійних параметрів мережі при *m* правилах висновку зменшується з 3*m* у зага-

льному випадку до $3Nm^{1/N}$. Кількість лінійних параметрів при подібній модифікації залишається без змін, тобто $m(N+1)$.

Завдання мережі полягає в такому відображенні пар даних $<x, y>$, щоб для вхідного вектора x^s розрахункове значення вихідної ознаки $y^{s*} = y(x^s)$ не сильно відрізнялося б від співставленого x^s фактичного значення цільової ознаки y^s.

Навчання мережі засновано на мінімізації цільової функції

$$E = \frac{1}{2}\sum_{s=1}^{S}\left(y^{s*} - y^s\right)^2,$$

де S – кількість навчаючих пар $<x^s, y^s>$, та є контрольованим.

Гібридний метод навчання застосовується для мережі Такагі-Сугено-Канга й інших нейро-нечітких мереж, подібних їй. У гібридному методі підлягаючі адаптації параметри розподіляються на дві групи: одна група складається з лінійних параметрів w^k_j третього шару, а інша група – з параметрів нелінійних функцій приналежності першого шару. Уточнення параметрів проводиться в два етапи.

На першому етапі при фіксації визначених значень параметрів функцій приналежності (у першому циклі – це значення, отримані в результаті ініціалізації) шляхом розв'язку системи лінійних рівнянь розраховуються лінійні параметри w^k_j. При відомих значеннях функції приналежності залежність $y(x_1, x_2, \ldots, x_N)$ можна подати в лінійній формі:

$$y(x_1,.x_2,...,x_N) = \sum_{k=1}^{m} w'_k\left(w^k_0 + \sum_{j=1}^{N} w^k_j x_j\right), \quad w'_k = \left(\sum_{p=1}^{m}\left(\prod_{j=1}^{N}\mu^{(1)}_p(x_j)\right)\right)^{-1}\prod_{j=1}^{N}\mu^{(1)}_k(x_j).$$

При S екземплярах навчаючої вибірки $<x^s, y^s>$, $s = 1, 2, ..., S$, одержимо систему з лінійних рівнянь у матричній формі: $Aw = y$, де $w = (w^1, w^2, \ldots, w^m)^T$, $w^k = (w^k_0, w^k_1, \ldots, w^k_N)^T$, $y = (y_1, y_2, \ldots, y_S)^T$, $A = (a_1, a_2, \ldots, a_S)^T$, $a_s = (a_{s1}, a_{s2}, \ldots, a_{sm})^T$, $a_{sk} = (\mu^{(2)}_k(x^s), \mu^{(2)}_k(x^s)x^s_1, \mu^{(2)}_k(x^s)x^s_2, ..., \mu^{(2)}_k(x^s)x^s_N)^T$. Розмірність матриці A дорівнює $S(N+1)m$, при цьому звичайно кількість рядків S значно більше кількості стовпців $(N+1)m$. Розв'язок цієї системи: $w = A^{-1}y$.

На другому етапі після фіксації значень лінійних параметрів w^k_j розраховуються вихідні сигнали мережі $y = \{y^{s*}\}$ $(s = 1, 2, \ldots, S)$: $y = Aw$, а слідом за ними – вектор помилки $e = \{e^s\}$, $e^s = y^{s*} - y^s$. Сигнали помилок направляються через підключену мережу за напрямком до входу мережі (зворотне поширення) аж до першого шару, де можуть бути розраховані компоненти градієнта цільової функції щодо конкретних параметрів $c_{jk}, \sigma_{jk}, b_{jk}$.

Після формування вектора градієнта параметри уточнюються з використанням одного з градієнтних методів навчання. Якщо застосовується найпростіший метод найшвидшого спуску, то відповідні формули адаптації приймають форму:

$$c_{jk}(t+1)=c_{jk}(t)-\alpha_c\frac{\partial E(t)}{\partial c_{jk}},\ \sigma_{jk}(t+1)=\sigma_{jk}(t)-\alpha_\sigma\frac{\partial E(t)}{\partial \sigma_{jk}},\ b_{jk}(t+1)=b_{jk}(t)-\alpha_b\frac{\partial E(t)}{\partial b_{jk}},$$

де t – номер ітерації, α_c, α_σ, α_b – коригувальні прирости (кроки навчання).

Після уточнення нелінійних параметрів знову запускається процес адаптації лінійних параметрів (перший етап) і нелінійних параметрів мережі (другий етап). Цей цикл повторюється аж до стабілізації всіх параметрів процесу навчання.

Остаточний вид формул настроювання параметрів залежить як від використовуваного визначення функції помилки на виході мережі, так і від виду функції приналежності. Так при використанні функцій Гауса як функцій приналежності та середньоквадратичної помилки, часткові похідні цільової функції приймають вигляд:

$$\frac{\partial E}{\partial c_{jk}}=\left(y^{s*}-y^s\right)\sum_{p=1}^{m}\left(w_0^p+\sum_{i=1}^{N}w_i^p x_i\right)\frac{\partial w_p^{'}}{\partial c_{jk}},\ \frac{\partial E}{\partial \sigma_{jk}}=\left(y^{s*}-y^s\right)\sum_{p=1}^{m}\left(w_0^p+\sum_{i=1}^{N}w_i^p x_i\right)\frac{\partial w_p^{'}}{\partial \sigma_{jk}},$$

$$\frac{\partial E}{\partial b_{jk}}=\left(y^{s*}-y^s\right)\sum_{p=1}^{m}\left(w_0^p+\sum_{i=1}^{N}w_i^p x_i\right)\frac{\partial w_p^{'}}{\partial b_{jk}},$$

$$\frac{\partial w_p^{'}}{\partial c_{jk}}=\beta_{pk}\left(\prod_{i=1,i\neq j}^{N}\mu_k^{(1)}(x_i)\right)\left(\frac{2b_{jk}}{\sigma_{jk}}\right)\left(\frac{x_j-c_{jk}}{\sigma_{jk}}\right)^{2b_{jk}-1}\left(1+\left(\frac{x_j-c_{jk}}{\sigma_{jk}}\right)^{-2b_{jk}}\right)^{-2},$$

$$\frac{\partial w_p^{'}}{\partial \sigma_{jk}}=\beta_{pk}\left(\prod_{i=1,i\neq j}^{N}\mu_k^{(1)}(x_i)\right)\left(\frac{2b_{jk}}{\sigma_{jk}}\right)\left(\frac{x_j-c_{jk}}{\sigma_{jk}}\right)^{2b_{jk}}\left(1+\left(\frac{x_j-c_{jk}}{\sigma_{jk}}\right)^{-2b_{jk}}\right)^{-2},$$

$$\frac{\partial w_p^{'}}{\partial c_{jk}}=\beta_{pk}\left(\prod_{i=1,i\neq j}^{N}\mu_k^{(1)}(x_i)\right)\left(-2\left(\frac{x_j-c_{jk}}{\sigma_{jk}}\right)^{2b_{jk}}\ln\left(\frac{x_j-c_{jk}}{\sigma_{jk}}\right)\right)\left(1+\left(\frac{x_j-c_{jk}}{\sigma_{jk}}\right)^{-2b_{jk}}\right)^{-2},$$

$$\beta_{pk}=\left(\delta_{pk}\sum_{q=1}^{m}\left(\prod_{j=1}^{N}\mu_q^{(1)}(x_j)\right)-\prod_{j=1}^{N}\mu_k^{(1)}(x_j)\right)\left(\sum_{q=1}^{m}\left(\prod_{j=1}^{N}\mu_q^{(1)}(x_j)\right)\right)^{-2},\ \delta_{pk}=\begin{cases}1, p=k,\\ 0, p\neq k.\end{cases}$$

Незважаючи на складну структуру наведених формул, що виражають компоненти вектора градієнта, вони дозволяють аналітично визначити величини, необхідні для уточнення параметрів нечіткої мережі.

При практичній реалізації гібридного методу навчання нечітких мереж домінуючим фактором їхньої адаптації вважається перший етап, на якому ваги w_j^k підбираються з використанням псевдоінверсії за один крок. Для зрівноважування його впливу другий етап (підбір нелінійних параметрів градієнтним методом) багаторазово повторюється в кожному циклі.

Гібридний метод є одним з найбільш ефективних способів навчання нейро-нечітких мереж. Його головна особливість полягає в розподілі процесу навчання на два відособлених у часі етапи. На кожному етапі уточнюється тільки частина параметрів мережі. Якщо взяти до уваги, що обчислювальна складність кожного методу оптимізації пропорційна (нелінійно) кількості параметрів, то зменшення розмірності задачі оптимізації істотно скорочує кількість математичних операцій і збільшує швидкість виконання методу. Завдяки цьому гібридний метод є значно більш ефективним, ніж звичайний градієнтний метод фронтального типу, відповідно до якого уточнення всіх параметрів мережі робиться паралельно й одночасно.

Нейро-нечітка мережа Ванга-Менделя (Wang-Mendel neuro fuzzy network) запропонована Вангом (L. Wang) та Менделем (G. Mendel) і являє собою чотиришарову мережу прямого поширення, що реалізує нечітке виведення Такагі-Сугено (рис. 4.7).

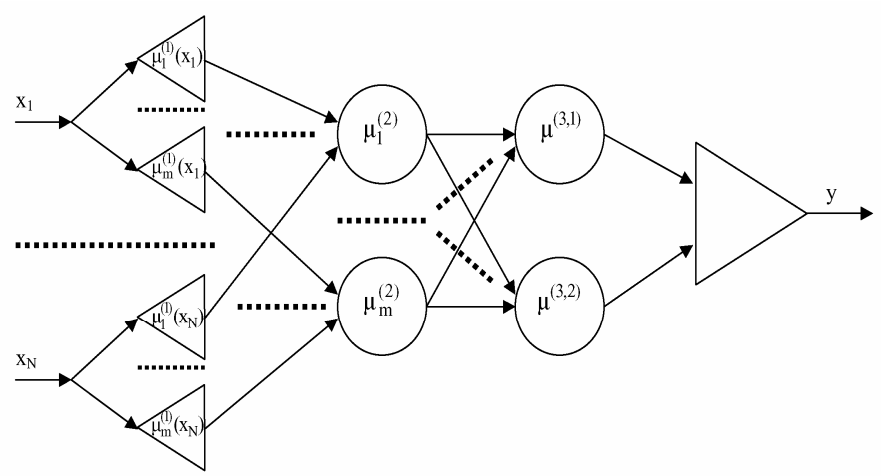


Рисунок 4.7 – Мережа Ванга-Менделя

Мережа Ванга-Менделя є окремим випадком мережі Такагі-Сугено-Канга. Частини, що визначають умови правил (перший і другий шари), у цих мереж є ідентичними. Розходження спостерігаються в поданні компонентів консеквентів: на відміну від мережі Такагі-Сугено-Канга, де консеквент подається поліномом першого порядку, у мережі Ванга-Менделя консеквент подається константою $w_k^{(3)}$, яку можна розглядати як поліном нульового порядку, що визначає центр функції приналежності наслідку.

Перший шар мережі Ванга-Менделя визначає значення функцій приналежності вхідних змінних $\mu_k^{(1)}(x_j)$, де j – номер вхідної змінної, $j = 1, 2,\ldots,N$; k – номер правила, $k = 1, 2, \ldots, m$.

Другий шар агрегує значення активації умов правил:

$$\mu_k^{(2)} = \prod_{j=1}^{N} \mu_k^{(1)}(x_j, c_{jk}, \sigma_{jk}, b_{jk}) \ , \ k = 1, 2, \ldots, m.$$

Третій (лінійний) шар агрегує m правил висновку (перший нейрон) і генерує нормалізуючий сигнал (другий нейрон):

$$\mu^{(3,1)} = \sum_{k=1}^{m} w_k^{(3)} \mu_k^{(2)}, \ \mu^{(3,2)} = \sum_{k=1}^{m} \mu_k^{(2)}.$$

Четвертий (вихідний) шар містить один нейрон і здійснює нормалізацію, формуючи вихідний сигнал y: $y(x_1, x_2, ..., x_N) = \dfrac{\mu^{(3,1)}}{\mu^{(3,2)}}$.

У мережі Ванга-Менделя параметричними є тільки перший і третій шари. У першому шарі налагоджуються параметри $c_{jk}, \sigma_{jk}, b_{jk}$ функцій приналежності $\mu_k^{(1)}(x_j, c_{jk}, \sigma_{jk}, b_{jk})$ а в третьому шарі – ваги $w_k^{(3)}$, що інтерпретуються як центри функції приналежності наслідку k-го правила.

Для мережі Ванга-Менделя можна в аналітичному виді виразити градієнт функції помилки від параметрів мережі, що дозволяє для її навчання використовувати метод зворотного поширення або гібридний метод, у якому частина параметрів налагоджується градієнтним методом, а частина – за допомогою обчислення псевдозворотної матриці. При використанні гібридного методу мережа Ванга-Менделя трактується як мережа Такагі-Сугено-Канга, у якій усі коефіцієнти при вхідних змінних у функціях консеквентів тотожно дорівнюють нулю, за винятком підлягаючих уточненню $w_k^{(3)}$. Також для мережі Ванга-Менделя можна використовувати методи адаптації узагальнено-регресійних нейронних мереж.

Нейро-нечітка мережа ANFIS (Adaptive Neuro Fuzzy Inference System – адаптивна система нейро-нечіткого виведення) запропонована Янгом (R. Jang) і реалізує систему нечіткого виведення Такагі-Сугено у вигляді п’ятишарової нейронної мережі прямого поширення сигналу (рис. 4.8).

Мережа ANFIS є ізоморфною базі правил Такагі-Сугено.

Перший шар містить нейрони, які відповідають нечітким термам вхідних змінних із функціями приналежності $\mu_{i,j}^{(1)}(x_i)$, де x_i – i-ий вхід, j – номер нечіткої множини, визначеної для i-го входу. Як функція приналежності за-

звичай обирається колоколообразна функція або функція Гауса. Входи мережі з'єднані тільки зі своїми термами. Кількість вузлів першого шару дорівнює сумі потужностей терм-множин вхідних змінних.

Другий шар містить m нейронів, на входи яких поступають значення з виходів вузлів першого шару, що формують антецеденти правил. Кожний k-ий нейрон другого шару знаходить ступінь виконання відповідного правила:

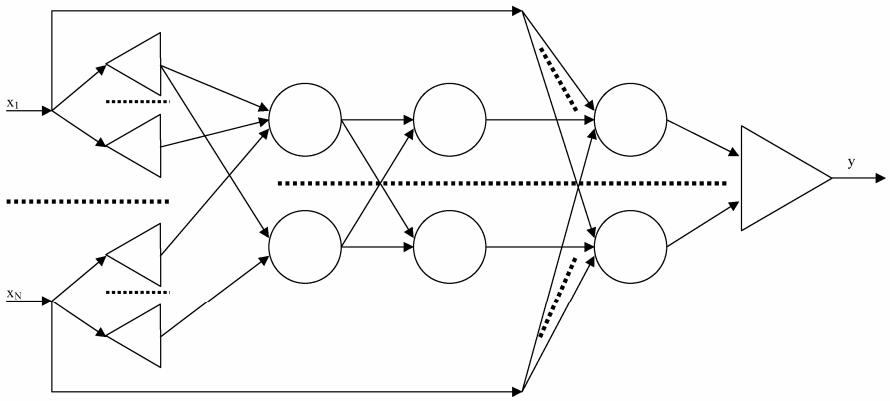


Рисунок 4.8 – Мережа ANFIS

$$\mu_k^{(2)} = \min_{\substack{i=1,2,...,N \\ j=1,2,...,k_i}} (w_{i,j}^{(2,k)} \mu_{i,j}^{(1)}) \text{ або } \mu_k^{(2)} = \prod_{i=1}^{N} \prod_{j=1}^{k_i} \mu_{i,j}^{(1)}, w_{i,j}^{(2,k)} = 1,$$

де $w_{i,j}^{(2,k)} = 1$, якщо j-ий терм i-ої ознаки входить в умову k-го правила, $w_{i,j}^{(2,k)} = 0$ – у протилежному випадку.

Третій шар містить m нейронів, які знаходять нормалізовані ступені виконання правил:

$$\mu_j^{(3)} = \frac{\mu_j^{(2)}}{\sum_{i=1}^{m} \mu_i^{(2)}}, j = 1, 2, ..., m.$$

Четвертий шар містить m нейронів, які обчислюють консеквенти (лінійні комбінації вхідних сигналів з урахуванням ступенів виконання) правил:

$$y_j = \mu_j^{(3)} \sum_{i=1`}^{N} w_i^{(4,j)} x_i, j = 1, 2, ..., m.$$

Кожний вузол четвертого шару з'єднаний з одним вузлом третього шару а також із усіма входами мережі.

П'ятий шар містить один нейрон, який обчислює загальний вихід мережі, складаючи консеквенти правил:

$$y = \sum_{j=1}^{m} y_j \ .$$

Для налагодження параметрів ANFIS застосовують комбінацію градієнтного спуску у вигляді методу зворотного поширення помилки і методу найменших квадратів. Метод зворотного поширення помилки налагоджує параметри антецедентів правил (функцій приналежності). Методом найменших квадратів оцінюються коефіцієнти висновків правил, тому що вони лінійно пов'язані з виходом мережі.

Кожна ітерація процедури настроювання виконується в два етапи. На першому етапі на входи подається навчаюча вибірка, і за відхилом між бажаною і дійсною поведінкою мережі ітераційним методом найменших квадратів знаходяться оптимальні параметри вузлів четвертого шару. На другому етапі залишковий відхіл передається з виходу мережі на входи, і методом зворотного поширення помилки модифікуються параметри вузлів першого шару. При цьому знайдені на першому етапі коефіцієнти висновків правил не змінюються. Ітераційна процедура настроювання продовжується поки відхіл перевищує заздалегідь установлене значення.

Нейро-нечітка мережа NNFLC (Neural Network Fuzzy Logic Controller – нечіткий контролер на основі нейронної мережі) – багатошарова нейронна мережа прямого поширення сигналу, що являє собою окремий випадок системи нечіткого виведення Такагі-Сугено (рис. 4.9). Модель NNFLC запропонована Лін (C. T. Lin) та Лі (C. S. Lee).

На входи мережі надходить вхідний N-мірний вектор x^s.

Перший шар містить функції приналежності, реалізовані як радіальні базисні нейрони: $y^{(1,i)}(x_i) = \exp\left(\frac{-(x_i - c_i)^2}{2\sigma_i^2}\right)$.

Другий шар моделює ТА-умови правил: $y^{(2,i)} = \min\limits_{j=1,2,...,N_1} y^{(1,j)}$.

Третій шар являє собою АБО-комбінацію правил з однаковими термами в консеквентах і виконує різні функції в робочому режимі й у режимі навчання. У режимі навчання шар налагоджує параметри функцій приналежності вихідних змінних. У робочому режимі формує значення виходу:

$$y^{(3,i)} = \max_{j=1,2,...,N_2} y^{(2,j)}.$$

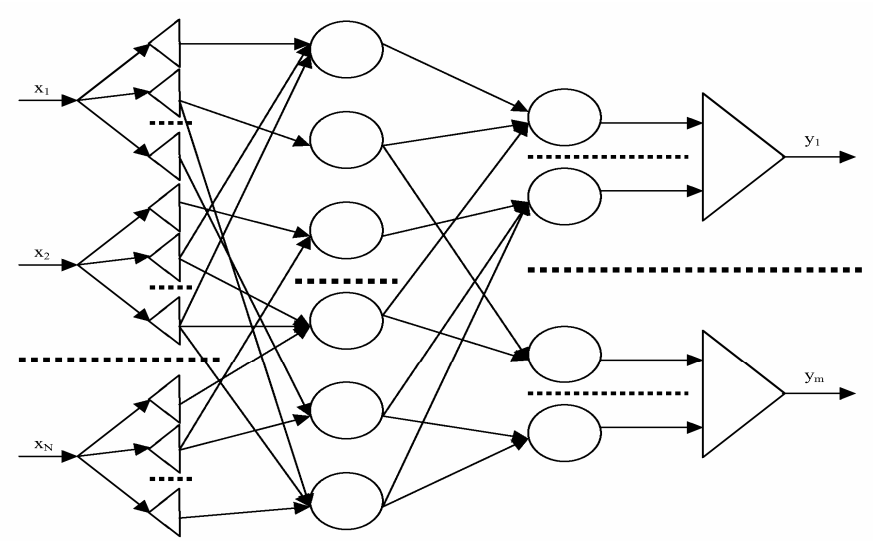


Рисунок 4.9 – Структура мережі NNFLC

Нейрони четвертого шару виконують дефазифікацию:

$$y^{(4,i)} = \left(\sum_{j=1}^{N_3} w_j^{(4,i)} y^{(3,j)} \right) \Bigg/ \sum_{j=1}^{N_3} w_j^{(4,i)}.$$

Структура мережі NNFLC ініціалізується за принципом формування повної матриці правил. Якщо x_i – вхідні змінні, $\tau(x_i)$ – кількість нечітких розбиттів x_i, тоді вихідна кількість правил:

$$T = \prod_{i=1}^{N} \tau(x_i).$$

Загальна схема навчання мережі NNFLC містить такі етапи: 1) формування навчаючих даних; 2) кластеризация, що самоорганізується (настроювання функцій приналежності); 3) конкурентне навчання (метод переможця); 4) видалення правил; 5) комбінування правил; 6) остаточне настроювання параметрів (тюнінг) функцій приналежності за допомогою методу зворотного поширення помилки.

Настроювання параметрів функцій приналежності містить у собі визначення центрів c_i та ширини σ_i для функцій приналежності вхідних змінних.

Метод переможця виявляє c_i: $\Delta c_w(t) = \alpha(t)(x - c_w(t))$, де $\alpha(t)$ – монотонно спадаючий рівень навчання.

Настроювання ширини σ_i здійснюється евристично, наприклад за принципом «першого найближчого сусіда»: $\sigma_i = (\sigma_i - \sigma_w)/\lambda$, де λ – параметр перекриття.

Метод переможця шукає матрицю ваг w_{ij}, що оцінює якість зв'язків лівої і правої частин правил: $\Delta w = y^{(3,i)}(y^{(2,j)} - w_j^{(3,i)})$.

Комбінування правил часто доцільно виконувати за участю експерта.

Остаточне настроювання функцій приналежності виконується за допомогою методу зворотного поширення помилки для функції помилки $e_i = 0{,}5(y^{(4,i)} - y^*_i)^2$.

Нейро-нечітка мережа SONFIN (Self Constructing Neural Fuzzy Inference Network – самоспоруджувана мережа нейро-нечіткого виведення) реалізує систему нечіткого виведення Такагі-Сугено (рис. 4.10). Нечіткі правила створюються й адаптуються в процесі навчання за допомогою одночасної ідентифікації структури і параметрів.

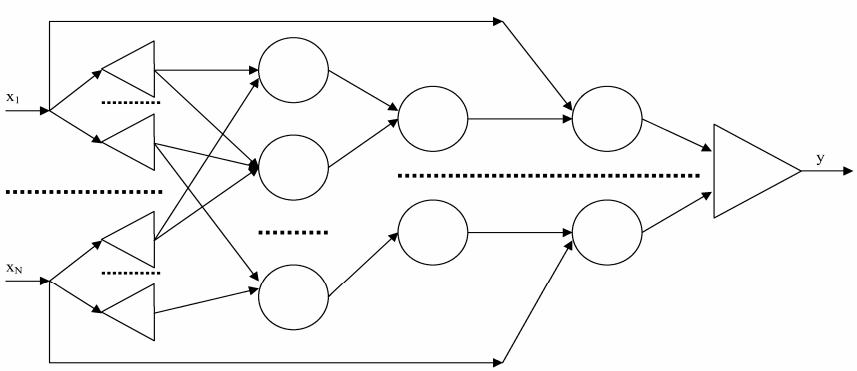


Рисунок 4.10 – Мережа SONFIN

Навчання здійснюється паралельно на двох етапах для побудови SONFIN. Структурне навчання включає ідентифікацію структури антецедента і консеквента нечіткого правила «якщо-то». При структурній ідентифікації антецедентів вхідний простір розподіляється на основі кластеризації. При структурній ідентифікації консеквентів на основі кластеризації кожному правилу спочатку зіставляється унікальне значення.

Згодом деякі додаткові значимі терми (вхідні змінні), відібрані на основі кореляційної міри для кожного правила, додаються до консеквенту (формуючи лінійне рівняння вхідних змінних).

Параметричне навчання засноване на методах навчання з учителем. Для параметричної ідентифікації оптимально налагоджуються параметри консеквентів (параметри лінійних рівнянь у консеквентах) методом найменших квадратів, а параметри антецедентів налагоджуються методом зворотного поширення помилки. Для збільшення здатності SONFIN подавати знання в мережу може бути включене лінійне перетворення для кожної вхідної змінної таким чином, щоб була потрібна менша кількість правил або більш висока точність могла б бути досягнута. Відповідні лінійні перетворення також динамічно навчаються на етапі ідентифікації параметрів SONFIN.

SONFIN може використовуватися для нормальної роботи в будь-який час протягом процесу навчання без ітеративного навчання за образами «вхід – вихід», коли потрібна оперативна обробка.

У SONFIN база правил створюється динамічно в процесі навчання, виконуючи такі етапи:

– Поділ простору вхід-вихід. Спосіб поділу вхідного простору визначає кількість правил, що витягаються з навчаючих даних, а також кількість нечітких множин на універсумі для кожної вхідної змінної. Для кожного екземпляра, що надходить на вхід, ступінь виконання правила може інтерпретуватися як ступінь приналежності екземпляра, що надходить, до відповідного кластеру. Центр і ширина відповідних функцій приналежності (нових сформованих нечітких правил) установлюються відповідно до евристики «перший найближчий сусід». Для кожного згенерованого правила наступним кроком є декомпозиція багатомірної функції приналежності на відповідні функції приналежності для кожної вхідної змінної. Для поділу вихідного простору застосовується подібний спосіб. Робота SONFIN може бути поліпшена за допомогою включення в структуру мережі правил, запропонованих експертами.

– Формування нечіткої бази правил. Генерація нового вхідного кластера відповідає генерації нового нечіткого правила, антецедент якого формується навчаючим методом, а консеквент формується, використовуючи метод, заснований на факті, що різні антецеденти можуть бути відображені до тієї самої нечіткої множини консеквента. Оскільки для дефазифікації використовується тільки центр кожної функції приналежності, консеквент кожного правила може бути розглянутий як унікальне значення (*сінглтон*). У порівнянні з узагальненими моделями, заснованими на нечітких правилах з унікальними значеннями виходу, у консеквенті SONFIN потрібною є менша кількість параметрів, особливо для складних систем з великою кількістю правил.

– Оптимальна ідентифікація структури консеквента. Модель Такагі-Сугено здатна моделювати складну систему за допомогою декількох правил. У SONFIN у консеквентах замість використання лінійної комбінації усіх вхідних змінних використовуються тільки найбільш істотні вхідні змінні як терми консеквентів. Найбільш істотні терми вибираються і додаються в мережу

щоразу, коли параметричне навчання не може поліпшити точність виходу мережі протягом процесу навчання. Схема ідентифікації структури консеквента в SONFIN являє собою різновид методу нарощування вузлів у штучних нейромережах. Коли вплив параметричного навчання скорочується (помилка виходу не зменшується), додаткові терми додаються в консеквент.

– Ідентифікація параметрів. Після того, як структура мережі налагоджена відповідно до поточного екземпляра навчаючої вибірки, мережа входить у стадію ідентифікації параметрів для того, щоб оптимально налагодити параметри мережі для того ж самого екземпляра. Параметричне навчання здійснюється для всієї мережі після структурного навчання, незалежно від того, чи були вузли (зв'язки) знову додані або існували спочатку. Для цього контрольованого навчання використовується метод зворотного поширення помилки.

SONFIN є обчислювально найвитратною серед усіх нейро-нечітких моделей. Вона здатна пристосовуватися до точності, що задається користувачем.

Нечіткий персептрон має структуру, аналогічну структурі звичайного персептрона – багатошарової нейронної мережі прямого поширення сигналу, але ваги замінюються нечіткими множинами.

Тришаровий нечіткий персептрон – це нейронна мережа прямого поширення (U, W, NET, O, A, ex), з такими властивостями:

– $U = \bigcup_{i=1}^{3} U_i$ – непуста множина нейронів, де U_1 – шар входів, U_2 – шар правил, U_3 – шар виходів. Для всіх $i, j \in \{1, 2, 3\}$: $U_i \neq 0$, $U_i \cap U_j = 0$, $i \neq j$;

– структура мережі (зв'язки) описується W: $U \times U \to F(R)$ таким чином, що є тільки зв'язки $W(u, v)$, де $u \in U_i$, $v \in U_{i+1}$, $i \in \{1,2\}$, $F(R)$ – множина всіх нечітких підмножин R;

– A описує функції активації A_u для кожного нейрона $u \in U$, що визначає a_u: для нейронів шару входів та шару правил $u \in U_1 \cup U_2$: A_u: $R \to R$, $a_u = A_u(\text{net}_u) = \text{net}_u$; для нейронів шару виходів $u \in U_3$: A_u: $F(R) \to F(R)$, $a_u = A_u(\text{net}_u) = \text{net}_u$;

– O описує для кожного нейрона $u \in U$ вихідну функцію O_u, що визначає вихід нейрона o_u: для шару входів та шару правил $u \in U_1 \cup U_2$: O_u: $R \to R$, $o_u = O_u(a_u) = a_u$; для шару виходів $u \in U_3$: O_u: $F(R) \to R$, $o_u = O_u(\text{net}_u) = \text{DEFUZZ}_u(\text{net}_u)$, де DEFUZZ_u – функція дефазифікації;

– NET описує для кожного нейрона $u \in U$ функцію NET_u, що визначає значення net_u: для шару входів $u \in U_1$: NET_u: $R \to R$, $\text{net}_u = \text{ex}_u$; для шару правил $u \in U_2$: NET_u: $(R \times F(R))^{U1} \to [0, 1]$, $net_u(x) = \underset{u' \in U_1}{T} (W(u', u)(o_{u'}))$, де T – t-норма; для шару виходів $u \in U_3$: NET_u: $([0,1] \times F(R))^{U2} \to F(R)$, net_u: $R \to [0, 1]$, $net_u(x) = \underset{u' \in U_2}{\perp} (T(o_{u'}, W(u', u)(x)))$, де $\perp$ – t-конорма;

– ex: $U_1 \to R$ для кожного нейрона шару входів $u \in U_1$ описує зовнішній вхід $\text{ex}(u) = \text{ex}_u$. Для інших шарів ex не визначається.

Нейро-нечітка мережа NEFCON (Neuro-Fuzzy Controller – нейро-нечіткий контролер) побудована на базі моделі загального нечіткого персептрона з додаванням семантичних обмежень Ф (рис. 4.11).

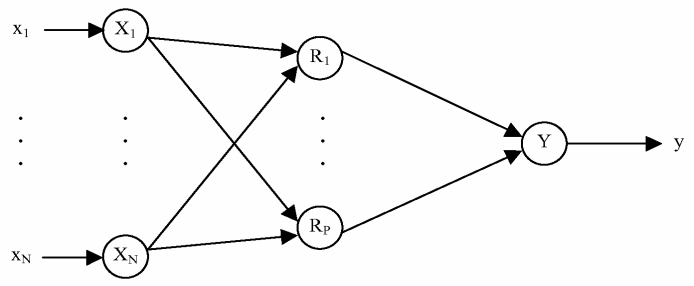


Рисунок 4.11 – Мережа NEFCON

Мережа NEFCON має такі властивості:

– $U = \bigcup_{i=1}^{3} U_i$, де $U_1 = \{X_1, X_2, \ldots, X_N\}$ – шар входів (N нейронів), значення вхідних змінних визначають стан системи, $U_2 = \{R_1, R_2, \ldots, R_P\}$ – шар правил, $U_3 = \{Y\}$ – шар виходів (один нейрон), вихідна змінна – це управляючий вплив на керовану систему;

– для всіх шарів функції активації нейронів $a_u = A_u(\text{net}_u) = \text{net}_u$;

– вихідна функція: для шару входів та шару правил $o_u = O_u(a_u) = a_u$; для шару виходів $o_Y = O_Y(\text{net}_Y) = \text{DEFUZZ}_Y(\text{net}_Y)$, де DEFUZZ_Y – функція дефазифікації.

– зв'язки між шарами: між першим та другим шарами $W(X_i, R) \in \{\mu_1^{(i)}, \ldots, \mu_{n_i}^{(i)}\}$, функції μ – трикутні, між шаром правил та шаром виходів $W(R, Y) = \{v_1, \ldots, v_{n_Y}\}$, функції v – трикутні.

– функція NET_u: для шару входів $\text{net}_X = \text{ex}_X$; для шару правил $\text{net}_R = \underset{X \in U_1}{T} (W(X,R)(o_X))$, де T – це t-норма; для шару виходів $\text{net}_Y(x) = \underset{R \in U_2}{\perp} (T(o_R, W(R,Y)(x)))$, де $\perp$ – це t-конорма.

База знань нечіткої системи неявно задається структурою мережі. Вхідні вузли виконують фазифікацію, логіка виведення подана функціями поширення, а вихідний вузол виконує дефазифікацію. На відміну від нейронних мереж, зв'язки в NEFCON є зваженими за допомогою нечітких множин замість реальних чисел. Правила з однаковими антецедентами використовують так звані поділяючі ваги, що гарантують цілісність бази правил.

Процес навчання NEFCON може бути поділений на дві головні стадії. Перша стадія призначена для навчання бази правил, а друга стадія оптимізує правила, змінюючи нечіткі множини правил.

Для навчання бази правил доступні два методи. Навчання з нарощуванням правил використовується, коли правильний вихід є невідомим і правила створені на основі оцінених значень виходу. У процесі навчання більша кількість правил додається відповідно до вимог. Для навчання зі скороченням правил спочатку створюються правила на основі нечітких розбиттів змінних процесу і непотрібні правила усуваються в ході навчання. Навчання зі скороченням правил є менш ефективним у порівнянні з нарощуванням правил. Однак воно може застосовуватися до невідомих процесів без ускладнень і не потребує знати оптимальне значення виходу.

Обидві стадії використовують нечітку помилку E, що описує якість поточного стану системи для навчання або оптимізації бази правил. Щоб одержати гарну базу правил необхідно, щоб простір станів процесу був добре охоплений протягом процесу навчання.

Через складність необхідних обчислень навчання зі скороченням правил має використовується тільки тоді, коли є тільки кілька вхідних змінних з не дуже великою кількістю нечітких множин. Для великих систем оптимальним буде навчання з нарощуванням правил. Апріорні знання завжди, коли можливо, повинні бути використані для зменшення складності навчання.

Метод генерації правил NEFCON.

Фаза 1. Повторити наступні кроки m_1 разів:

Крок 1. Для поточного вхідного вектора (x_1, ... , x_N) знайти такі нечіткі множини $\bar{\mu}^{(1)},...,\bar{\mu}^{(N)}$, що: $\forall i, j : \bar{\mu}^{(i)}(x_i) \geq \bar{\mu}_j^{(i)}(x_i))$.

Крок 2. Якщо в системі ще немає правила з антецедентом «Якщо $x_1 \in \bar{\mu}^{(1)}$ та ... та $x_N \in \bar{\mu}^{(N)}$», тоді знайти таку нечітку множину $\bar{v}$, для якої $\forall j : \bar{v}(o) \geq v_j(o)$, де значення o визначається за формулою:

$$o = \begin{cases} m + |E|(\eta_{\max} - m), \text{ якщо } E \geq 0, \\ m - |E|(m - \eta_{\min}), \text{ якщо } E < 0, \end{cases}$$

де $m = 0{,}5(\eta_{\max} + \eta_{\min})$ – середнє значення виходу керованої системи; $\eta_{\min}$, $\eta_{\max}$ – відповідно мінімальне та максимальне можливі значення виходу керованої системи; E – значення помилки.

Крок 3. Додати правило «Якщо $x_1 \in \bar{\mu}^{(1)}$ та ... та $x_N \in \bar{\mu}^{(N)}$, тоді $y \in \bar{\nu}$ «в базу знань.

Фаза 2. Повторити наступні кроки m_2 разів:

Крок 1. Пропустити поточний вхідний вектор через мережу та визначити для кожного правила R_p бажаний внесок в загальний вихід системи: $t_p = v_{j_p}^{-}(o_p) + \Delta b_{j_p}$.

Крок 2. Для кожного правила R_p визначити нову вихідну нечітку множину таку, що $\forall j : \bar{v}_p(t_p) \geq v_j(t_p))$, та змінити консеквент правила R_p відповідно.

Крок 3. Видалити з системи всі правила, які не використовуються при обробці більш, ніж визначений відсоток екземплярів.

Функції приналежності бази правил змінюються за допомогою методу зворотного поширення помилки. Метод може налагоджувати функції приналежності і може застосовуватися тільки, якщо вже є база нечітких правил. Ідея методу навчання аналогічна: збільшувати вплив правила, якщо його дія здійснюється в правильному напрямку (заохочування), і зменшувати вплив, якщо правило поводиться контр-продуктивно (покарання). Якщо які-небудь відомості про початкові функції приналежності є відсутніми, тоді повиний використовуватися однорідний поділ нечітких змінних.

Метод навчання нечітких множин NEFCON здатний навчати нечіткі множини також як нечіткі правила, реалізовані системою нечіткого виведення типу Мамдани. Цей метод може розглядатися як розширення методу GARIC, що також використовує навчання з підкріпленням, але має потребу в попередньо визначеній базі правил.

Для навчання мережі NEFCON використовується значення помилки $E = \eta_{\text{opt}} - \eta$, де η_{opt} – оптимальний (бажаний) вихід керованої системи, η – реальний вихід керованої системи. Враховується також напрямок помилки. Напрямок помилки визначається за формулою:

$$\tau = \begin{cases} 1, \text{ якщо } |E'| \geq |E| \text{ й } E'E \geq 0; \\ 0, \text{ якщо } |E'| < |E| \text{ й } E'E \geq 0; \\ -1, \text{ якщо } E'E < 0, \end{cases}$$

де E' – помилка в момент часу t+1; E – помилка в момент часу t.

Для нечітких множин визначені семантичні обмеження Ф. Наступний метод використовується для налагодження нечітких множин. Кінцевий критерій вважається досягнутим, якщо рівень помилки залишається достатньо низьким впродовж деякого часу.

Крок 1. Визначити E для поточного стану системи.

Крок 2. Визначити абсолютний внесок ac_p та відносний внесок rc_p кожного p-го правила в вихід мережі o_Y:

$$ac_p = v_{j_p}^{-}(o_p), \; rc_p = \frac{o_Y - ac_p}{\eta_{\max} - \eta_{\min}}, \; v_{j_p}^{-}(o_p) = \frac{\sum\limits_{v_{j_p}(y)>0} y \cdot \min(o_p, v_{j_p}(y))}{\sum\limits_{v_{j_p}(y)>0} \min(o_p, v_{j_p}(y))},$$

де v_{j_p} – це вихідна функція приналежності j_p; j_p – номер нечіткої множини, яка є консеквентом p-го правила; $v^{-}_{j_p}(o_p)$ – псевдозворотна функція від v_{j_p} в точці o_p; o_p – значення вихідної функції нейрона R_p, p = 1, 2,...,P; o_Y – значення виходу мережі; η_{min}, η_{max} – відповідно мінімальне та максимальне можливі значення виходу керованої системи.

Крок 3. Визначити вихідне значення o_Y, застосувати цей вплив до контрольованої системи та отримати новий стан системи.

Крок 4. Визначити нове значення помилки E' та напрямок помилки τ у відповідності до нового стану системи.

Крок 5. Визначити поправку для нечітких множин v:

$$\Delta b_{j_p} = \tau \operatorname{sgn}(\eta_{\text{opt}}) |E'| o_p |o_Y - ac_p| \sigma ,$$

де b_{j_p} – центр трикутної функції приналежності v_{j_p}; j_p – номер нечіткої множини, яка є консеквентом p-го правила; τ – це напрямок помилки; E' – нове значення помилки; $\operatorname{sgn}(\eta_{\text{opt}})$ – знак оптимального виходу системи; o_p – значення вихідної функції нейрона R_p, p = 1, 2,...,P; o_Y – значення виходу мережі; ac_p – абсолютний внесок p-го правила в вихід мережі; $\sigma > 0$ – параметр навчання.

Якщо це не суперечить семантичним обмеженням Ф, знайдена поправка приймається до нечіткої множини v_{j_p}.

Крок 6. Визначити поправку для нечітких множин μ:

$$\Delta b^{(i)}_{j_p} = \tau \operatorname{sgn}(\eta_{\text{opt}})(x_i - b^{(i)}_{j_p}) |E'| rc_p \sigma ,$$

де $b^{(i)}_{j_p}$ – центр трикутної функції приналежності $\mu^{(i)}_{j_p}$; i – номер входу, i = 1, 2, ..., N; j_p – номер нечіткої, множини, яка визначена в антецеденті p-го правила для i-го входу; τ – це напрямок помилки; $\operatorname{sgn}(\eta_{\text{opt}})$ – це знак оптимального виходу системи; x_i – значення i-го входу; E' – нове значення помилки; rc_p – відносний внесок p-го правила в вихід мережі; $\sigma > 0$ – це параметр навчання.

Якщо це не суперечить семантичним обмеженням Ф, знайдена поправка приймається до нечіткої множини $\mu^{(i)}_{j_p}$.

Цей метод навчає нечіткі множини, зсуваючи їхні центри.

NEFCON має два різновиди: NEFPROX (для апроксимації функцій) і NEFCLASS (для задач класифікації).

Нейро-нечітка мережа NEFCLASS (Neuro-Fuzzy Classification – нейро-нечітка класифікація) – це тришаровий нечіткий персептрон (рис. 4.12), який має такі властивості:

– $U = \bigcup_{i=1}^{3} U_i$, де $U_1 = \{X_1, X_2, \ldots, X_N\}$ – шар входів (N нейронів), $U_2 = \{R_1, R_2, \ldots, R_P\}$ – шар правил (P – кількість правил не повинна перевищувати $P_{\max}$), $U_3 = \{C_1, C_2, \ldots, C_M\}$ – шар виходів (M – кількість класів);

– зв'язки між шарами: між першим та другим шарами $W(X_i, R) \in \left\{\mu_1^{(i)}, \ldots, \mu_{n_i}^{(i)}\right\}$, функції μ – трикутні; між шаром правил та шаром виходів $W(R_i, C_j) = 1$, якщо $j = cn_i$, $W(R_i, C_j) = 0$, якщо $j \neq cn_i$, де cn_i – клас, до якого слід віднести екземпляр згідно з і-им правилом;

– для всіх шарів функції активації нейронів $a_u = A_u(\text{net}_u) = \text{net}_u$;

– вихідна функція для всіх нейронів $o_u = O_u(a_u) = a_u$;

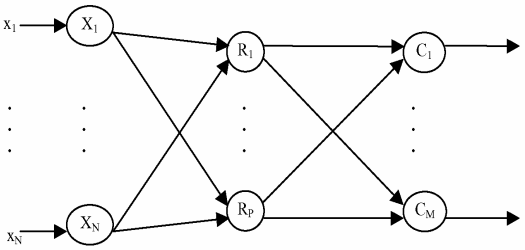


Рисунок 4.12 – Структура мережі NEFCLASS

– функція NET_u: для шару входів $\text{net}_X = \text{ex}_X$; для шару правил $net_R = \underset{X \in U_1}{T} \left(W(X,R)(o_X)\right)$, де T – t-норма; для шару виходів:

$$\text{net}_C = \left(\sum_{i=1}^{P} W(R_i, C) \cdot o_{R_i}\right)\left(\sum_{i=1}^{P} W(R_i, C)\right)^{-1}.$$

Оскільки NEFCLASS – система класифікації, то висока точність вихідних значень не є потрібною – у мережі використовується інтерпретація «переможець одержує все» для виходу. Найбільше значення серед всіх виходів береться рівним одиниці, інші – рівними нулю.

NEFCLASS використовує трикутні функції приналежності. Навчаючий метод функцій приналежності використовує міру помилки, що повідомляє, чи повинний ступінь виконання правила бути вищим або нижчим. Ця інформація використовується для зміни нечітких множин входу.

Система NEFCLASS подібна системі NEFCON, за винятком невеликих відмінностей в методі навчання й інтерпретації правил. Як і в системі NEFCON, у системі NEFCLASS ідентичні лінгвістичні значення вхідних змінних подаються тими самими нечіткими множинами.

Навчаюча вибірка подається у вигляді $L = \{\{x_1,t_1\},\{x_2,t_2\}, ..., \{x_S,t_S\}\}$, де S – кількість екземплярів, $x_i \in R^N$ – вхідні, а $t_i \in \{0,1\}^M$ – вихідні параметри.

Метод генерації правил NEFCLASS.

Крок 1. Взяти черговий екземпляр $\{x, t\}$.

Крок 2. Для кожного x_i знайти таку функцію приналежності, що $\mu_{j_i}^{(i)}(x_i) = \max\limits_{j=\{1,2,\ldots,n_i\}} \{\mu_j^{(i)}(x_i)\}$, де n_i – кількість нечітких термів, визначених для x_i.

Крок 3. Якщо кількість правил в системі менша за максимальну $P < P_{\max}$, і при цьому ще немає вузла R, такого, що: $W(x_1,R) = \mu_{j1}^{(1)}$, ..., $W(x_N,R) = \mu_{iN}^{(N)}$, тоді створити відповідне правило, встановивши зв'язки $W(R,c_j)$ відповідно до значень t.

Крок 4. Якщо ще є необроблені екземпляри і кількість правил в системі менша за максимальну: $P < P_{\max}$, тоді перейти до кроку 1, інакше – зупинити процес.

Метод навчання з додаванням правил, застосовуваний для NEFCLASS, є набагато менш витратним у порівнянні з методом навчання зі скороченням правил, застосовуваним для NEFCON.

Метод навчання нечітких множин системи NEFCLASS.

Крок 1. Подати на вхід системи NEFCLASS черговий екземпляр $\{x, t\}$.

Крок 2. Для кожного з виходів обчислити помилку $\delta_{C_i} = t_i - o_{C_i}$.

Крок 3. Для кожного нейрона R, в якого $o_R > 0$:

– визначити поправку правила:

$$\delta_R = o_R(1-o_R) \sum_{C \in U_3} W(R,C)(\delta_C),$$

де o_R – значення вихідної функції нейрона R; $W(R,C)$ – вага зв'язку між нейронами C та R; R – нейрон шару правил; C – нейрон шару виходів; δ_C – помилка виходу для нейрона C;

– знайти нейрон X', для якого:

$$W(X',R)(o_{X'}) = \min_{X \in U_1} \{W(X,R)(o_X)\};$$

– для нечіткої множини $W(X', R)$ знайти поправки для параметрів a, b, c: $\delta_b = \sigma \cdot \delta_R \cdot (c-a) \cdot \operatorname{sgn}(o_{X'} - b)$, $\delta_a = -\sigma \cdot \delta_R \cdot (c-a) + \delta_b$, $\delta_c = \sigma \cdot \delta_R \cdot (c-a) + \delta_b$, де $\sigma > 0$ – параметр навчання; a, b, c – параметри трикутної функції приналежності; $\operatorname{sgn}(o_{X'} - b)$ – напрямок поправки; $o_{X'}$ – значення вихідної функції нейрона X'.

Поправки приймаються для ваги зв'язку $W(X', R)$, якщо вони не суперечать визначеним семантичним обмеженням Ф.

Крок 4. Якщо чергову ітерацію закінчено та досягнуто кінцевий критерій, тоді процес зупиняється, інакше – перейти до кроку 1 (критерієм зазвичай вважається кількість помилок класифікації).

У порівнянні з нейронними мережами NEFCLASS використовує набагато більш просту стратегію навчання. При пошуку правил не виконується квантування

векторів (кластерів) і не потрібно обчислювати значення градієнта для навчання функцій приналежності. Додатковими перевагами NEFCLASS є інтерпретабільність, можливість ініціалізації, що враховує апріорні знання, а також простота.

Нейро-нечітка мережа NEFPROX (Neuro-Fuzzy Function Approximation – нейро-нечітка апроксимація функцій) – це модель нейро-нечіткої системи (рис. 4.13), що може апроксимувати будь-яку $N \times M$ – функцію.

Модель побудована на загальній моделі нечіткого персептрона і має такі властивості:

– $U = \bigcup_{i=1}^{3} U_i$, де $U_1 = \{X_1, X_2, \ldots, X_N\}$ – шар входів (N нейронів), $U_2 = \{R_1, R_2, \ldots, R_P\}$ – шар правил (P – кількість правил), $U_3 = \{Y_1, Y_2, \ldots, Y_M\}$ – шар виходів (один нейрон);

– для всіх шарів функції активації нейронів $a_u = A_u(\text{net}_u) = \text{net}_u$;

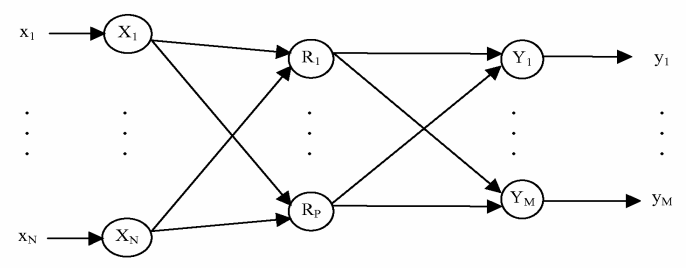


Рисунок 4.13 – Мережа NEFPROX

– зв'язки між шарами: між першим та другим шарами $W(X_i, R) \in \{\mu_1^{(i)}, \ldots, \mu_{n_i}^{(i)}\}$, функції μ – трикутні; між шаром правил та шаром виходів $W(X_j, R) \in \{v_1^{(j)}, \ldots, v_{n_j}^{(j)}\}$, функції v – трикутні.

– вихідна функція: для шару входів та шару правил $o_u = O_u(a_u) = a_u$; для шару виходів $o_Y = O_Y(\text{net}_Y) = \text{DEFUZZ}_Y(\text{net}_Y)$, де DEFUZZ_Y – відповідна функція дефазифікації.

– функція NET_u: для шару входів $\text{net}_X = \text{ex}_X$; для шару правил $\text{net}_R = \underset{X \in U_1}{T} (W(X,R)(o_X))$, де T – t-норма. Для шару виходів $\text{net}_Y(x) = \underset{R \in U_2}{\perp} (T(o_R, W(R,Y)(x)))$, де $\perp$ – це t-конорма.

Система NEFPROX заснована на контрольованому навчанні і являє собою змінену версію моделі NEFCON без навчання з підкріпленням. Модель NEFPROX подібна до NEFCON і NEFCLASS за винятком того, що NEFCON має тільки один вихідний вузол, а система NEFCLASS не використовує функції приналежності в консеквентах.

Задача навчання задана у вигляді $L = \{\{x_1,y_1\},\{x_2,y_2\},...,\{x_s,y_s\}\}$, s – кількість екземплярів, $x_i \in R^N$ – вхідні параметри, $y_i \in R^M$ – вихідні параметри.

Метод генерації правил NEFPROX.

Крок 1. Взяти черговий екземпляр $\{x, t\}$.

Крок 2. Для кожного x_i знайти функцію приналежності таку, що $\mu_{j_i}^{(i)}(x_i) = \max\limits_{j=\{1,2,..,n_i\}} \{\mu_j^{(i)}(x_i)\}$, де n_i – кількість нечітких термів, визначених для x_i.

Крок 3. Якщо кількість правил в системі менша за максимальну $P < P_{\max}$, і при цьому ще немає вузла R такого, що: $W(x_1,R) = \mu_{j1}^{(1)}$, ..., $W(x_N,R) = \mu_{jN}^{(N)}$, тоді створити таке правило.

Крок 4. Для кожного зв'язку нового вузла R з вихідними вузлами знайти необхідну нечітку множину згідно з процедурою: серед всіх функцій приналежності, визначених для виходу Y_i знайти $v_{j_i}^{(i)}$ таку, що: $v_{j_i}^{(i)}(t_i) = \max\limits_{j\in\{1,..,n_i\}} \{v_j^{(i)}(t_i)\}$ та $v_{j_i}^{(i)}(t_i) \ge 0{,}5$.

Якщо не знайдено такої функції приналежності, тоді створити нову $v_{\text{new}}^{(i)}$ таку, що: $v_{\text{new}}^{(i)}(t_i) = 1$, додати її до функцій приналежності, визначених для виходу Y_i та встановити: $W(R,Y_i) = v_{\text{new}}^{(i)}$.

Крок 5. Якщо ще є необроблені екземпляри, тоді перейти до кроку 1, інакше – зупинити процес.

Крок 6. Після закінчення для кожного виходу кожного правила $W(R,Y)$ перевірити, чи є серед функцій приналежності v_j, визначених для виходу Y, така, значення якої є більшим ніж у тієї, яка зараз встановлена. Якщо є, тоді змінити зв'язок $W(R, Y)$.

Метод навчання нечітких множин системи NEFPROX.

Крок 1. Подати на вхід системи NEFPROX черговий екземпляр $\{x, y\}$.

Крок 2. Для кожного з виходів обчислити помилку $\delta_{Y_i} = y_i - o_{Y_i}$.

Крок 3. Для кожного нейрона з шару правил R, в якого $o_R > 0$:

– для кожного вихідного вузла Y_i знайти поправки для параметрів a, b, c: $\delta_{b_i} = \sigma\delta_{Y_i}(c-a)o_R(1-W(R,Y_i)y_i)$, $\delta_{a_i} = -\sigma o_R(c-a)+\delta_{b_i}$, $\delta_{c_i} = \sigma o_R(c-a)+\delta_{b_i}$, де $\sigma > 0$ – параметр навчання; δ_{Y_i} – помилка i-го виходу; a, c – параметри трикутної функції приналежності; o_R – значення вихідної функції нейрона R; $W(R,Y_i)$ – вага зв'язку між нейроном R та Y_i; y_i – значення i-го виходу чергового екземпляру. Якщо поправки не суперечать визначеним семантичним обмеженням Ф, прийняти їх для зв'язку $W(R, Y_i)$.

– знайти поправку правила:

$$\delta_R = o_R(1-o_R) \sum_{Y\in U_3} (2W(R,Y)(y_i)-1)|\delta_Y|,$$

де o_R – значення вихідної функції нейрона R; Y – нейрон шару виходів; $W(R, Y)$ – вага зв'язку між нейронами R та Y; y_i – значення i-го виходу чергового екземпляру; δ_Y – помилка виходу, з яким пов'язаний нейрон Y;

– для всіх нечітких множин $W(X, R)$, для яких $W(X, R)(o_X) > 0$, знайти поправки для параметрів a, b, c: $\delta_b = \sigma\delta_R(c-a)(1-W(X,R)o_X)\text{sgn}(o_X - b)$, $\delta_a = -\sigma\delta_R(c-a)(1-W(X,R)o_X) + \delta_b$, $\delta_c = \sigma\delta_R(c-a)(1-W(X,R)o_X) + \delta_b$, де $\sigma > 0$ – параметр навчання; δ_R – поправка правила; a, b, c – параметри трикутної функції приналежності; $W(X, R)$ – вага зв'язку між нейроном X та R; o_X – значення вихідної функції нейрона X; $\text{sgn}(o_X - b)$ – це напрямок зсуву нечіткої множини; та прийняти ці поправки для зв'язків $W(X, R)$, якщо вони не суперечать визначеним семантичним обмеженням Ф.

Крок 4. Якщо чергова ітерація закінчена та досягнутий кінцевий критерій, тоді процес зупиняється, інакше – перейти до кроку 1.

Нейро-нечітка мережа NNDFR (Neural Network Driven Fuzzy Reasoning – нейронна мережа для нечітких умовиводів), що запропонована Ткагі (H. Takagi) та Хаяши (J. Hayashi), здатна виконувати кластеризацію в просторі, що містить нелінійні межі кластерів (рис. 4.14).

На входи мережі подаються значення ознак розпізнаваного екземпляра. Перший і другий шари мережі містять V двошарових персептронів f_v з одним схованим шаром, що реалізують часткові моделі, тобто виражають правила: Якщо $x^s \in C^v$, то $y^s = f_v(x^s_1, x^s_2, \ldots, x^s_N)$, $v = 1, 2, \ldots, V$.

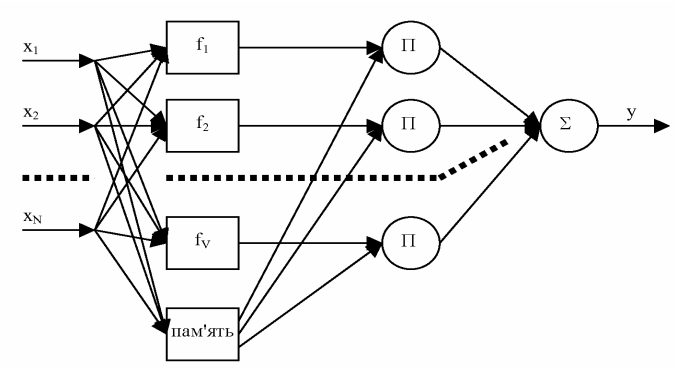


Рисунок 4.14 – Структура мережі NNDFR

Крім того на першому і другому шарах розміщується багатошаровий персептрон пам'яті (замість багатошарового персептрона для пам'яті можливе використання й інших типів нейромереж), що слугує для опису бажаної

кривої поверхні рішень – розподіляє простір ознак на кластери. Нейрони третього шару виконують вибір персептронної моделі для кожного кластера. Четвертий шар містить один нейрон, що здійснює об'єднання f_v.

Метод «видалення правил назад» використовується для навчання NNDFR і містить такі кроки.

Крок 1. Формування навчаючої вибірки $\{x^s, y^s\}$.

Крок 2. Навчання багатошарового персептрона пам'яті. Евристично визначається кількість кластерів (правил). Багатошаровий персептрон пам'яті навчається виділяти кластери. Якщо x^s належить до C^v, то v-ий вихід персептрона пам'яті дорівнює 1, у противному випадку – вихід дорівнює 0.

Крок 3. Багатошарові персептрони f_v навчаються окремо методом зворотного поширення помилки з метою мінімізації помилки.

Крок 4. Після окремого навчання багатошарові персептрони поєднуються і навчаються разом.

Крок 5. Виконується видалення малозначимих вхідних змінних для кожного f_v: якщо зміна деякої вхідної змінної істотно не впливає на помилку, тоді відповідна змінна видаляється.

Нейро-нечітка мережа GARIC (Generalized Approximate Reasoning based Intelligent Control – інтелектуальне керування на основі узагальненого приблизного виведення) – це розширена версія ARIC (Approximate Reasoning based Intelligent Control – інтелектуальне керування на основі наближеного виведення), що реалізує нечіткий контролер, використовуючи дві спеціалізовані нейронні мережі прямого поширення, які мають різне функціональне призначення: мережу оцінювання стану дій (Action state Evaluation Network) і мережу вибору дії (Action Selection Network).

Місце GARIC у контурі керування показане на рис. 4.15.

Мережа вибору дії GARIC – це п'ятишарова мережа прямого поширення (рис. 4.15, *б*), яка вибирає ситуативний вплив (дію) у, ґрунтуючись на стані керованого процесу, з метою максимізації v – передбачення майбутнього підкріплення як функції параметрів мережі вибору дії.

Вхідний шар складають вхідні змінні. Перший схований шар зберігає лінгвістичні значення усіх вхідних змінних. Кожний вхідний вузол зв'язаний тільки з тими вузлами першого схованого шару, що містять асоційовані з ним лінгвістичні значення. Другий схований шар містить вузли нечітких правил, що визначають ступінь виконання правил, використовуючи *операцію softmin*:

$$\mathrm{softmin}(x_1, x_2, ..., x_N) = \frac{\sum_{i=1}^{N} x_i \exp(-\alpha x_i)}{\sum_{i=1}^{N} \exp(-\alpha x_i)}.$$

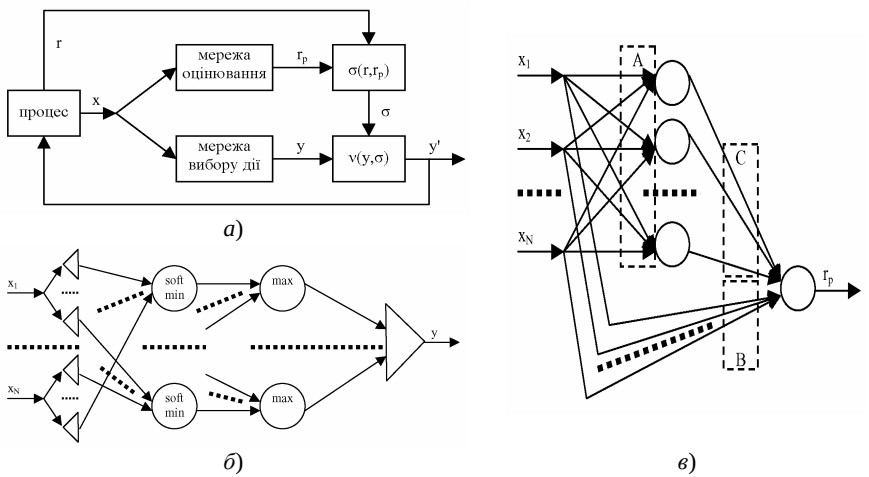


Рисунок 4.15 – Нейро-нечіткий контролер GARIC:
а) контур керування; б) мережа вибору дії; в) мережа оцінювання

Висновки правил обчислюються в залежності від сили їхніх антецедентів, обчислених шаром вузлів правил. Третій схований шар містить лінгвістичні значення вихідної змінної керування. Вихідний шар виконує дефазифікацію: $y = \mathrm{MOM}(\mu_i, y_i)$, $i = 1, 2, \ldots, m$, де μ_i – ступінь приналежності вхідного вектора до i-го правила, y_i – значення i-го терму вихідної змінної.

Мережа вибору не використовує зважені зв'язки, але процес навчання змінює параметри, що зберігаються вузлами мережі. Мережа навчається за допомогою методу зворотного поширення помилки.

Мережа оцінювання (рис. 4.15, *в*) – це адаптивний критик або передбачувач, що оцінює результати мережі вибору дії і передбачає майбутнє підкріплення. Мережа оцінювання являє собою лінійний багатошаровий персептрон. На вхід мережа одержує поточний стан процесу $x = (x_1, \ldots, x_N)$. Мережа передбачає придатність стану r_p, що потім з'єднується із сигналом зовнішньої помилки r для розрахунку внутрішнього підкріплення σ.

Виходи нейронів першого схованого шару визначаються за формулою:

$$z_j(t, t+1) = \psi\left(\sum_{i=1}^{N} a_{ij}(t) x_i(t+1)\right), \; j = 1,2,\ldots,N_1,$$

де $\psi(x) = (1 + \exp(-x))^{-1}$ – сигмоїдна функція активації.

Вихідний шар одержує на вході значення як із вхідного $\{x_i\}$, так і з першого схованого шару $\{z_j\}$. Вихід мережі r_p – це міра придатності стану, передбачення майбутнього підкріплення:

$$r_p(t-1,t)=\sum_{i=1}^{N} b_i(t-1)x_i(t)+\sum_{j=1}^{N_1} c_j(t-1)z_j(t).$$

Передбачення майбутнього підкріплення використовується при навчанні функцій приналежності.

Сигнал неузгодженості (внутрішнє підкріплення) визначається як:

$$\sigma(r,r_p)=\begin{cases}0, t=0,\\ r(t)-r_p(t-1,t-1), \text{стан відмови},\\ r(t)-\gamma r_p(t-1,t)-r_p(t-1,t-1), \text{інакше},\end{cases}$$

де γ – евристичний фактор: $0 \le \gamma \le 1$, $\sigma(r, r_p)$ – різниця реального і прогнозованого «внутрішнього» підкріплень r.

Для навчання мережі оцінювання використовується змішаний пошук на основі градієнтного спуску і навчання з підкріпленням для настроювання параметрів вузлів. Гібридне навчання зупиняється, якщо вихід мережі оцінювання припиняє змінюватися. Матриці $B = \{b_i\}$ та $C = \{c_j\}$ змінюються стратегією нагорода-штраф: $b_i(t) = b_i(t-1)+\beta\sigma(t)x_i(t-1)$, $c_j(t) = c_j(t-1)+\beta\sigma(t)z_j(t-1,t-1)$, $\beta > 0$. Матриця $A = \{a_{ij}\}$ навчається спрощеним методом зворотного поширення помилки: $a_{ij}(t) = a_{ij}(t-1) + \beta\sigma(t)z_j(t-1,t-1)(1 - z_j(t-1, t-1))\text{sgn}(c_j(t-1))x_i(t-1)$, де $\text{sgn}(x) = 1$, якщо $x > 0$; $\text{sgn}(x) = 0$, інакше.

Стохастичний модифікатор впливу (stochastic action modifier) виконує випадкову модуляцію $y' = v(y,\sigma)$ сигналу y колоколообразною випадково розподіленою функцією з центром y та розкидом $\sigma(r, r_p) = \exp(-r_p(t-1))$, що необхідно для покриття вхідного простору і поліпшення узагальнюючої здатності. Замість фактичного виходу $y(t)$ до керованого процесу додається керуючий вплив $y'(t)$.

Різниця $y'(t) - y(t)$ є великою, коли внутрішнє підкріплення $r_p(t)$ є низьким, і малою, коли $r_p(t)$ є високим. Тобто, якщо останній вплив, прийнятий контролером був поганим, тоді керуючий вплив відхиляється сильніше; а коли попередня дія була гарною, тоді дається тільки невелике відхилення. Чисельно кількість відхилення визначається з формули: $s(t) = (y'(t) - y(t))\exp(r_p(t-1))$, і використовується як коефіцієнт швидкості навчання при відновленні параметрів мережі вибору дії.

Недоліком GARIC є складність архітектури і процедури навчання.

Нейро-нечітка мережа FINEST (Fuzzy Inference Environment Software with Tuning – Програмне забезпечення середовища нечіткого виведення з настроюванням) розроблена для настроювання параметрів нечіткого виведення, і припускає настроювання нечітких предикатів і функцій комбінування, а також настроювання функції імплікації.

На рис. 4.16 зображено схему мережі FINEST.

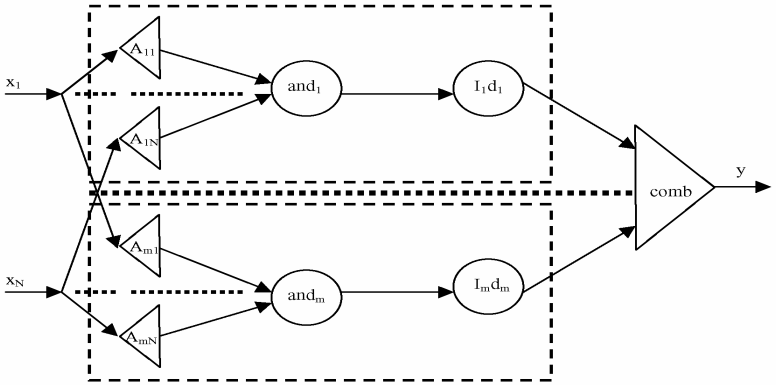


Рисунок 4.16 – Схема мережі FINEST

Вхідні значення x_i – це факти, а вихідне значення y – висновок нечіткого виведення. Перший шар виконує фазифікацію. Другий шар агрегує оцінки істинності передумов кожного i-го правила. Третій шар формує висновок правила I. Четвертий шар комбінує всі правила.

На рис. 4.16 використовуються позначення: and_i – функція, що характеризує оператор агрегації i-го правила; I_i – функція імплікації i-го правила; comb – глобальна функція комбінування. Функції and_i, I_i, comb та функції приналежності кожного нечіткого предиката визначаються з деякими параметрами.

Найбільш важливі особливості мережі:

– Удосконалений узагальнений modus ponens: 1) оператори агрегації мають об'єднуючий та скорочуючий характер; 2) використовується параметризована функція імплікації; 3) використовується функція комбінування, що здатна зменшувати нечіткість; 4) формування ланцюжка зворотного виведення здійснюється на основі узагальненого modus ponens.

– Оператори агрегації з об'єднуючим і скорочуючим характером визначені з використанням деяких параметрів, що вказують силу ефекту об'єднання, область, на котру впливає ефект, і т. д., а також розроблений механізм настроювання цих параметрів. У такий же спосіб механізм настроювання може налагоджувати функцію імплікації і функцію комбінування.

– Розроблене середовище програмного забезпечення й методи для підтримки формування ланцюжків прямого і зворотного виведення, заснованого на поліпшеному узагальненому modus ponens, а також для настроювання різних параметрів системи.

Для налагодження параметрів мережі FINEST використовується метод зворотного поширення помилки. Можливо налагодити будь-який параметр вузлів мережі, якщо відома функція похідної щодо параметрів. Таким чином,

FINEST забезпечує механізм, заснований на поліпшеному узагальненому modus ponens для настроювання нечітких предикатів і функцій комбінування, а також для настроювання функції імплікації. Для належного застосування методу настроювання необхідна параметризація процедури виведення.

Нейро-нечітка мережа FUN (FUzzy Net – нечітка мережа) складається з вхідного, трьох схованих і вихідних шарів. Нейрони кожного шару мають різні функції активації, що відповідають різним стадіям в обчисленні нечіткого висновку. Функція активації може бути окремо обрана для кожної задачі. Для впровадження нечітких правил і функцій приналежності в мережі визначені спеціальні нейрони, що за допомогою своїх функцій активації можуть оцінювати логічні вирази.

Мережа ініціалізується нечіткою базою знань і відповідними функціями приналежності. Рис. 4.17 ілюструє мережу FUN.

Вхідні вузли мережі відповідають вхідним змінним. Нейрони першого схованого шару містять функції приналежності і виконують фазифікацію вхідних змінних. Другий схований шар визначає кон'юнкції (нечітке «ТА»). Функції приналежності вихідних змінних визначаються третім схованим шаром. Нейрони третього шару мають функції активації нечітке «АБО». Нейрони вихідного шару здійснюють дефазифікацію вихідних змінних.

Правила і функції приналежності використовуються для побудови початкової мережі FUN. База правил потім оптимізується за допомогою зміни структури мережі або даних у нейронах. Для навчання правил змінюються зв'язки між правилами і нечіткі значення. Для навчання функцій приналежності змінюються параметри вузлів у першому і третьому схованих шарах. Мережа FUN може бути навчена одним зі стандартних методів навчання нейронних мереж із учителем.

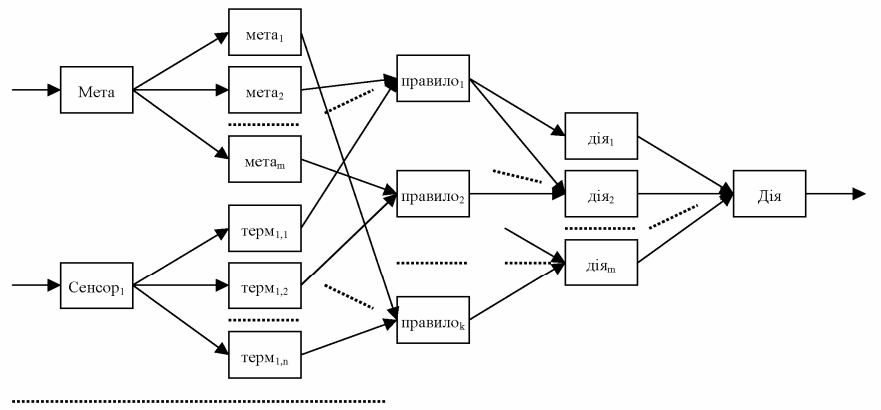


Рисунок 4.17 – Мережа FUN

Правила реалізуються в мережі за допомогою зв'язків між шарами. Навчання правил реалізоване як стохастичний пошук у просторі правил: змінюється випадково обраний зв'язок, після чого визначається значення цільової функції. Якщо значення цільової функції погіршилося, зміна скасовується, у протилежному випадку – приймається. Пошук виконується доти, поки не буде досягнуте бажане значення цільової функції.

Оскільки метод навчання повинний зберігати семантику правил, він повинний здійснюватися таким способом, щоб у тому самому правилі не могли виникнути два значення однієї і тієї ж змінної. Це досягається шляхом обміну зв'язками між різними значеннями однієї і тієї ж змінної. Мережа FUN використовує змішаний пошук на основі градієнтного спуску і стохастичного пошуку для модифікації функцій приналежності. Максимальна зміна у випадковому напрямку спочатку призначається на всі дескриптори функцій приналежності. Далі у випадковому порядку вибирають один дескриптор функції приналежності однієї лінгвістичної змінної, після чого оцінюють якість роботи мережі з цим дескриптором, зміненим відповідно до припустимої зміни для даного дескриптора. Якщо робота мережі покращилася, тоді нове значення приймається і наступного разу інша зміна здійснюється в тому ж самому напрямку. У протилежному випадку, якщо мережа погіршує роботу, зміна цілком скасовується. Щоб гарантувати збіжність, зміни зменшуються після кожного кроку навчання і скорочуються асимптотично до нуля у відповідності з коефіцієнтом швидкості навчання.

Як видно, система FUN ініціалізується задаванням фіксованих кількості правил та кількості початкових нечітких множин для кожної змінної. Після чого мережа навчається за допомогою стохастичної процедури, що випадково змінює параметри функцій приналежності та зв'язків у межах структури мережі.

Нейро-нечітка мережа EFuNN (Evolving Fuzzy Neural Network – нечітка нейронна мережа, що розвивається) заснована на моделі ECOS (Evolving COnnectionist Systems – коннекціоністська система, що розвивається) для формування адаптивних інтелектуальних систем на основі розвитку і росту, а також гібридного (контрольованого / неконтрольованого) навчання. Мережа може доповнюватися новими вхідними даними, включаючи нові ознаки, нові класи і т. д., за допомогою локального настроювання елементів.

Мережа EFUNN має п'ятишарову архітектуру (рис. 4.18). У мережі EFUNN усі вузли створюються в процесі навчання.

Перший (вхідний) шар містить вхідні змінні.

Другий шар вузлів містить нечіткі квантувачі діапазону кожної вхідної змінної.

Кожна вхідна змінна подається тут групою просторово упорядкованих нейронів, щоб подати нечітке квантування цієї змінної. Вузли, що відповідають функціям приналежності (трикутній, гаусовій і т. д), можуть бути змінені в процесі навчання.

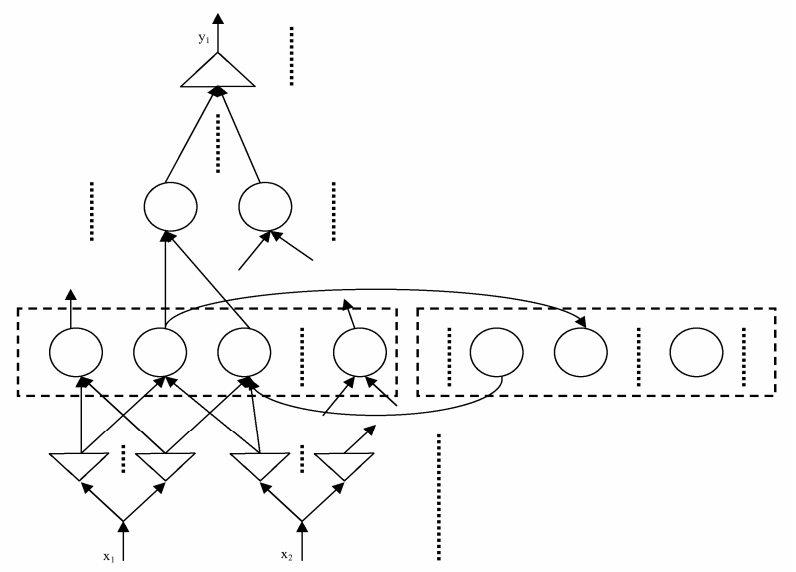


Рисунок 4.18 – Мережа EFuNN

Третій шар містить вузли правил, що розвиваються в процесі контрольованого або неконтрольованого гібридного навчання. Вузли правил подають прототипи асоціацій даних «вхід–вихід», графічно зображених як зв'язки між просторами вузлів нечіткого входу і нечіткого виходу. Кожен вузол правила r визначається двома векторами ваг зв'язків: $w_1(r)$ та $w_2(r)$. Вектор $w_2(r)$ налагоджується за допомогою контрольованого навчання, заснованого на помилці виходу, а вектор $w_1(r)$ налагоджується за допомогою неконтрольованого навчання, заснованого на мірі подоби в межах локальної області вхідного простору задачі.

Четвертий шар нейронів здійснює нечітке квантування вихідних змінних. На виході п'ятого шару формуються чіткі значення вихідних змінних.

Запропоновано два способи поширення активації вузлів. У способі «1 з *n*» у EFUNN максимальна активація вузла правила поширюється до наступного шару. У способі «багато з *n*» усі значення активацій вузлів правил, що вище порога активації, поширюються далі в структурі зв'язків.

Нейро-нечітка мережа dmEFuNN (Dynamic Evolving Fuzzy Neural Network – нечітка нейронна мережа, що динамічно розвивається) – це змінена версія мережі EFUNN, заснована на тому, що поширюється не тільки активація вузла правила-переможця, але також для кожного нового вхідного вектора динамічно відбирається група вузлів правил і значення їхніх активацій використовуються для того, щоб обчислити динамічні параметри вихідної

функції. На відміну від мережі EFUNN, що використовує зважені нечіткі правила Мамдані, мережа dmEFuNN використовує нечіткі правила Такагі-Сугено. Архітектуру мережі зображено на рис. 4.19.

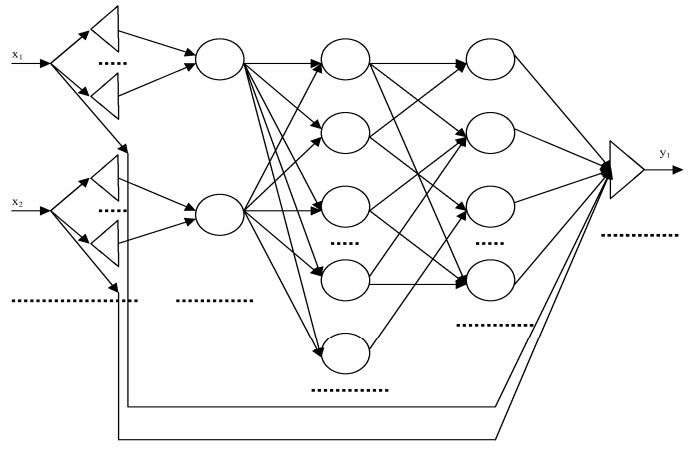


Рисунок 4.19 – Мережа dmEFuNN

Перший (входи), другий (нечіткі терми) і третій (вузли, що розвиваються) шари dmEFuNN мають точно таку ж структуру і функції як у EFUNN. Четвертий шар (нечіткий шар виведення) вибирає *m* вузлів правил із третього шару, що мають найближчу нечітку нормалізовану локальну відстань до нечіткого вхідного вектора. П'ятий шар формує нечіткі правила Такагі-Сугено, використовуючи зважену середньоквадратичну помилку. Останній шар обчислює вихід dmEFuNN. Кількість активованих вузлів *m*, що використовуються для обчислення значень виходу dmEFuNN, є не меншою ніж збільшена на одиницю кількість вхідних вузлів.

Подібно до EFUNN мережа dmEFuNN може навчатися як з оптимізацією глобальної так і локальної помилки.

У dmEFuNN для нового вхідного вектора, для якого вихідний вектор є невідомим, підпростір складається з *m* знайдених вузлів правил і нечітке правило Такагі-Сугено першого порядку формується з використанням методу найменших квадратів. Це правило використовується для обчислення вихідного значення dmEFuNN.

Таким чином, dmEFuNN діє як універсальний апроксиматор функцій, що використовує *m* лінійних функцій у невеликому *m*-мірному підпросторі вузла. Точність апроксимації залежить від розміру підпросторів вузлів: чим менше підпростір, тим вище точність. Це означає, що, коли є достатня кіль-

кість навчаючих векторів даних і створена достатня кількість вузлів правил, тоді прийнятна точність може бути досягнута.

Гібридна мережа з нечіткою самоорганізацією поєднує в собі шар з нечіткою самоорганізацією, що виконує функції препроцесора, і багатошаровий (звичайно двошаровий) персептрон у якості постпроцесора.

Схему мережі зображено на рис. 4.20 (пунктиром виділено шар, що самоорганізується, а штрих-пунктиром – багатошаровий персептрон). Мережа має N входів, на які надходять значення ознак розпізнаваного екземпляра.

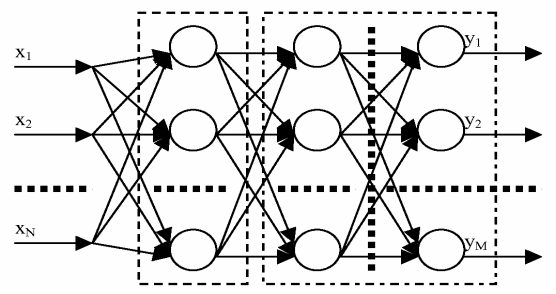


Рисунок 4.20 – Структура гібридної нечіткої мережі

Шар, що самоорганізується, містить V нейронів, кожний з яких відповідає центру визначеного кластера. Виходи шару, що самоорганізується, використовуються як входи багатошарового персептрона. Якщо на вхід мережі подається вектор x^s, то на виході шару з самоорганізацією формується вектор u, що складається з коефіцієнтів приналежності x^s до центрів кластерів C^v: $u = \{u^s_v\}$.

Сховані шари персептрона містять N_η нейронів, де η – номер шару персептрона. Кількість входів персептрона гібридної мережі дорівнює кількості нейронів, що самоорганізуються. Кількість схованих шарів і число нейронів у цих шарах може бути, у принципі, довільною, хоча звичайно для відновлення даних з необхідною точністю достатньо одного шару.

Вихідний (останній) шар персептрона містить M нейронів, кількість яких залежить від розмірності вектора цільової ознаки y^s (тобто кількості сигналів y^s_i – складових вихідного вектора y^s), зіставленого вхідному вектору x^s.

На практиці гібридна мережа, як правило, є більш ефективною, ніж одиночна мережа з нечіткою самоорганізацією і ніж самостійний багатошаровий персептрон. Цей висновок випливає з факту, що функціонування гібридної мережі розподіляється на два незалежних етапи, реалізованих окремо.

На етапі самоорганізації простір вхідних даних розподіляється на кластери, при цьому кількість кластерів (нейронів, що самоорганізуються) може бути довільною і визначатися умовами розв’язуваної задачі.

Багатошаровий персептрон приписує кожній групі кластерів відповідний їй очікуваний результат. Наприклад, при вирішенні задачі класифікації це може виглядати як віднесення до одного конкретного класу декількох кластерів даних.

Метод навчання гібридної мережі складається з двох етапів.

На першому етапі проводиться навчання шару, що самоорганізується, що складається в підборі позицій центрів кластерів. Для цього можна застосовувати як метод FCM, так і метод Густавсона-Кесселя. По завершенні першого етапу, коли стабілізувалися значення коефіцієнтів приналежності усіх векторів, що відповідають вхідним сигналам для багатошарового персептрона, починається другий етап навчання.

На другому етапі значення параметрів частини мережі, що самоорганізується, залишаються незмінними, а уточнюються тільки ваги нейронів персептрона. Це звичайне навчання багатошарового персептрона, для якого входом є множина коефіцієнтів приналежності вектора x^s до центрів шару, що самоорганізується. У залежності від типу розв'язуваної задачі виходом мережі може бути код класу, до якого належить вхідний вектор x^s, або дійсне значення, зіставлене вектору x^s. По завершенні другого етапу навчання ваги заморожуються і мережа стає готовою до функціонування в режимі експлуатації.

4.7 Синтез ефективних нейро-нечітких моделей

Нейро-нечіткі мережі поєднують у собі концепції нечітких систем та нейромереж: вони дозволяють одержувати моделі, у яких виведення здійснюється на основі апарата нечіткої логіки, але відповідні функції приналежності налагоджуються з використанням методів навчання нейромереж. Такі моделі є логічно прозорими, але відомі методи їхнього синтезу є дуже повільними і високо ітеративними. Крім того, точність і узагальнюючі властивості нейронечітких моделей, одержуваних на основі традиційних методів, є недостатніми і залежать як від тривалості навчання, так і від кількості термів для кожної з вхідних змінних, що задається користувачем.

Тому актуальною є розробка методів, що дозволяють у неітеративному режимі (без тривалої оптимізаційної підгонки параметрів) синтезувати ефективні нейро-нечіткі моделі з високими узагальнюючими властивостями.

Для синтезу нейро-нечітких моделей у неітеративному режимі пропонується враховувати апріорну інформацію про екземпляри навчаючої вибірки, а скорочення надмірності та підвищення якості апроксимації і рівня узагальнення моделей пропонується досягати шляхом редукції блоків мережі, що дублюються.

4.7.1 Апріорна інформація про навчаючу вибірку

Нехай ми маємо навчаючу вибірку x, що складається з S екземплярів x^s, де s – номер екземпляра, $s = 1, 2, ..., S$. Кожен s-ий екземпляр будемо характеризувати набором значень N ознак x^s_j, де j – номер ознаки s-го екземпляра, $j = 1, 2, ..., N$. Крім того, кожному екземпляру x^s зіставимо цільову ознаку y^{s*} – номер класу s-го екземпляра.

До апріорної інформації про екземпляри навчаючої вибірки можна віднести показники індивідуальної інформативності ознак, а також параметри, що характеризують межі областей групування екземплярів.

Розіб'ємо інтервал значень кожної ознаки екземплярів навчаючої вибірки на інтервали, у яких номер класу залишається незмінним. Це дасть нам можливість визначити, з одного боку, скільки буде потрібно поділяючих площин, перпендикулярних вісі кожної ознаки, а з іншого боку, дозволить визначити ліву і праву межі інтервалів для класів за віссю кожної ознаки. Кількість інтервалів, а також значення меж і номера класів інтервалів для кожної ознаки можна знайти, виконавши кроки 1–12.

Крок 1. Ініціалізація. Задати навчаючу вибірку екземплярів, подану у вигляді масиву даних x, у якому ознаки лінеаризовано за рядками, а екземпляри – за стовпцями, а також відповідний масив $y^* = \{y^{s*}\}$, що містить номера класів, зіставлені екземплярам навчаючої вибірки. Створити масив $\{D_j\}$, рівний за розміром кількості ознак N, елементи якого будуть містити кількість інтервалів для кожної ознаки. Установити: $D_j = 0$, $j = 1, 2, ..., N$, де j – номер поточної ознаки. Занести кількість екземплярів навчаючої вибірки до змінної S. Установити номер поточної ознаки: $i = 1$.

Крок 2. Якщо $i \le N$, тоді перейти до кроку 3, інакше – перейти до кроку 12.

Крок 3. Занести до буфера ознаки x вектор значень i-ої ознаки з навчаючої вибірки: $x(j) = x^s_i$; занести до буфера класу у копію масиву y^*: $y(s) = y^{s*}$, $s = 1, 2, ..., S$.

Крок 4. Відсортувати масиви x та y в порядку зростання масиву x (кроки 4.1–4.7 реалізують найпростіший метод бульбашкового сортування, який можна замінити на практиці більш швидким методом).

Крок 4.1 Установити номер поточного екземпляра: $s = 1$.

Крок 4.2 Якщо $s \le S$, тоді перейти до кроку 4.3, інакше – до кроку 5.

Крок 4.3 Установити номер поточного екземпляра: $k = s + 1$.

Крок 4.4 Якщо $k \le S$, тоді перейти до кроку 4.5, у протилежному випадку – перейти до кроку 4.7.

Крок 4.5 Якщо $x(s) > x(k)$, тоді установити: $z = x(s)$, $x(s) = x(k)$, $x(k) = z$, $z = y(s)$, $y(s) = y(k)$, $y(k) = z$, де z – буферна змінна.

Крок 4.6 Установити: $k = k + 1$. Перейти до кроку 4.4.

Крок 4.7 Установити: $s = s + 1$. Перейти до кроку 4.2.

Крок 5. Установити: $s = 1$, $k = 1$.

Крок 6. Якщо $s \le S$, тоді установити: $a^t = x(s)$, де a^t – буфер для збереження лівої межі k-го інтервалу i-ої ознаки, та перейти до кроку 7, у протилежному випадку – перейти до кроку 11.

Крок 7. Поки $s < S$ та $y(s) = y(s + 1)$ виконувати: $s = s + 1$.

Крок 8. Якщо $s = S$ та $y(s) = y(s–1)$, тоді установити: $K(i, k) = y(s)$, $A(i, k) = a^t$, $B(i, k) = x(s)$, $k = k+1$, $s = s+1$, перейти до кроку 10. Тут $K(i, k)$ – номер класу, зіставлений екземплярам навчаючої вибірки, значення i-ої ознаки яких попадає усередину k-го інтервалу; $A(i, k)$ і $B(i, k)$ – ліва і права межі k-го інтервалу i-ої ознаки, відповідно.

Крок 9. Якщо $s < S$ та $y(s) \neq y(s+1)$, тоді установити: $K(i, k) = y(s)$, $A(i, k) = a^t$, $B(i, k) = x(s)$, $k = k + 1$, $s = s + 1$, $D_i = D_i + 1$, у протилежному випадку – установити: $K(i, k) = y(s)$, $A(i, k) = x(s)$, $B(i, k) = x(s)$, $k = k + 1$, $s = s + 1$.

Крок 10. Перейти до кроку 6.

Крок 11. Установити: $i = i+1$, перейти до кроку 2.

Крок 12. Зупинення.

У результаті виконання кроків 1–12 для навчаючої пари $\{x, y\}$ ми одержимо масив $\{D_j\}$, що містить для кожної ознаки кількість інтервалів, на які вона розбивається, а також масиви $\{A(i, k)\}$, $\{B(i, k)\}$ та $\{K(i, k)\}$, що містять інформацію про межі інтервалів і номери класів, зіставлених їм для всіх ознак.

На основі цих масивів сформуємо масив $\{K(q)\}$, що містить номера класів для інтервалів ознак, упорядкованих у порядку зростання номерів ознак і номерів інтервалів значень ознаки.

Даний метод поряд із визначенням меж інтервалів значень ознак дозволяє визначити й оцінки інформативності ознак. Як міру інформативності ознаки стосовно вихідного параметра (міру впливу ознаки на вихідний параметр) будемо використовувати кількість інтервалів, на які розбивається діапазон значень ознаки, таких, що екземпляри, зі значенням ознаки, які попали в один інтервал, відносяться до одного класу, а екземпляри суміжних інтервалів відносяться до різних класів (чим менше кількість інтервалів, тим більше інформативність ознаки і навпаки).

Показники інформативності ознак I_j, таким чином, будемо визначати за формулою: $I_j = \left(\min_{i=1,2,...,N} D_i \right) \Big/ D_j$, $j = 1, 2, ..., N$.

4.7.2 Редукція кількості нечітких термів

Перед синтезом нейро-нечіткої моделі для забезпечення її простоти і високих узагальнюючих й апроксимаційних властивостей зробимо видалення надлишкових блоків визначення приналежності до інтервалів значень ознак (нечітких термів).

Приймемо таке допущення. Інтервали значень ознак x_k та x_q [$A(i, k)$; $B(i, k)$] та [$A(j, q)$; $B(j, q)$] є еквівалентними тим сильніше, чим більше екземплярів, що потрапили в k-ий інтервал значень i-ої ознаки, потраплять у q-ий інтервал значень j-ої ознаки та будуть мати при цьому однакові номери класів.

Коефіцієнт еквівалентності між k-им інтервалом значень i-ої ознаки для s-го екземпляра та q-им інтервалом значень j-ої ознаки для g-го екземпляра визначимо за формулою:

$$n(x_i^s, x_j^g, k, q) = \begin{cases} 0, K(i,k) \neq K(j,q), \\ 0, B(i,k) > x_i^s \text{ або } x_i^s < A(i,k), \\ 0, B(j,q) > x_j^g \text{ або } x_j^g < A(j,q), \\ 1, K(i,k) = K(j,q), A(i,k) \le x_i^s \le B(i,k), A(j,q) \le x_j^g \le B(j,q), \end{cases}$$

$s = 1, 2,..., S$; $g = 1, 2, ..., S$; $i = 1, 2, ..., N$; $j = 1, 2, ..., N$; $k = 1, 2, ..., k_i$; $q = 1, 2, ..., k_j$.

Кількість екземплярів з однаковими номерами класів, що потрапили одночасно до k-го інтервалу значень i-ої ознаки та до q-го інтервалу значень j-ої ознаки визначимо за формулою:

$$N(i, k, j, q) = \sum_{\substack{s=1 \\ g \neq s}}^{S} \sum_{g=1,}^{S} n(x_i^s, x_j^g, k, q),$$

$i = 1, 2, ..., N$; $j = 1, 2, ..., N$; $k = 1, 2, ..., k_i$; $q = 1, 2, ..., k_j$.

Нехай $N_{i,k}$ – кількість екземплярів навчаючої вибірки, що потрапили до k-го інтервалу значень i-ої ознаки.

Коефіцієнт взаємної еквівалентності між k-им інтервалом значень i-ої ознаки та q-им інтервалом значень j-ої ознаки $e_{i,k,j,q}$ визначимо за формулою:

$$e_{i,k,j,q} = \min\left\{\frac{N(i,k,j,q)}{N_{i,k}}, \frac{N(i,k,j,q)}{N_{j,q}}\right\} = \frac{N(i,k,j,q)}{\min\{N_{i,k}, N_{j,q}\}},$$

$i = 1,2,...,N$; $j = 1, 2, ..., N$; $k = 1, 2, ..., k_i$; $q = 1, 2, ..., k_j$.

Коефіцієнт взаємної еквівалентності між i-ою та j-ою ознаками для всіх екземплярів вибірки визначимо за формулою:

$$e_{i,j} = \frac{\sum_{k=1}^{k_i} \sum_{q=1}^{k_j} e_{i,k,j,q}}{\max\{k_i, k_j\}}, \; i = 1, 2, ..., N; \; j = 1, 2, ..., N.$$

Редукцію кількості ознак і блоків визначення приналежності значень ознак до інтервалів для нейро-нечіткої моделі будемо здійснювати шляхом виконання послідовності кроків 1–7.

Крок 1. Ініціалізація. Задати $x = \{x^s{}_i\}$ та $y = \{y^s\}$, $s = 1, 2, ..., S$; $i = 1, 2, ..., N$.

Крок 2. Обчислити характеристики навчаючої вибірки.

Крок 2.1 Знайти: $A(i, k)$, $B(i, k)$, $K(i, k)$, I_i, $N_{i,k}$, k_i.

Крок 2.2 Визначити: $N(i, k, j, q)$, $e_{i,k,j,q}$, $e_{i,j}$.

Крок 3. Ранжирувати ознаки в порядку убування I_i. Установити: $i = N$.

Крок 4. Якщо $i > 1$, тоді виконувати кроки 4.1 та 4.2.

Крок 4.1 Для $\forall j, j \neq i, j = 1,2,...,(i-1)$: якщо $e_{i,j} = 1$, тоді: видалити x_i, установити $N = N$–1.

Крок 4.2 Установити: $i = i$+1. Перейти до кроку 4.

Крок 5. Установити: $i = N$.

Крок 6. Якщо $i \geq 1$, тоді виконати кроки 6.1 та 6.2.

Крок 6.1 Установити: $k = k_i$.

Крок 6.2 Якщо $k \geq 1$, тоді виконати кроки 6.2.1–6.2.3.

Крок 6.2.1 Розрахувати:

$$c = \sum_{j=1}^{N-1} \sum_{q=1}^{k_j} e_{i,k,j,q}, e_{i,k,j,q} = 1.$$

Крок 6.2.2 Якщо $c \geq 1$, тоді: видалити k-ий інтервал i-ої ознаки, встановити: $k_i = k_i - 1$.

Крок 6.2.3 Установити: $k = k - 1$. Перейти до кроку 6.2.

Крок 7. Зупинення.

Після скорочення кількості ознак і блоків визначення приналежності значень ознак до інтервалів сформуємо набір правил виду:

$$\text{Якщо } A(i, k) \leq x^s{}_i \leq B(i, k), \text{ то } y^s{}_i = K(i, k),$$

де $s = 1, 2, ..., S$; $i = 1, 2, ..., N$; $k = 1, 2, ..., k_i$, $y^s{}_i$ – номер класу, до якого належить s-ий екземпляр за i-ою ознакою.

4.7.3 Об’єднання суміжних термів по ознаках

При побудові нейро-нечітких моделей на основі термів, сформованих за допомогою методу 4.7.1, можлива ситуація, коли деякі ознаки будуть містити інтервали значень, до яких потрапляє відносно невелика кількість екземплярів одного класу, а в суміжні до них інтервали буде потрапляти істотно більша кількість екземплярів іншого класу. Ця ситуація найчастіше може пояснюватися тим, що класи містять взаємопроникнення або в даних присутні викиди (нетипові чи помилкові значення).

Для підвищення узагальнюючих й апроксимаційних властивостей, а також спрощення та збільшення швидкості роботи нейро-нечітких моделей є доцільним здійснити виключення інтервалів з невеликим числом екземплярів і виконати об’єднання суміжних з ними інтервалів шляхом виконання таких кроків.

Крок 1. Ініціалізація. Задати S – кількість екземплярів навчаючої вибірки, N – кількість ознак, що характеризують екземпляри навчаючої вибірки, масиви для меж і номерів класів інтервалів значень ознак $\{A(j, k)\}$, $\{B(j, k)\}$, $\{K(j, k)\}$, де j – номер ознаки, $j = 1, 2, \ldots, N$, k – номер інтервалу значень j-ої

ознаки, $k = 1, 2, \ldots, k_j$; k_j – кількість інтервалів, на які розбивається діапазон значень j-ої ознаки. Задати граничне значення α.

Крок 2. Для всіх інтервалів всіх ознак знайти $S_{j,k}$ – кількість екземплярів у k-му інтервалі значень j-ої ознаки, $k = 1, 2, \ldots, k_j$, $j = 1, 2, \ldots, N$. Для кожної ознаки визначити середню кількість екземплярів, що потрапляють в інтервал її значень:

$$\overline{S}_j = \frac{1}{k_j} \sum_{k=1}^{k_j} S_{j,k},\ j = 1, 2, \ldots, N.$$

Крок 3. Для кожного інтервалу значень кожної ознаки визначити коефіцієнти достовірності номера класу:

$$\mu_{j,k}^{K} = \min\left\{1; \frac{S_{j,k}}{\overline{S}_j}\right\},\ k = 1, 2, \ldots, k_j, j = 1, 2, \ldots, N.$$

Крок 4. Виконати видалення суміжних інтервалів.

Крок 4.1 Прийняти: $j = 1$.

Крок 4.2 Якщо $j > N$, тоді перейти до кроку 5, у протилежному випадку – перейти до кроку 4.3.

Крок 4.3 Прийняти: $k = k_j - 1$.

Крок 4.4 Якщо $k > 1$, тоді перейти до кроку 4.8, у протилежному випадку – перейти до кроку 4.5.

Крок 4.5 Якщо $K(j, k-1) = K(j, k+1) \neq K(j, k)$ та $\alpha < S_{j,k-1} + S_{j,k+1} - S_{j,k}$, тоді перейти до кроку 4.6, у протилежному випадку – перейти до кроку 4.7.

Крок 4.6. Прийняти: $B(j, k-1) = B(j, k+1)$, $\mu_{j,k}^{K} = \mu_{j,k-1}^{K} + \mu_{j,k+1}^{K} - \mu_{j,k}^{K}$. Видалити інтервал $k+1$, видалити інтервал k. Прийняти: $k_j = k_j - 2$, $k = k - 1$.

Крок 4.7. Прийняти: $k = k - 1$. Перейти до кроку 4.4.

Крок 4.8. Прийняти: $j = j + 1$. Перейти до кроку 4.2.

Крок 5. Зупинення.

Умову видалення інтервалу на кроці 4.5 можна замінити на більш гнучку: якщо $K(j, k-1) = K(j, k+1) \neq K(j, k)$ та $S_{j,k-1} - S_{j,k} > \alpha$ та $S_{j,k+1} - S_{j,k} > \alpha$, тоді ...

Запропоновані раніше умови видалення інтервалу враховували тільки кількість екземплярів, що потрапили в суміжні інтервали, але ігнорували розмір інтервалів. Для підвищення точності при видаленні й об'єднанні інтервалів доцільно виходити з певної величини, що характеризує щільність інтервалів, тобто відношення кількості екземплярів, що потрапили в інтервал, до його довжини.

Умову видалення інтервалу з грубим обліком щільності інтервалів $\rho_{j,k}$ можна записати у такий спосіб: якщо $K(j, k-1) = K(j, k+1) \neq K(j, k)$ та $\rho_{j,k-1} + \rho_{j,k+1} - \rho_{j,k} > \alpha$, тоді …, де $\rho_{j,k} = \dfrac{S_{j,k}}{B(j,k) - A(j,k)}$.

Визначимо функцію, що враховує щільність екземплярів для *k*-го інтервалу *j*-ої ознаки:

$$\rho_{j,k}(x_j) = S_{j,k} e^{-\frac{(x_j - \bar{x}_{j,k})^2}{2\sigma_{j,k}^2}},$$

$$\bar{x}_{j,k} = \frac{1}{S_{j,k}} \sum_{s=1}^{S} x_j^s,\ A(j,k) \le x_j^s \le B(j,k),$$

$$\sigma_{j,k}^2 = \frac{1}{S_{j,k} - 1} \sum_{s=1}^{S} \left(x_j^s - \bar{x}_j\right)^2,\ A(j,k) \le x_j^s \le B(j,k).$$

Тоді умову видалення інтервалу з урахуванням щільності інтервалів можна представити в такий спосіб: якщо $K(j, k–1) = K(j, k+1) \neq K(j, k)$ та

$$\int_{A(j,k-1)}^{B(j,k+1)} \rho_{j,k-1}(x_j)dx_j + \int_{A(j,k-1)}^{B(j,k+1)} \rho_{j,k+1}(x_j)dx_j - \int_{A(j,k-1)}^{B(j,k+1)} \rho_{j,k}(x_j)dx_j > \alpha \text{ , тоді …}$$

Оскільки $\int_a^b \rho_{j,k}(x_j)dx_j = S_{j,k}\left(e^{-\frac{(b-\bar{x}_{j,k})^2}{2\sigma_{j,k}^2}} - e^{-\frac{(a-\bar{x}_{j,k})^2}{2\sigma_{j,k}^2}}\right)$,

дану умову можна подати в такий спосіб: якщо $K(j, k–1) = K(j, k+1) \neq K(j, k)$ та

$$S_{j,k-1}\left(e^{-\frac{(B(j,k+1)-\bar{x}_{j,k-1})^2}{2\sigma_{j,k-1}^2}} - e^{-\frac{(A(j,k-1)-\bar{x}_{j,k-1})^2}{2\sigma_{j,k-1}^2}}\right) + S_{j,k+1}\left(e^{-\frac{(B(j,k+1)-\bar{x}_{j,k+1})^2}{2\sigma_{j,k+1}^2}} - e^{-\frac{(A(j,k-1)-\bar{x}_{j,k+1})^2}{2\sigma_{j,k+1}^2}}\right) -$$

$$- S_{j,k}\left(e^{-\frac{(B(j,k+1)-\bar{x}_{j,k})^2}{2\sigma_{j,k}^2}} - e^{-\frac{(A(j,k-1)-\bar{x}_{j,k})^2}{2\sigma_{j,k}^2}}\right) > \alpha, \text{ тоді …}$$

4.7.4 Синтез тришарових розпізнаючих нейро-нечітких моделей

Визначивши характеристики екземплярів навчаючої вибірки і виконавши редукцію нечітких термів, задамо функції приналежності для інтервалів значень ознак $\mu_{i,k}(x_i)$, де *i* – номер ознаки, *k* – номер інтервалу значень *i*-ої ознаки.

Як функції приналежності можна використовувати трапецієподібні функції:

$$\mu_{i,k}(x_i)=\begin{cases}0,\ x_i \le 0{,}5(A(i,k)+B(i,k-1)),\\ \dfrac{x_i-0{,}5(A(i,k)+B(i,k-1))}{0{,}5(A(i,k)-B(i,k-1))},\ 0{,}5(A(i,k)+B(i,k-1)) \le x_i < A(i,k),\\ 1,\ A(i,k) \le x_i \le B(i,k),\\ \dfrac{0{,}5(A(i,k+1)+B(i,k))-x_i}{0{,}5(A(i,k+1)-B(i,k))},\ B(i,k) \le x_i < 0{,}5(B(i,k)+A(i,k+1)),\\ 0,\ 0{,}5(B(i,k)+A(i,k+1)) \le x_i,\end{cases}$$

або П-образні функції: $\mu_{i,k}(x_i)=\mu_{i,k\,S}(x_i)\mu_{i,k\,Z}(x_i)$, де $\mu_{i,k\,S}(x_i)-$ *S*-образна функція, а $\mu_{i,k\,Z}(x_i)$ – *Z*-образна функція:

$$\mu_{i,k\,S}(x_i)=\begin{cases}0,\ x_i < 0{,}5(A(i,k)+B(i,k-1)),\\ \dfrac{1}{2}+\dfrac{1}{2}\cos\left(\dfrac{x_i-A(i,k)}{0{,}5(A(i,k)-B(i,k-1))}\pi\right),\ 0{,}5(A(i,k)+B(i,k-1)) \le x_i \le A(i,k),\\ 1,\ x_i > A(i,k);\end{cases}$$

$$\mu_{i,k\,Z}(x_i)=\begin{cases}1,\ x_i < B(i,k),\\ \dfrac{1}{2}+\dfrac{1}{2}\cos\left(\dfrac{x_i-B(i,k)}{0{,}5(A(i,k+1)-B(i,k))}\pi\right),\ B(i,k) \le x_i \le 0{,}5(B(i,k)+A(i,k+1)),\\ 0,\ x_i > 0{,}5(B(i,k)+A(i,k+1)).\end{cases}$$

Далі задамо спосіб знаходження приналежностей $\mu^0\left(x^s\right)$ та $\mu^1\left(x^s\right)$ розпізнаваного екземпляра x^s до класів 0 та 1, відповідно:

$$\mu^0\left(x^s\right)=\max\mu_{i,k}(x_i),K(i,k)=0;\ \mu^1\left(x^s\right)=\max\mu_{i,k}(x_i),K(i,k)=1,$$

$$i = 1, 2, ...,N;\ k = 1, 2, ...,k_i.$$

Після чого визначимо спосіб дефазифікації:

$$y^s=\begin{cases}1,\mu^1(x^s)>\mu^0(x^s),\\ 0,\mu^1(x^s)\le\mu^0(x^s).\end{cases}$$

Запропонований метод дозволить синтезувати розпізнаючі моделі на основі тришарової нейро-нечіткої мережі, схему якої зображено на рис. 4.21.

На входи мережі надходять значення ознак розпізнаваного екземпляра. Вузли першого шару мережі визначають приналежності розпізнаваного екземпляра до інтервалів значень ознак. Вузли другого шару мережі визначають приналежності розпізнаваного екземпляра до класів. Єдиний вузол третього шару здійснює дефазифікацію.

Нейрони нейро-нечіткої мережі, синтезованої на основі запропонованого методу, будуть мати функції постсинаптичного потенціалу і функції активації, що задаються формулами:

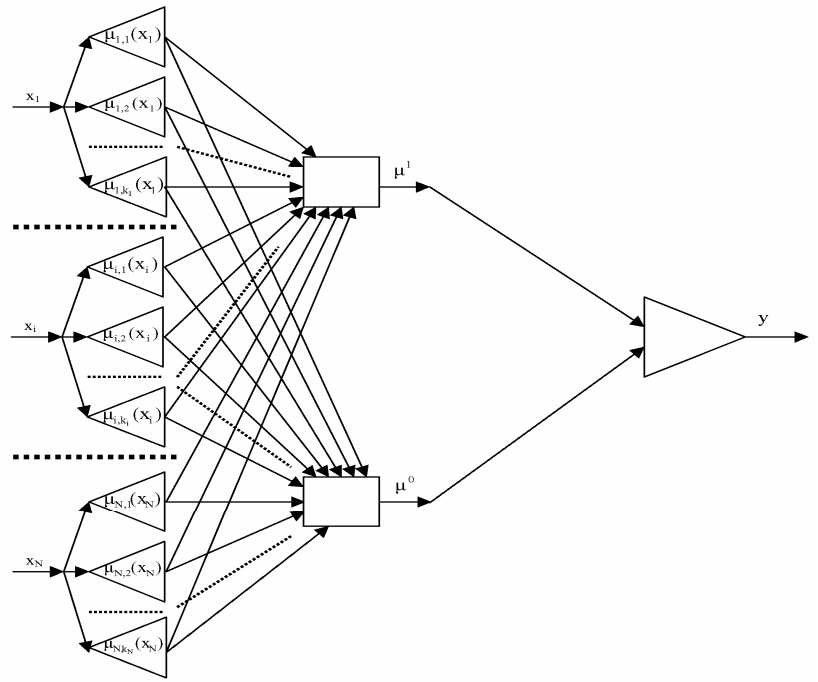


Рисунок 4.21 – Схема нейро-нечіткої мережі

$$\phi^{(3,1)}\left(w^{(3,1)}, x^{(3,1)}\right) = \sum_{j=1}^{2} w_j^{(3,1)} x_j^{(3,1)} + w_0^{(3,1)},$$

$$\psi^{(3,1)}(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0, \end{cases}$$

$$\varphi_j^{(2,i)}\left(w^{(2,i)}, x^{(2,i)}\right) = \min_j\left(w_j^{(2,i)}, x_j^{(2,i)}\right), \ \psi^{(2,i)}(x) = \max_j \varphi_j^{(2,i)}\left(\mathrm{w}_j^{(2,i)}, \mathrm{x}_j^{(2,i)}\right), \ i = 1, 2,$$

де $\psi^{(\eta,i)}(x)$ – функція активації i-го нейрона η-го шару мережі, $\varphi_j^{(\eta,i)}\left(w^{(\eta,i)}, x^{(\eta,i)}\right)$ – функція постсинаптичного потенціалу j-го входу i-го нейрона η-го шару мережі, $w^{(\eta,i)}, x^{(\eta,i)}$ – набори вагових коефіцієнтів та вхідних значень i-го нейрона η-го шару мережі, відповідно.

Вагові коефіцієнти нейронів $w_j^{(\eta,i)}$, де j – номер входу, i – номер нейрона, η – номер шару, будуть визначатися за формулою:

$$w_j^{(\eta,i)} = \begin{cases} 0,\ \eta = 2,\ i = 1,\ K(p,q) = 0,\ j = z(p,q),\ p = 1, 2, ..., N,\ q = 1, 2, ..., k_p, \\ 0,\ \eta = 2,\ i = 2,\ K(p,q) = 1,\ j = z(p,q),\ p = 1, 2, ..., N,\ q = 1, 2, ..., k_p, \\ 1,\ \eta = 2,\ i = 1,\ K(p,q) = 1,\ j = z(p,q),\ p = 1, 2, ..., N,\ q = 1, 2, ..., k_p, \\ 1,\ \eta = 2,\ i = 2,\ K(p,q) = 0,\ j = z(p,q),\ p = 1, 2, ..., N,\ q = 1, 2, ..., k_p, \\ 0,\ \eta = 2,\ i = 1, 2,\ j = 0, \\ 0,\ \eta = 3,\ i = 1,\ j = 0, \\ 1,\ \eta = 3,\ i = 1,\ j = 1, \\ -1,\ \eta = 3,\ i = 1,\ j = 2, \end{cases}$$

де $z(p,q) = q + \sum_{v=1}^{P-1} k_v$.

Запропонований метод синтезу нейро-нечітких моделей налагоджує параметри функцій приналежності неітеративно в процесі синтезу нейро-нечіткої моделі на основі попередньо визначених параметрів інтервалів значень ознак, на відміну від традиційного підходу, коли параметри функцій приналежності налагоджуються шляхом ітеративної оптимізації.

4.7.5 Синтез чотиришарових розпізнаючих нейро-нечітких моделей з урахуванням інформативності ознак

Метод синтезу чотиришарових нейро-нечітких моделей з використанням характеристик навчаючої вибірки й оцінок інформативності ознак подамо як неітеративну послідовність кроків 1–7.

Крок 1. Ініціалізація. Задати навчаючу вибірку $x = \{x^s_j\}$, $j = 1, 2, ..., N$, $s = 1, 2, ..., S$, та значення цільової ознаки $y^* = \{y^{s*}\}$, зіставлені екземплярам навчаючої вибірки.

Крок 2. Знайти $A(i, k)$, $B(i, k)$, $K(i, k)$, D_j, $K(j)$, використовуючи метод 4.7.1. Для кожної q-ої ознаки визначити k_q – кількість інтервалів, на які розбивається її діапазон значень.

Крок 3. Використовуючи метод 4.7.1 визначити показники інформативності ознак I_j, $j = 1, 2, ..., N$.

Крок 4. Задати функції приналежності для інтервалів значень ознак $\mu_{i,k(x_i)}$, де i – номер ознаки, k – номер інтервалу значень i-ої ознаки. Як функції приналежності, подібно до методу 4.7.3, пропонується використовувати трапецієподібні функції або П-образні функції.

Крок 5. Сформувати правила, що поєднують приналежності до інтервалів значень ознак у приналежності до класів одномірних нечітких класифікацій за відповідними ознаками:

$$\mu_i{}^0 = \min\ \mu_{i,k(x_i)},\ K(i, k) = 0,\ k = 1, 2, ..., k_i,\ i = 1, 2, ..., N;$$

$$\mu_i^1 = \min\ \mu_{i,k}(x_i),\ K(i,k) = 1,\ k = 1, 2, ..., k_i,\ i = 1, 2, ..., N.$$

Крок 6. Сформувати правила, що поєднують приналежності до класів одномірних класифікацій за ознаками:

$$\mu^0(y) = \max\ \mu_i^0,\ \mu^1(y) = \max\ \mu_i^1,$$
$$i = 1, 2, ..., N.$$

Крок 7. Задати правило для визначення чіткого результату класифікації:

$$y = \begin{cases} 0,\ \mu^0(y) \geq \mu^1(y), \\ 1,\ \mu^0(y) \geq \mu^1(y). \end{cases}$$

Запропонований метод дозволить синтезувати нейро-нечітку мережу, вагові коефіцієнти нейронів якої $w_j^{(\eta,i)}$, де j – номер входу, i – номер нейрона, η – номер шару, будуть визначатися за формулою:

$$w_j^{(\eta,i)} = \begin{cases} 0, \eta = 2, i = 1, 3, ..., (2N-1), z(j-1) < j < (1+z(j)), K(j) = 0,\ j = 1, 2, ..., z(N), \\ 0, \eta = 2, i = 2, 4, ..., 2N, z(j-1) < j < (1+z(j)), K(j) = 1,\ j = 1, 2, ..., z(N), \\ 1, \eta = 2, i = 1, 3, ..., (2N-1),\ j \leq z(j-1) \text{ або } j \geq (1+z(j)),\ j = 1, 2, ..., z(N), \\ 1, \eta = 2, i = 2, 4, ..., 2N,\ j \leq z(j-1) \text{ або } j \geq (1+z(j)),\ j = 1, 2, ..., z(N), \\ 1, \eta = 2, i = 1, 3, ..., (2N-1), z(j-1) < j < (1+z(j)), K(j) \neq 0,\ j = 1, 2, ..., z(N), \\ 1, \eta = 2, i = 2, 4, ..., 2N, z(j-1) < j < (1+z(j)), K(j) \neq 1,\ j = 1, 2, ..., z(N), \\ 0, \eta = 3, i = 1,\ j = 2, 4, ..., 2N, \\ 0, \eta = 3, i = 2,\ j = 1, 3, ..., (2N-1), \\ 1, \eta = 3, i = 2,\ j = 2, 4, ..., 2N, \\ 1, \eta = 3, i = 1,\ j = 1, 3, ..., (2N-1), \\ 0, \eta = 4, i = 1,\ j = 0, \\ 1, \eta = 4, i = 1,\ j = 1, \\ -1, \eta = 4, i = 1,\ j = 2, \end{cases}$$

де $z(p) = \sum_{q=1}^{P} k_q$.

Функції постсинаптичного потенціалу, а також функції активації нейронів мережі будуть задаватися формулами:

$$\varphi^{(4,1)}\left(w^{(4,1)}, x^{(4,1)}\right) = \sum_{j=1}^{2} w_j^{(4,1)} x_j^{(4,1)} + w_0^{(4,1)},\quad \psi^{(4,1)}(x) = \begin{cases} 0, x < 0, \\ 1, x \geq 0, \end{cases}$$

$$\varphi^{(3,i)}\left(w^{(3,i)}, x^{(3,i)}\right) = \min_{j=1,2,...,2N}\left(w_j^{(3,i)}, x_j^{(3,i)}\right),\ \psi^{(3,i)}(x) = \max_{j=1,2,...,2N} \varphi^{(3,i)}\left(w_j^{(3,i)}, x_j^{(3,i)}\right),\ i = 1, 2,$$

$$\varphi^{(2,i)}\left(w^{(2,i)}, x^{(2,i)}\right) = \max_j\left(w_j^{(2,i)}, x_j^{(2,i)}\right),\ \psi^{(2,i)}(x) = \min_j \varphi^{(2,i)}\left(w_j^{(2,i)}, x_j^{(2,i)}\right),\ i = 1, 2, ..., 2N,$$

де $\varphi^{(\eta,i)}\left(w^{(\eta,i)}, x^{(\eta,i)}\right)$ – функція постсинаптичного потенціалу i-го нейрона η-го шару мережі, $\psi^{(\eta,i)}(x)$ – функція активації i-го нейрона η-го шару мережі,

$w^{(\eta,i)}, x^{(\eta,i)}$ – набори вагових коефіцієнтів і вхідних значень i-го нейрона η-го шару мережі, відповідно.

Схему нейро-нечіткої мережі, синтезованої на основі запропонованого методу, зображено на рис. 4.22.

Як входи мережі використовуються ознаки розпізнаваного екземпляра x_j. Перший шар мережі складають функції приналежності розпізнаваного екземпляра до інтервалів значень ознак. Вузли другого шару поєднують приналежності до інтервалів значень ознак у приналежності до класів одномірних нечітких класифікацій за відповідними ознаками. Вузли третього шару мережі поєднують приналежності до класів одномірних класифікацій за ознаками. Єдиний вузол четвертого шару здійснює дефазифікацію.

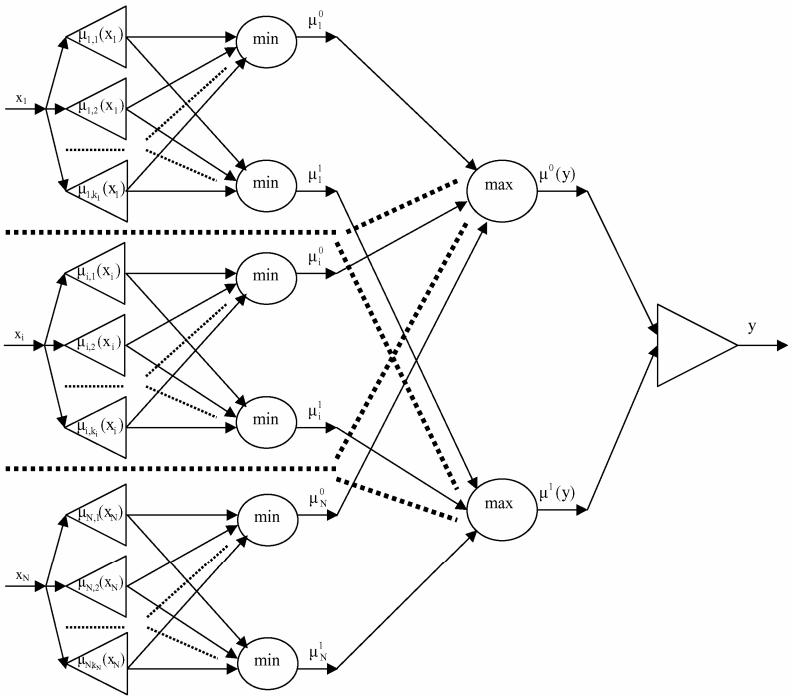


Рисунок 4.22 – Схема нейро-нечіткої мережі

Зазначимо, що в запропонованих методі синтезу і моделі нейро-нечіткої мережі параметри функцій приналежності налагоджуються неітеративно в процесі синтезу нейро-нечіткої моделі на основі попередньо визначених параметрів інтер-

валів значень ознак, на відміну від традиційного підходу, коли параметри функцій приналежності налагоджуються шляхом ітеративної оптимізації.

При визначенні вагових коефіцієнтів нейро-нечіткої мережі, наведеної вище, не використовувалася інформація про інформативність ознак. Однак її можна ввести в мережу за допомогою перевизначення значень ваг третього шару:

$$w_j^{(3,i)} = \begin{cases} I_j,\ i = 2,\ j = 2,\ 4,...,2N, \\ I_j,\ i = 1,\ j = 1,\ 3\ ,...,(2N-1). \end{cases}$$

Оцінки інформативності ознак I_j можна використовувати як в абсолютному виді, так і в пронормованому: $I_{j\,\text{норм.}} = \dfrac{I_j - \min\limits_j (I_j)}{\max\limits_j (I_j) - \min\limits_j (I_j)}$.

4.7.6 Синтез ієрархічних логічно прозорих нейро-нечітких мереж

У задачах технічної та біомедичної діагностики, а також при побудові систем підтримки прийняття рішень в економіці й управлінні виникає необхідність розуміння того, яким чином приймається те чи інше рішення.

Розглянуті раніше класичні моделі нейро-нечітких мереж характеризуються високою складністю для аналізу і сприйняття людиною, оскільки нейроелементи схованих шарів цих мереж містять велику кількість входів, а кількість зв'язків між нейронами також є великою. Тому виникає необхідність у розробці методу, що дозволяє синтезувати нейро-нечіткі мережі, що мають властивість логічної прозорості, тобто зручні для аналізу і сприйняття людиною.

Логічну прозорість нейро-нечітких мереж можливо забезпечити шляхом побудови ієрархії й узагальнення правил прийняття рішень.

Для синтезу ієрархічної нейро-нечіткої моделі на основі вихідної навчаючої вибірки необхідно для кожного класу (або інтервалу значень цільової змінної – для задач з дійсним виходом) сформувати набір правил, до умов і висновків яких входять номера інтервалів значень ознак. Це дозволить сформувати нечіткі терми і базу знань для побудови нейро-нечіткої мережі.

Для отриманого набору правил кожного класу варто сформувати ієрархію прийняття рішення про віднесення розпізнаваного екземпляра до даного класу. Це пропонується здійснювати шляхом групування й об'єднання правил, що відрізняються тільки значенням терму, окремо для кожної ознаки в порядку збільшення інформативності ознак.

Метод синтезу ієрархічних логічно прозорих нейро-нечітких мереж, що реалізує дані розуміння, полягає у виконанні послідовності таких кроків.

Крок 1. Ініціалізація. Задати навчаючу вибірку $<x, y>$, $x = \{x^s\}$, $x^s = \{x^s_j\}$, $y = \{y^s\}$, $s = 1, 2, \ldots, S$, $j = 1, 2, \ldots, N$, де N – кількість ознак, S – кількість екземплярів навчаючої вибірки, x^s_j – значення j-ої ознаки s-го екземпляра, y^s – номер класу, зіставлений s-му екземпляру навчаючої вибірки. Для задач з дійсним

виходом номер класу визначається як номер інтервалу значень вихідної змінної, до якого відноситься *s*-ий екземпляр.

Крок 2. Визначити значення характеристик навчаючої вибірки і сформувати структури даних.

Крок 2.1 На основі навчаючої вибірки за допомогою методу 4.7.1 знайти $\{A(j, k)\}$, $\{B(j, k)\}$, $\{K(j, k)\}$, $\{I_j\}$. де k – номер інтервалу значень j-ої ознаки, $k = 1, 2, \ldots, k_j$; k_j – кількість інтервалів, на які розбивається діапазон значень j-ої ознаки.

Крок 2.2 Визначити загальну кількість інтервалів для всіх ознак навчаючої вибірки: $Z = \sum_{j=1}^{N} k_j$.

Крок 2.3 Сформувати масив вузлів $\{u(p)\}$, $p = 1, 2, ..., Z$, яким послідовно зіставити інтервали значень ознак у порядку збільшення номера ознаки і номера інтервалу значень ознаки. Додати вузол $u(Z+1)$ для об'єднання нечітких термів вихідної змінної. Установити загальну кількість вузлів: $z = Z+1$.

Крок 2.4 Сформувати матрицю зв'язків між вузлами $\{v(i, j)\}$, де $v(i, j) = 0$, якщо зв'язок між i-им та j-им вузлами відсутній, $v(i, j) = 1$, якщо є зв'язок, спрямований від i-го вузла до j-го вузла. Установити: $v(i,j) = 0$, i, $j = 1, 2, ..., z$.

Крок 2.5 Сформувати масив типів вузлів $\{t(j)\}$, де $t(j) = 0$, якщо j-ий вузол має тип «ТА»; $t(j) = 1$, якщо j-ий вузол має тип «АБО»; $t(j) = –1$, якщо j-ий вузол є вхідним нечітким термом або вихідним вузлом мережі. Задати типи вузлів: $t(j)=–1$, $j = 1, 2, ..., z$.

Крок 2.6 Сформувати масив номерів рівнів ієрархії для вузлів $\{h\{j\}\}$, де $h(j)$ – номер рівня ієрархії для j-го вузла. Задати рівні ієрархії для вузлів: $h(j) = 1$, $j = 1, 2, \ldots, z–1$; $h(z) = 0$.

Крок 2.7 Сформувати нечіткі терми – задати функції приналежності для інтервалів значень ознак $\mu_{i,k}(x_i)$, де i – номер ознаки, k – номер інтервалу значень i-ої ознаки. Як функції приналежності, подібно до методу 4.7.4, пропонується використовувати трапецієподібні функції або П-образні функції.

Крок 3. Установити поточний номер класу $q = 1$.

Крок 4. Якщо $q > K$, де K – кількість класів, на які поділяються екземпляри навчаючої вибірки, тоді перейти до кроку 16.

Крок 5. На основі навчаючої вибірки і параметрів, визначених за допомогою методу 4.7.1, сформувати матрицю правил r. Установити номер поточного рівня ієрархії: $h = 2$.

Крок 5.1 Установити: $s = 1$, $s_t = 1$.

Крок 5.2 Якщо $s > S$, тоді перейти до кроку 6.

Крок 5.3 Якщо $y^s = q$, тоді для $\forall j = 1, 2, \ldots, N$, занести у комірку $r(s_t, j)$ номер інтервалу, у який потрапило значення j-ої ознаки s-го екземпляра вибірки, установити: $s_t = s_t+1$. Номера інтервалів значень ознак записуються в суцільній нумерації по всіх ознаках у порядку зростання номера ознаки і номера інтервалу значень усередині ознаки.

Крок 5.4. Установити: $s = s + 1$. Перейти до кроку 5.2.

Крок 6. Для кожного стовпця матриці r знайти $n(j)$ – кількість інтервалів j-ої ознаки, що належать до q-го класу, $j = 1, 2, \ldots, N$. Упорядкувати стовпці матриці r у порядку зростання $n(j)$.

Крок 7. Установити: $j = N$, $s_t = s_t - 1$.

Крок 8. Якщо $j < 2$, тоді перейти до кроку 12.

Крок 9. Для j-ої ознаки знайти в r та узагальнити правила з однаковими частинами лівіше j-го стовпця.

Крок 9.1 Установити: $s = 1$.

Крок 9.2 Якщо $s > s_t$, тоді перейти до кроку 10.

Крок 9.3 Для інтервалу $r(s, j)$ знайти всі інтервали тієї ж j-ої ознаки, що мають однакові з ним ліві частини правил – рядків матриці r. Занести номера вузлів для цих інтервалів у вектор $\aleph = \{\aleph_i\}$, де $\aleph_i$ – i-ий елемент вектора $\aleph$.

Крок 9.4 Визначити ℓ – довжину вектора $\aleph$. Якщо $\ell > 0$, тоді: додати новий вузол $u(z+1)$; прийняти: $z = z + 1$, $h(z) = h + 1$; установити для вузла z тип «АБО»: $t(z) = 1$; додати зв'язок: $v(\aleph_i, z) = 1$; для $p = 1, 2, \ldots, s_t$: якщо $r(p, j) \in \aleph$, тоді установити: $r(p, j) = z$.

Крок 9.5 Залишити в матриці r з кожної групи однакових правил тільки одне правило. Скорегувати відповідним чином s_t.

Крок 9.6 Установити: $s = s + 1$. Перейти до кроку 9.2.

Крок 10. Для $s = 1, 2, \ldots, s_t$: додати новий вузол $u(z + 1)$; прийняти: $z = z + 1$, $h(z) = h + 2$; установити для вузла z тип «ТА»: $t(z) = 0$; додати зв'язок: $v(r(s, j), z) = 1$, $v(r(s, j - 1), z) = 1$; установити: $r(s, j - 1) = z$.

Крок 11. Установити: $h = h + 2$, $j = j - 1$. Перейти до кроку 8.

Крок 12. Залишити в матриці r з кожної групи однакових правил тільки одне правило. Скорегувати відповідним чином s_t.

Крок 13. Якщо $s_t > 1$, тоді: додати новий вузол $u(z + 1)$; прийняти: $z = z + 1$, $h(z) = h + 1$; установити для вузла z тип «АБО»: $t(z) = 0$; додати зв'язок: $v(s, z) = 1$; для $s = 1, 2, \ldots, s_t$; установити: $r(1, 1) = z$; видалити з r усі правила, крім першого.

Крок 14. Установити зв'язок: $v(r(1,1), Z + q) = 1$.

Крок 15. Установити: $q = q + 1$. Перейти до кроку 4.

Крок 16. Установити: $h(Z+1) = 1 + \max\limits_{i=1,2,\ldots,z} h(i)$.

Для кожного рівня ієрархії $\eta = 1, 2, \ldots, h(Z+1)$, визначити N_η – кількість вузлів, що знаходяться на рівні ієрархії η.

Крок 17. Зупинення.

Схему ієрархічної логічно прозорої нейро-нечіткої мережі, синтезованої на основі запропонованого методу, подано на рис. 4.23.

На входи мережі подаються чіткі значення ознак розпізнаваного екземпляра. Перший шар мережі містить вузли, що визначають приналежності значень ознак до нечітких термів – інтервалів значень ознак. Сховані шари мережі

реалізують ієрархічне виведення рішення про приналежність до нечітких термів вихідної змінної. Парні сховані шари (на рис. 4.23 позначені окружностями) містять нейрони типу «АБО», непарні сховані шари (на рис. 4.23 позначені квадратами) містять нейрони типу «ТА». Останній шар мережі містить один нейрон, що виконує об'єднання нечітких значень приналежностей розпізнаваного екземпляра до класів і приведення результату до чіткого значення.

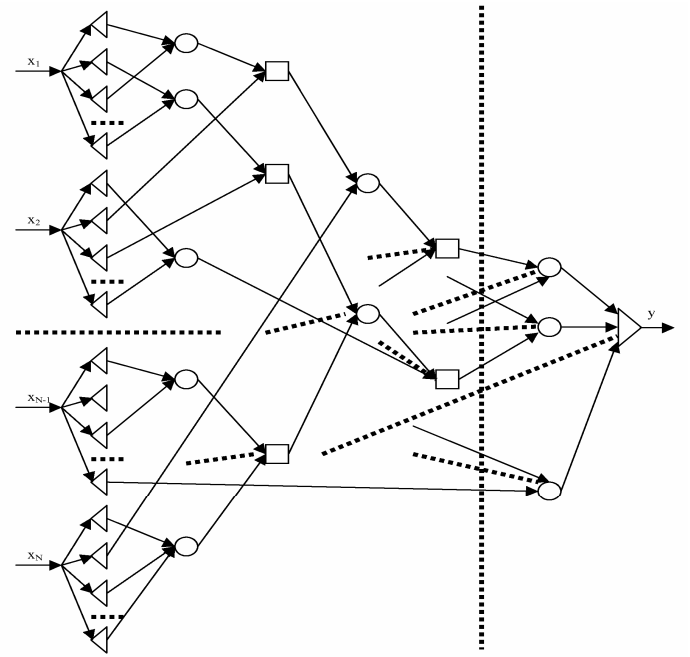


Рисунок 4.23 – Схема ієрархічної нейро-нечіткої мережі

Дискримінантні функції нейронів будуть визначатися в такий спосіб:

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \min\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 2, 4, \ldots, h(Z+1)-1;$$

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \max\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 3, 5, \ldots, h(Z+1)-2;$$

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \frac{\sum_{j=1}^{K} w_j^{(\eta,i)} x_j^{(\eta,i)}}{\sum_{j=1}^{K} x_j^{(\eta,i)}},\ \eta = h(Z+1),\ i = 1.$$

Функції активації нейронів мережі будуть визначатися за формулами:

$$\psi^{(\eta,i)}(x) = \max_j \{\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)})\},\ \eta = 2, 4, \ldots, h(Z+1)-1;$$

$$\psi^{(\eta,i)}(x) = \min_j \{\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)})\},\ \eta = 3, 5, \ldots, h(Z+1)-2;$$

для задач класифікації: $\psi^{(\eta,i)}(x) = \text{round}(x),\ \eta = h(Z+1),\ i = 1;$

для задач оцінювання: $\psi^{(\eta,i)}(x) = x,\ \eta = h(Z+1),\ i = 1.$

Вагові коефіцієнти нейронів $w_j^{(\eta,i)}$, де j – номер входу, i – номер нейрона, η – номер шару, будуть визначатися за формулою:

$$w_j^{(\eta,i)} = \begin{cases} j, \eta = h(Z+1), i = 1,\ j = 1, 2, ..., K, \\ v(j,i), \eta = 2, 4, \ldots, h(Z+1)-1, i = 1, 2, ..., N_\eta,\ j = 1, 2, ..., N_{\eta-1}, \\ 1 - v(j,i), \eta = 3, 5, \ldots, h(Z+1)-2, i = 1, 2, ..., N_\eta,\ j = 1, 2, ..., N_{\eta-1}. \end{cases}$$

Мережа, синтезована на основі наведених вище формул, буде реалізувати виведення на основі ієрархічної max-min композиції.

Дану мережу можна також модифікувати для реалізації виведення на основі ієрархічної max-max композиції. Парні сховані шари (на рис. 4.23 позначені окружностями) будуть, як і в попередньому випадку, містити нейрони типу «АБО», а непарні сховані шари (на рис. 4.23 позначені квадратами) будуть містити також нейрони типу «АБО».

Параметри нейронів першого й останнього шарів будуть визначатися також як і раніше. Для нейронів схованих шарів дискримінантні функції будуть визначатися в такий спосіб:

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \min \{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 2, 3, \ldots, h(Z+1)-1,$$

а функції активації будуть визначатися за формулами:

$$\psi^{(\eta,i)}(x) = \max_j \{\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)})\},\ \eta = 2, 3, \ldots, h(Z+1)-1.$$

Вагові коефіцієнти нейронів схованих шарів будуть визначатися за формулою: $w_j^{(\eta,i)} = v(j,i), \eta = 2, 3, \ldots, h(Z+1)-1, \text{i} = 1, 2, ..., N_\eta,\ j = 1, 2, ..., N_{\eta-1}.$

За допомогою запропонованого методу також можна синтезувати мережу, що реалізує виведення на основі ієрархічної max-prod композиції. Парні сховані шари (на рис. 4.23 позначені окружностями) будуть, як і в попередньому випадку, містити нейрони типу «АБО», а непарні сховані шари (на рис. 4.23 позначені квадратами) будуть містити нейрони типу «PROD».

Параметри нейронів першого й останнього шарів будуть визначатися також як і раніше. Для нейронів схованих шарів дискримінантні функції будуть визначатися в такий спосіб:

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \min \{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 2, 4, \ldots, h(Z+1)-1;$$

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \max \{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 3, 5, \ldots, h(Z+1)-2;$$

а функції активації будуть визначатися за формулами:

$$\psi^{(\eta,i)}(x) = \max_j \{\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)})\},\ \eta = 2, 4, \ldots, h(Z+1)-1;$$

$$\psi^{(\eta,i)}(x) = \prod_j \varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}),\ \eta = 3, 5, \ldots, h(Z+1)-2.$$

Вагові коефіцієнти нейронів схованих шарів будуть визначатися за формулою:

$$w_j^{(\eta,i)} = \begin{cases} v(j,i), \eta = 2,4,\ldots,h(Z+1)\text{-}1, i = 1,2,...,N_\eta,\ j = 1,2,...,N_{\eta\text{-}1}, \\ 1 - v(j,i), \eta = 3,5,\ldots,h(Z+1)\text{-}2, i = 1,2,...,N_\eta,\ j = 1,2,...,N_{\eta\text{-}1}. \end{cases}$$

За допомогою запропонованого методу також можна синтезувати мережу, що реалізує виведення на основі ієрархічної max-average композиції. Парні сховані шари (на рис. 4.23 позначені окружностями) будуть, як і в попередньому випадку, містити нейрони типу «АБО», а непарні сховані шари (на рис. 4.23 позначені квадратами) будуть містити нейрони типу «AVERAGE».

Параметри нейронів першого й останнього шарів будуть визначатися також як і раніше. Для нейронів схованих шарів дискримінантні функції будуть визначатися в такий спосіб:

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \min\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 2, 3, \ldots, h(Z+1)-1,$$

а функції активації будуть визначатися за формулами:

$$\psi^{(\eta,i)}(x) = \max_j \{\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)})\},\ \eta = 2, 4, \ldots, h(Z+1)-1;$$

$$\psi^{(\eta,i)}(x) = \frac{1}{N_{\eta\text{-}1}} \sum_{j=1}^{N_{\eta\text{-}1}} \varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}),\ \eta = 3, 5, \ldots, h(Z+1)-2.$$

Вагові коефіцієнти нейронів схованих шарів будуть визначатися за формулою: $w_j^{(\eta,i)} = v(j,i), \eta = 2,3,\ldots,h(Z+1)\text{-}1, i = 1,2,...,N_\eta,\ j = 1,2,...,N_{\eta\text{-}1}.$

За допомогою запропонованого методу також можна синтезувати мережу, що реалізує виведення на основі ієрархічної average-max композиції. Парні сховані шари (на рис. 4.23 позначені окружностями) будуть містити нейрони типу «AVERAGE», а непарні сховані шари (на рис. 4.23 позначені квадратами) будуть містити нейрони типу «АБО».

Параметри нейронів першого й останнього шарів будуть визначатися також як і раніше. Для нейронів схованих шарів дискримінантні функції будуть визначатися в такий спосіб:

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \min\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 2, 3, \ldots, h(Z+1)-1,$$

а функції активації будуть визначатися за формулами:

$$\psi^{(\eta,i)}(x) = \frac{1}{N_{\eta\text{-}1}} \sum_{j=1}^{N_{\eta\text{-}1}} \varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}),\ \eta = 2, 4, \ldots, h(Z+1)-1;$$

$$\psi^{(\eta,i)}(x) = \max_j \{\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)})\},\ \eta = 3, 5, \ldots, h(Z+1)-2.$$

Вагові коефіцієнти нейронів схованих шарів будуть визначатися за формулою:

$$w_j^{(\eta,i)} = v(j,i), \eta = 2,3,\ldots, h(Z+1)\text{-}1, i = 1,2,...,N_\eta, j = 1,2,...,N_{\eta\text{-}1}.$$

За допомогою запропонованого методу також можна синтезувати мережу, що реалізує виведення на основі ієрархічної average-min композиції. Парні сховані шари (на рис. 4.23 позначені окружностями) будуть містити нейрони типу «AVERAGE», а непарні сховані шари (на рис. 4.23 позначені квадратами) будуть містити нейрони типу «TA».

Параметри нейронів першого й останнього шарів будуть визначатися також як і раніше. Для нейронів схованих шарів дискримінантні функції будуть визначатися в такий спосіб:

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \min\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 2, 4, \ldots, h(Z+1)-1,$$

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \max\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 3, 5, \ldots, h(Z+1)-2;$$

а функції активації будуть визначатися за формулами:

$$\psi^{(\eta,i)}(x) = \frac{1}{N_{\eta\text{-}1}} \sum_{j=1}^{N_{\eta\text{-}1}} \varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}),\ \eta = 2, 4, \ldots, h(Z+1)-1;$$

$$\psi^{(\eta,i)}(x) = \min_j\{\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)})\},\ \eta = 3, 5, \ldots, h(Z+1)-2.$$

Вагові коефіцієнти нейронів схованих шарів будуть визначатися як:

$$w_j^{(\eta,i)} = \begin{cases} v(j,i), \eta = 2,4,\ldots, h(Z+1)\text{-}1, i = 1,2,...,N_\eta, j = 1,2,...,N_{\eta\text{-}1}, \\ 1 - v(j,i), \eta = 3,5,\ldots, h(Z+1)\text{-}2, i = 1,2,...,N_\eta, j = 1,2,...,N_{\eta\text{-}1}. \end{cases}$$

За допомогою запропонованого методу також можна синтезувати мережу, що реалізує виведення на основі ієрархічної average-prod композиції. Парні сховані шари (на рис. 4.23 позначені окружностями) будуть містити нейрони типу «AVERAGE», а непарні сховані шари (на рис. 4.23 позначені квадратами) будуть містити нейрони типу «PROD».

Параметри нейронів першого й останнього шарів будуть визначатися також як і раніше. Для нейронів схованих шарів дискримінантні функції будуть визначатися в такий спосіб:

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \min\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 2, 4, \ldots, h(Z+1)-1,$$

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \max\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 3, 5, \ldots, h(Z+1)-2;$$

а функції активації будуть визначатися за формулами:

$$\psi^{(\eta,i)}(x) = \frac{1}{N_{\eta\text{-}1}} \sum_{j=1}^{N_{\eta\text{-}1}} \varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}),\ \eta = 2, 4, \ldots, h(Z+1)-1;$$

$$\psi^{(\eta,i)}(x) = \prod_j \varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}),\ \eta = 3, 5, \ldots, h(Z+1)-2.$$

Вагові коефіцієнти нейронів схованих шарів будуть визначатися за формулою:

$$w_j^{(\eta,i)} = \begin{cases} v(j,i), \eta = 2,4,\ldots, h(Z+1)-1, i = 1,2,...,N_\eta, j = 1,2,...,N_{\eta-1}, \\ 1-v(j,i), \eta = 3,5,\ldots, h(Z+1)-2, i = 1,2,...,N_\eta, j = 1,2,...,N_{\eta-1}. \end{cases}$$

За допомогою запропонованого методу також можна синтезувати мережу, що реалізує виведення на основі ієрархічної average-average композиції. Парні (на рис. 4.23 позначені окружностями) і непарні (на рис. 4.23 позначені квадратами) сховані шари будуть містити нейрони типу «AVERAGE».

Параметри нейронів першого й останнього шарів будуть визначатися також як і раніше. Для нейронів схованих шарів дискримінантні функції будуть визначатися в такий спосіб:

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \min\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\}, \ \eta = 2, 3, \ldots, h(Z+1)-1,$$

а функції активації будуть визначатися за формулами:

$$\psi^{(\eta,i)}(x) = \frac{1}{N_{\eta-1}} \sum_{j=1}^{N_{\eta-1}} \varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}), \ \eta = 2, 3, \ldots, h(Z+1)-1.$$

Вагові коефіцієнти нейронів схованих шарів будуть визначатися за формулою: $w_j^{(\eta,i)} = v(j,i), \eta = 2,3,\ldots, h(Z+1)-1, i = 1,2,...,N_\eta, j = 1,2,...,N_{\eta-1}.$

За допомогою запропонованого методу також можна синтезувати мережу, що реалізує виведення на основі ієрархічної sum-prod композиції. Парні сховані шари (на рис. 4.23 позначені окружностями) будуть містити нейрони типу «SUM», а непарні сховані шари (на рис. 4.23 позначені квадратами) будуть містити нейрони типу «PROD».

Параметри нейронів першого й останнього шарів будуть визначатися також як і раніше. Для нейронів схованих шарів дискримінантні функції будуть визначатися в такий спосіб:

$$\varphi^{(\eta,i)}(x) = \sum_{j=1}^{N_{\eta-1}} w_j^{(\eta,i)} x_j^{(\eta,i)}, \ \eta = 2, 4, \ldots, h(Z+1)-1,$$

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \max\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\}, \ \eta = 3, 5, \ldots, h(Z+1)-2;$$

а функції активації будуть визначатися за формулами:

$$\psi^{(\eta,i)}(x) = \frac{1}{1+e^{-x}}, \ \eta = 2, 4, \ldots, h(Z+1)-1,$$

$$\psi^{(\eta,i)}(x) = \prod_j \varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}), \ \eta = 3, 5, \ldots, h(Z+1)-2.$$

Ваги нейронів схованих шарів будуть визначатися за формулою:

$$w_j^{(\eta,i)} = \begin{cases} v(j,i), \eta = 2,4,\ldots, h(Z+1)-1, i = 1,2,...,N_\eta, j = 1,2,...,N_{\eta-1}, \\ 1-v(j,i), \eta = 3,5,\ldots, h(Z+1)-2, i = 1,2,...,N_\eta, j = 1,2,...,N_{\eta-1}. \end{cases}$$

За допомогою запропонованого методу також можна синтезувати мережу, що реалізує виведення на основі ієрархічної max-sum композиції. Парні сховані шари (на рис. 4.23 позначені окружностями) будуть містити нейрони типу «АБО», а непарні сховані шари (на рис. 4.23 позначені квадратами) будуть містити нейрони типу «SUM».

Параметри нейронів першого й останнього шарів будуть визначатися також як і раніше. Для нейронів схованих шарів дискримінантні функції будуть визначатися в такий спосіб:

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \min\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 2, 4, \ldots, h(Z+1)-1;$$

$$\varphi^{(\eta,i)}(x) = \sum_{j=1}^{N_{\eta-1}} w_j^{(\eta,i)} x_j^{(\eta,i)},\ \eta = 3, 5, \ldots, h(Z+1)-2;$$

а функції активації будуть визначатися за формулами:

$$\psi^{(\eta,i)}(x) = \max_j\{\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)})\},\ \eta = 2, 4, \ldots, h(Z+1)-1,$$

$$\psi^{(\eta,i)}(x) = \frac{1}{1+e^{-x}},\ \eta = 3, 5, \ldots, h(Z+1)-2.$$

Вагові коефіцієнти нейронів схованих шарів будуть визначатися за формулою:

$$w_j^{(\eta,i)} = v(j,i), \eta = 2,3,\ldots,h(Z+1)\text{-}1, i = 1,2,\ldots,N_\eta, j = 1,2,\ldots,N_{\eta\text{-}1}.$$

За допомогою запропонованого методу також можна синтезувати мережу, що реалізує виведення на основі ієрархічної sum-max композиції. Парні сховані шари (на рис. 4.23 позначені окружностями) будуть містити нейрони типу «SUM», а непарні сховані шари (на рис. 4.23 позначені квадратами) будуть містити нейрони типу «АБО».

Параметри нейронів першого й останнього шарів будуть визначатися також як і раніше. Для нейронів схованих шарів дискримінантні функції будуть визначатися в такий спосіб:

$$\varphi^{(\eta,i)}(x) = \sum_{j=1}^{N_{\eta-1}} w_j^{(\eta,i)} x_j^{(\eta,i)},\ \eta = 2, 4, \ldots, h(Z+1)-1;$$

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \min\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 3, 5, \ldots, h(Z+1)-2;$$

а функції активації будуть визначатися за формулами:

$$\psi^{(\eta,i)}(x) = \frac{1}{1+e^{-x}},\ \eta = 2, 4, \ldots, h(Z+1)-1,$$

$$\psi^{(\eta,i)}(x) = \max_j\{\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)})\},\ \eta = 3, 5, \ldots, h(Z+1)-2.$$

Вагові коефіцієнти нейронів схованих шарів будуть визначатися за формулою: $w_j^{(\eta,i)} = v(j,i), \eta = 2,3,\ldots,h(Z+1)\text{-}1, i = 1,2,\ldots,N_\eta, j = 1,2,\ldots,N_{\eta\text{-}1}.$

За допомогою запропонованого методу також можна синтезувати мережа, що реалізує виведення на основі ієрархічної sum-min композиції. Парні сховані шари (на рис. 4.23 позначені окружностями) будуть містити нейрони типу «SUM», а непарні сховані шари (на рис. 4.23 позначені квадратами) будуть містити нейрони типу «ТА».

Параметри нейронів першого й останнього шарів будуть визначатися також як і раніше. Для нейронів схованих шарів дискримінантні функції будуть визначатися в такий спосіб:

$$\varphi^{(\eta,i)}(\mathrm{x}) = \sum_{j=1}^{N_{\eta-1}} \mathrm{w}_{\mathrm{j}}^{(\eta,\mathrm{i})} \mathrm{x}_{\mathrm{j}}^{(\eta,\mathrm{i})}, \quad \eta = 2, 4, \ldots, h(Z+1)-1,$$

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \max\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\}, \ \eta = 3, 5, \ldots, h(Z+1)-2,$$

а функції активації будуть визначатися за формулами:

$$\psi^{(\eta,i)}(\mathrm{x}) = \frac{1}{1+e^{-x}}, \ \eta = 2, 4, \ldots, h(Z+1)-1,$$

$$\psi^{(\eta,i)}(x) = \min_j\{\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)})\}, \ \eta = 3, 5, \ldots, h(Z+1)-2.$$

Вагові коефіцієнти нейронів схованих шарів будуть визначатися за формулою:

$$w_j^{(\eta,i)} = \begin{cases} v(j,i), \eta = 2,4,\ldots,h(Z+1)-1, i = 1,2,\ldots,N_\eta, \ j = 1,2,\ldots,N_{\eta-1}, \\ 1-v(j,i), \eta = 3,5,\ldots,h(Z+1)-2, i = 1,2,\ldots,N_\eta, \ j = 1,2,\ldots,N_{\eta-1}. \end{cases}$$

Запропонований метод синтезу нейро-нечітких мереж узагальнює класичні методи нечіткого виведення і дозволяє будувати логічно прозорі нейро-нечіткі мережі, що є зручними не тільки для аналізу і сприйняття, але також мають добрі узагальнюючі властивості та легко реалізуються апаратно за рахунок спрощення структури обробляючих елементів і скорочення кількості зв'язків між нейроелементами.

4.7.7 Синтез нейро-нечітких мереж з групуванням ознак

При побудові розпізнаючих моделей, особливо в задачах класифікації зображень і діагностики, не рідко виникає ситуація, коли окремо деякі ознаки слабо впливають на вихідну ознаку, а спільно впливають на неї сильно. Це може пояснюватися тим, що окремі ознаки, що доступні нами для спостереження і вимірювання, є непрямими проявами більш важливих ознак, недоступних для спостереження і (або) вимірювання, називаних *факторами*.

Спроба побудови моделі за непрямими ознаками може бути успішною, але така модель найчастіше вкрай незручна для подальшого аналізу, оскільки адекватно не відображає реальні залежності між ознаками або виділяє фактори неявно.

Тому є доцільним перед побудовою моделі розбивати ознаки на групи у залежності від значення певної міри їхнього взаємного зв'язку, здійснювати узагальнення ознак кожної групи шляхом розрахунку значення деякої згортки і далі будувати модель залежності вихідної ознаки від значень згорток для кожної групи ознак. Це дозволить не тільки спростити подальший аналіз моделі, але й істотно скоротити розмірність навчаючої вибірки, а, отже, спростити розпізнаючу модель.

Групування ознак пропонується здійснювати шляхом виконання кроків 1–10.

Крок 1. Задати навчаючу вибірку $<x, y>$, $x = \{x^s\}$, $x^s = \{x^s_j\}$, $y = \{y^s\}$, де x^s – s-ий екземпляр вибірки, x^s_j – значення j-ої ознаки s-го екземпляра вибірки, y^s – значення цільової ознаки s-го екземпляра вибірки.

Крок 2. Визначити значення показника інформативності для кожної ознаки I_j, $j = 1, 2, \ldots, N$. Для цього можна використовувати метод 4.7.1, модуль коефіцієнта парної кореляції, коефіцієнт кореляції Фехнера, а також коефіцієнт кореляції знаків.

Крок 3. Для $i, j = 1, 2, \ldots, N$, $i \neq j$, знайти відстані між ознаками в просторі екземплярів навчаючої вибірки $d(x_j, x_i)$:

$$d(x_j, x_i) = \sqrt{\sum_{s=1}^{S} (x_j^s - x_i^s)^2}.$$

Крок 4. За допомогою методу 4.7.2 знайти коефіцієнти еквівалентності ознак e_{ji}, $i, j = 1, 2, \ldots, N$, $i \neq j$. Прийняти: $e_{ji} = e_{ji}^{-1}$.

Крок 5. Установити номер поточної групи: $g = 0$.

Крок 6. Якщо $\exists x_j$: $\forall I_j \neq -1$ та $\exists d(x_j, x_i) \neq -1$, $i, j = 1, 2, \ldots, N$, $i \neq j$, тоді перейти до кроку 7, у протилежному випадку – перейти до кроку 10.

Крок 7. Знайти ознаку x_i: $I_i = \max I_j$, $j = 1, 2, \ldots, N$.

Крок 8. Установити: $g = g + 1$. Додати нову групу ознак G^g. Включити в групу G^g ознаку x_i. Установити: $I_i = -1$.

Крок 9. Якщо $\exists x_j$: $\forall I_j \neq -1$, $\exists d(x_i, x_j) \neq -1$, $F(x_i, x_j) = 1$, i, $j = 1, 2, \ldots, N$, $i \neq j$, де F – певний булевий оператор, що визначає приналежність ознак до однієї групи, тоді включити в групу G^g ознаку x_j, установити: $I_j = -1$, $d(x_j, x_k) = -1$, $d(x_k, x_j) = -1$, $k = 1, 2, \ldots, N$, перейти до кроку 9, у протилежному випадку – установити: $I_i = -1$, $d(x_i, x_k) = -1$, $d(x_k, x_i) = -1$, $k = 1, 2, \ldots, N$, перейти до кроку 6.

Крок 10. Зупинення.

Оператор F може бути визначений одним з таких способів:

$$F(x_i, x_j) = \begin{cases} 1, d(x_i, x_j) < \alpha\overline{d}, \\ 0, d(x_i, x_j) \geq \alpha\overline{d}; \end{cases} \quad F(x_i, x_j) = \begin{cases} 1, e_{ij} < \beta\overline{e}, \\ 0, e_{ij} \geq \beta\overline{e}; \end{cases}$$

$$F(x_i, x_j) = \begin{cases} 1, \hat{d}(x_i, x_j) < \gamma\overline{\hat{d}}, \\ 0, \hat{d}(x_i, x_j) \geq \gamma\overline{\hat{d}}; \end{cases} \quad F(x_i, x_j) = \begin{cases} 1, d(x_i, x_j) < \alpha\overline{d}, e_{ij} < \beta\overline{e}, \\ 0, d(x_i, x_j) \geq \alpha\overline{d} \text{ або } e_{ij} \geq \beta\overline{e}; \end{cases}$$

$$F(x_i, x_j) = \begin{cases} 1, d(x_i, x_j) < \alpha\overline{d}, \ \hat{d}(x_i, x_j) < \gamma\overline{\hat{d}}, \\ 0, d(x_i, x_j) \geq \alpha\overline{d} \text{ або } \hat{d}(x_i, x_j) \geq \gamma\overline{\hat{d}}; \end{cases} \quad F(x_i, x_j) = \begin{cases} 1, e_{ij} < \beta\overline{e}, \hat{d}(x_i, x_j) < \gamma\overline{\hat{d}}, \\ 0, e_{ij} \geq \beta\overline{e} \text{ або } \hat{d}(x_i, x_j) \geq \gamma\overline{\hat{d}}; \end{cases}$$

$$F(x_i, x_j) = \begin{cases} 1, d(x_i, x_j) < \alpha\overline{d}, e_{ij} < \beta\overline{e}, \hat{d}(x_i, x_j) < \gamma\overline{\hat{d}}, \\ 0, d(x_i, x_j) \geq \alpha\overline{d} \text{ або } e_{ij} \geq \beta\overline{e} \text{ або } \hat{d}(x_i, x_j) \geq \gamma\overline{\hat{d}}. \end{cases}$$

Параметри $\bar{d}, \hat{d}, \bar{\hat{d}}, \bar{e}$ можуть бути визначені за відповідними формулами:

$$\bar{d} = \frac{1}{0{,}5N^2 - N} \sum_{i=1}^{N} \sum_{j=i+1}^{N} d(x_i, x_j),\ \bar{e} = \frac{1}{0{,}5N^2 - N} \sum_{i=1}^{N} \sum_{j=i+1}^{N} e_{ij},$$

$$\bar{\hat{d}} = \frac{1}{0{,}5N^2 - N} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \hat{d}(x_i, x_j),$$

$$\hat{d}(x_i, x_j) = \sqrt{((i-1) \bmod \text{width} - (j-1) \bmod \text{width})^2 + ((i-1) \operatorname{div} \text{width} - (j-1) \operatorname{div} \text{width})^2},$$

де width – ширина зображення, div – операція цілочисленого ділення, mod – операція «залишок від цілочисленого ділення».

Після групування ознак можна побудувати нейро-нечіткі моделі.

У випадку, коли метою аналізу є визначення семантики схованих ознак (факторів, що відповідають групам) доцільно синтезувати мережу, зображену на рис. 4.24. Запропонована мережа може мати альтернативне подання (рис. 4.25), яке доцільно використовувати, якщо метою аналізу є установлення впливу факторів на конкретний клас.

На входи запропонованих мереж будуть подаватися значення відповідних ознак розпізнаваного екземпляра. На першому шарі обох мереж розташовані блоки визначення приналежності розпізнаваного екземпляра до нечітких термів. Нечіткі терми можна сформувати також, як і в методі 4.7.3. Другий шар містить нейрони, що визначають приналежність розпізнаваного екземпляра за групами ознак кожного класу. Кількість нейронів другого шару $N_2 = VQ$, де V – кількість груп ознак, Q – кількість класів. Третій шар мережі містить нейрони, що поєднують приналежності за різними групами ознак у приналежності до класів. Кількість нейронів третього шару $N_3 = Q$.

Єдиний нейрон четвертого шару мережі здійснює об'єднання принадлежностей до класів і виконує дефазифікацію.

Вагові коефіцієнти нейронів $w_j^{(\eta,i)}$, де j – номер входу, i – номер нейрона, η – номер шару, нейро-нечіткої мережі з групуванням ознак будуть визначатися за формулою:

$$w_j^{(\eta,i)} = \begin{cases} j, \eta = 4, i = 1, j = 1, 2, ..., K, \\ 0, \eta = 3, i = 1, 2, ...Q, v = 1, 2, ..., V, j = (v-1)Q + i, \\ 1, \eta = 3, i = 1, 2, ...Q, v = 1, 2, ..., V, j \neq (v-1)Q + i, \\ 1, \eta = 2, x_p \in G^v \text{ й } K(p, g) = q, j = z(p) + g, g = 1, 2, ..., D_p, p = 1, 2, ..., N, \\ \quad i = (v-1)Q + q, q = 1, 2, ..., Q, v = 1, 2, ..., V, \\ 0, \eta = 2, x_p \in G^v \text{ або } K(p, g) = q, j = z(p) + g, g = 1, 2, ..., D_p, p = 1, 2, ..., N, \\ \quad i = (v-1)Q + q, q = 1, 2, ..., Q, v = 1, 2, ..., V, \end{cases}$$

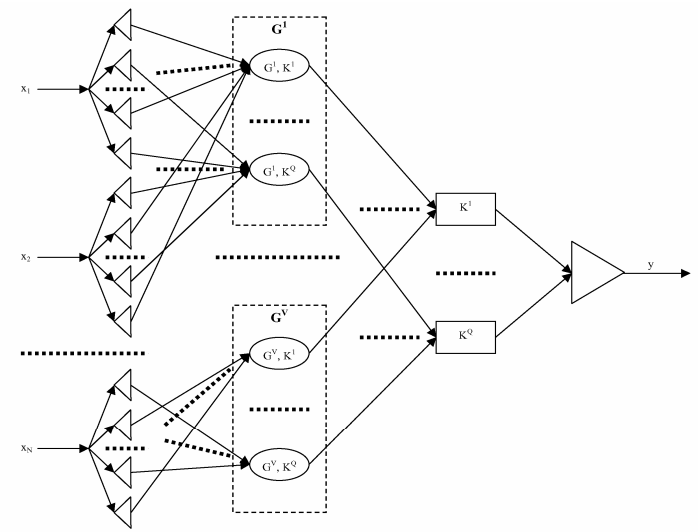


Рисунок 4.24 – Нейро-нечітка мережа з групуванням ознак

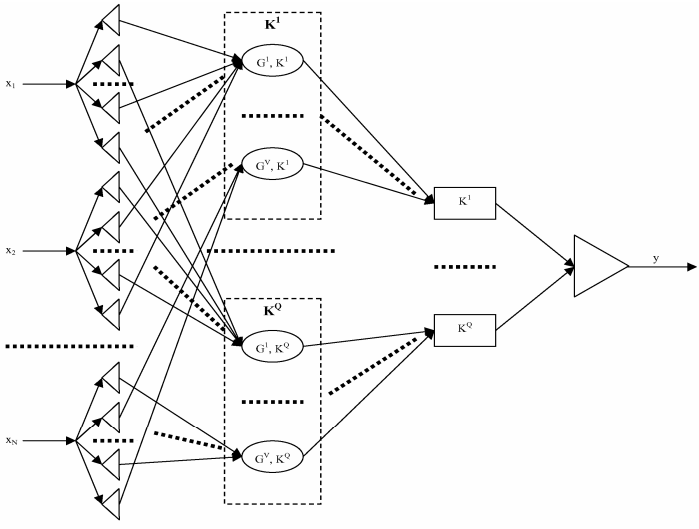


Рисунок 4.25 – Альтернативне подання
нейро-нечіткої мережі з групуванням ознак

$$z(p) = \sum_{r=1}^{p-1} D_r,\ z = \sum_{j=1}^{N} D_j.$$

Дискримінантні функції нейронів обох мереж будуть визначатися як:

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \min\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 2,\ i = 1, 2, \ldots, VQ;$$

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \max\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\},\ \eta = 3,\ i = 1, 2, \ldots, Q;$$

$$\varphi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)}) = \frac{\sum_{j=1}^{K} w_j^{(\eta,i)} x_j^{(\eta,i)}}{\sum_{j=1}^{K} x_j^{(\eta,i)}},\ \eta = 4,\ i = 1.$$

Функції активації нейронів обох мереж будуть визначатися за формулами:

$$\psi^{(\eta,i)}(x) = \max_j \{\phi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)})\},\ \eta = 2,\ i = 1, 2, \ldots, VQ;$$

$$\psi^{(\eta,i)}(x) = \min_j \{\phi^{(\eta,i)}(w_j^{(\eta,i)}, x_j^{(\eta,i)})\},\ \eta = 3,\ i = 1, 2, \ldots, Q;$$

$$\psi^{(\eta,i)}(x) = \text{round}(x),\ \eta = 4,\ i = 1.$$

Для альтернативного подання мережі вагові коефіцієнти нейронів $w_j^{(\eta,i)}$ будуть визначатися за формулою:

$$w_j^{(\eta,i)} = \begin{cases} j, \eta = 4, i = 1,\ j = 1, 2, \ldots, K, \\ 0, \eta = 3, i = 1, 2, \ldots Q, v = 1, 2, \ldots, V,\ j = (i-1)V + v, \\ 1, \eta = 3, i = 1, 2, \ldots Q, v = 1, 2, \ldots, V,\ j \neq (i-1)V + v, \\ 1, \eta = 2, x_p \in G^v \text{ й } K(p,g) = q,\ j = z(p) + g,\ g = 1,2,\ldots, D_p,\ p = 1,2,\ldots, N, \\ \quad i = (q\text{-}1)V + v, v = 1, 2, \ldots, V,\ q = 1, 2, \ldots, Q, \\ 0, \eta = 2, x_p \in G^v \text{ або } K(p,g) = q,\ j = z(p) + g,\ g = 1,2,\ldots, D_p,\ p = 1,2,\ldots, N, \\ \quad i = (q\text{-}1)V + v, v = 1, 2, \ldots, V, q = 1, 2, \ldots, Q. \end{cases}$$

У випадку, якщо всі ознаки, віднесені до групи G^g, є однотипними (наприклад, є відрахунками сигналу чи рівнями інтенсивності точок зображення), їх можна замінити на узагальнену ознаку, що являє собою одну зі згорток:

$$x_i = \sum_{j=1}^{N} x_j, x_j \in G^g,\ \ x_i = \frac{1}{N_G} \sum_{j=1}^{N} x_j, x_j \in G^g,$$

$$x_i = \min_{j=1,2,\ldots,N} x_j, x_j \in G^g,\ x_i = \max_{j=1,2,\ldots,N} x_j, x_j \in G^g.$$

Поділ ознак на групи й об'єднання ознак груп за допомогою згорток дозволить синтезувати нейро-нечіткі мережі, зображені на рис. 4.26.

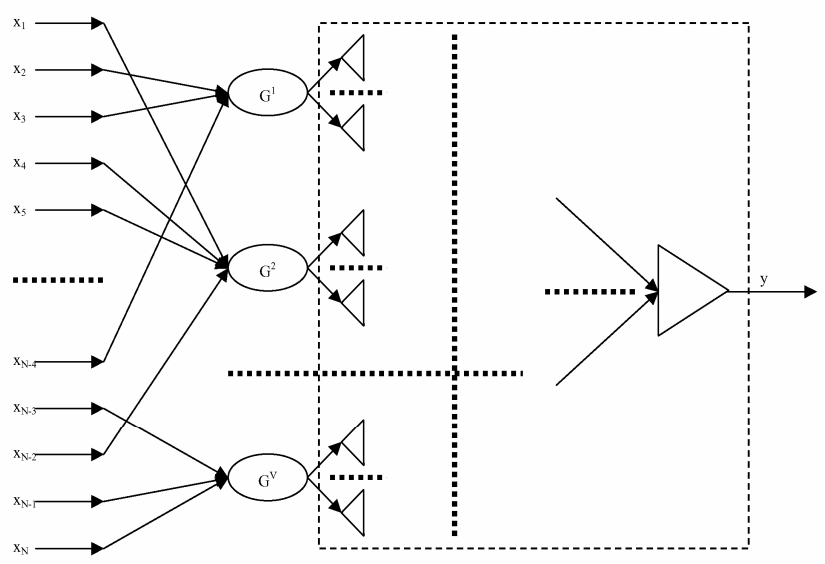


Рисунок 4.26 – Нейро-нечітка мережа з групуванням та згорткою ознак

На входи нейро-нечіткої мережі будуть подаватися значення відповідних ознак розпізнаваного екземпляра.

На першому шарі мережі розташовані нейрони, що здійснюють об'єднання ознак у групи та розраховують значення згорток для кожної групи ознак. Кількість нейронів першого шару $N_1 = V$.

Другий шар мережі містить блоки визначення приналежності розпізнаваного екземпляра до нечітких термів значень згорток. Нечіткі терми для інтервалів значень згорток можна сформувати також, як і в методі 4.7.3, замінивши ознаки на згортки.

Третій і наступний шари мережі реалізують об'єднання приналежностей до нечітких термів у приналежності до класів. Ці шари можуть бути подані розпізнаючими нейро-нечіткими мережами різних типів.

Єдиний нейрон останнього шару мережі здійснює об'єднання приналежностей до класів і виконує дефазифікацію.

Як дискримінантну функцію для сум-згорток нейрони першого шару будуть використовувати:

$$\phi^{(\eta,i)}(w^{(\eta,i)}, x^{(\eta,i)}) = w_0^{(\eta,i)} + \sum_{j=1}^{N_{\eta-1}} w_j^{(\eta,i)} x_j^{(\eta,i)}.$$

Як функцію активації для сум-згорток нейрони першого шару будуть використовувати: $\psi(x) = x$.

Для сум-згорток вагові коефіцієнти нейронів першого шару будуть задаватися формулою:

$$w_j^{(\eta,i)} = \begin{cases} 0, \eta = 1, i = 1, 2, ..., V, \ j = 0, \\ \alpha, x_j \in G^i, \eta = 1, i = 1, 2, ..., V, \ j = 1, 2, ..., N, \\ 0, x_j \notin G^i, \eta = 1, i = 1, 2, ..., V, \ j = 1, 2, ..., N, \end{cases}$$

де α = 1 – для суми значень ознак і $\alpha = N_G^{-1}$ – для середнього арифметичного значень ознак.

Як дискримінантну функцію нейрони першого шару для згортки max будуть використовувати: $\phi^{(\eta,i)}(w^{(\eta,i)}, x^{(\eta,i)}) = \min\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\}$.

Як функцію активації для згортки max нейрони першого шару будуть використовувати: $\psi^{(\eta,i)} = \max\limits_i \{\phi^{(\eta,i)}(w^{(\eta,i)}, x^{(\eta,i)})\}$.

При використанні згортки max вагові коефіцієнти нейронів першого шару будуть задаватися формулою:

$$w_j^{(\eta,i)} = \begin{cases} 1, x_j \in G^i, \eta = 1, i = 1, 2, ..., V, \ j = 1, 2, ..., N, \\ 0, x_j \notin G^i, \eta = 1, i = 1, 2, ..., V, \ j = 1, 2, ..., N. \end{cases}$$

Як дискримінантну функцію нейрони першого шару для згортки min будуть використовувати: $\phi^{(\eta,i)}(w^{(\eta,i)}, x^{(\eta,i)}) = \max\{w_j^{(\eta,i)}, x_j^{(\eta,i)}\}$.

Як функцію активації для згортки min нейрони першого шару будуть використовувати: $\psi^{(\eta,i)} = \min\limits_i \{\phi^{(\eta,i)}(w^{(\eta,i)}, x^{(\eta,i)})\}$.

При використанні згортки min вагові коефіцієнти нейронів першого шару будуть задаватися формулою:

$$w_j^{(\eta,i)} = \begin{cases} 0, x_j \in G^i, \eta = 1, i = 1, 2, ..., V, \ j = 1, 2, ..., N, \\ 1, x_j \notin G^i, \eta = 1, i = 1, 2, ..., V, \ j = 1, 2, ..., N. \end{cases}$$

Параметри нейронів інших шарів мережі будуть визначатися правилами для відповідних типів нейро-нечітких мереж.

При вирішенні задач розпізнавання для випадку трьох і більше класів доцільною може виявитися побудова не єдиної моделі з виходом, що приймає три і більше значення, а синтез окремих моделей для визначення приналежностей до кожного класу. Це дозволить формувати набори груп ознак своїх для кожного класу.

Наприклад, у задачах розпізнавання зображень може виявитися, що одна і та ж ознака може відноситися до різних груп ознак для різних класів. Побудова окремих моделей для кожного класу дозволить зберегти внутрикласове семантичне навантаження ознак, у той час, як побудова єдиної моделі для всіх класів може привести до часткової втрати семантики ознак.

Для побудови окремих моделей для кожного класу необхідно у вихідній вибірці даних замінити значення цільової ознаки на «1», якщо екземпляр належить до класу, для визначення якого будується модель, і на «0» – у протилежному випадку. Після чого можна застосувати для побудови моделі по кожному класу той самий метод, що і для виходу, який приймає багато значень.

4.7.8 Об'єднання нечітких термів у кластери

Нехай ми маємо вибірку даних <*x*, *y*>. Кількість класів (можливих значень цільової змінної) позначимо як *K*. Кожен клас може містити один або декілька кластерів. Необхідно знайти таке розбиття навчаючої вибірки на кластери, що буде у певному розумінні оптимальним.

Крок 1. Ініціалізація. Задати навчаючу вибірку <*x*, *y*>, $x = \{x^s\}$, $x_s = \{x^s_j\}$. $Y = \{y^s\}$, де *j* – номер ознаки (*j* = 1, 2, …, *N*), *N* – кількість ознак, *S* – кількість екземплярів навчаючої вибірки, x^s_j – значення *j*-ої ознаки для *s*-го екземпляра. Прийняти: *Q* = *S*, де *Q* – кількість кластерів.

Крок 2. За допомогою методу 4.7.1 знайти *A*(*j*, *k*), *B*(*j*, *k*), *K*(*j*, *k*), I_j, де *k* – номер інтервалу значень *j*-ої ознаки, $k = 1, 2,\ldots,k_j$; k_j – кількість інтервалів, на які розбивається діапазон значень *j*-ої ознаки. Визначити *n*(*j*, *k*, *p*) – кількість екземплярів навчаючої вибірки, що потрапили в *k*-ий інтервал значень *j*-ої ознаки та належать до *p*-го класу. Сформувати функції приналежності для нечітких термів (інтервалів значень ознак) $\mu_{j,k}$, де *j* – номер ознаки, *k* – номер інтервалу значень *j*-ої ознаки. Як функції приналежності, подібно до методу 4.7.3, пропонується використовувати трапецієподібні функції або П-образні функції.

Крок 3. Для вибірки x сформувати кластери $\{C^q\}$, $C^q_j = k$, де *q* – номер кластера, *j* – номер ознаки *q*-го кластера, *k* – номер інтервалу значень *j*-ої ознаки, у який потрапив екземпляр x^q. Кожному кластеру C^q зіставити номер класу y_q, до якого він належить.

Крок 4. Визначити відстані між кластерами:

$$R(M,L) = R(L,M) = \sum_{j=1}^{N} \left| C_j^M - C_j^L \right| , \; M = 1, 2, \ldots, Q; \; L = 1, 2, \ldots, Q.$$

Крок 5. Для кластерів кожного класу виконувати об'єднання поки не задовольняться критерії закінчення пошуку.

Крок 5.1 Установити: *q* =1.

Крок 5.2 Якщо *q* > *K*, тоді перейти до кроку 6.

Крок 5.3 Установити: $M = 1$.

Крок 5.4 Якщо $M > Q$, тоді перейти до кроку 5.10.

Крок 5.5 Установити: $L = M + 1$.

Крок 5.6 Якщо $L > Q$, тоді перейти до кроку 5.9.

Крок 5.7 Якщо $y_M = y_L = q$ та $R(M, L) \le R(\alpha, \beta)$ та $y_\alpha = y_\beta = q$, $\alpha = 1, \ldots, Q$, $\beta = 1, \ldots, Q$, $\alpha \ne \beta$; $\alpha \ne M$ при $\beta = L$; $\alpha \ne L$ при $\beta = M$; $\beta \ne M$ при $\alpha = L$; $\beta \ne L$ при $\alpha = M$, тоді виконати кроки 5.7.1–5.7.4; у протилежному випадку – перейти до кроку 5.8.

Крок 5.7.1 Об'єднати кластери C^M та C^L:

$$C_j^M = \begin{cases} C_j^M, C_j^M = C_j^L, \\ C_j^M, C_j^M \ne C_j^L, n(j, C_j^M, q) \ge n(j, C_j^L, q), \\ C_j^L, C_j^M \ne C_j^L, n(j, C_j^M, q) < n(j, C_j^L, q). \end{cases}$$

Прийняти для $v = L, \ldots, Q-1$: $C^v = C^{v+1}$, $R(v, u) = R(v+1, u)$, $R(u, v) = R(u, v+1)$, $u = 1, 2, \ldots, Q$. Установити: $Q = Q - 1$. Для $L = 1, 2, \ldots, Q$, знайти:

$$R(M, L) = R(L, M) = \sum_{j=1}^{N} \left| C_j^M - C_j^L \right| \cdot$$

Крок 5.7.2 Для поточного розбиття на кластери знайти чіткий номер класу y^{s*}, $s = 1, 2, \ldots, S$.

Крок 5.7.2.1 Знайти значення функцій приналежності розпізнаваного екземпляра x^s до нечітких термів (інтервалів значень ознак) $\mu_{j,k}(x^s)$.

Крок 5.7.2.2 Знайти значення функцій приналежності розпізнаваного екземпляра x^s до нечітких термів q-го кластера, p-го класу:

$$\mu_{j,k}^{p,q}(x^s) = \begin{cases} \mu_{j,k}(x^s), K(j,k) = y_q = p, \\ 0, \neg(K(j,k) = y_q = p); \end{cases}$$

$p = 1,2, \ldots, K$; $q = 1, 2, \ldots, Q$; $j = 1, 2, \ldots, N$; $k = 1, 2, \ldots, k_j$.

Крок 5.7.2.3 Знайти значення функцій приналежності розпізнаваного екземпляра x^s за j-ою ознакою до q-го кластера, p-го класу:

$$\mu_j^{p,q}(x^s) = \max_{k=1,2,\ldots,k_j} \mu_{j,k}^{p,q}(x^s), \; p = 1, 2, \ldots, K; \; q = 1, 2, \ldots, Q; \; j = 1, 2, \ldots, N.$$

Крок 5.7.2.4 Знайти значення функцій приналежності розпізнаваного екземпляра x^s до q-го кластера, p-го класу:

$$\mu^{p,q}(x^s) = \frac{1}{N} \sum_{j=1}^{N} \mu_j^{p,q}(x^s) \text{ або } \mu^{p,q}(x^s) = \max_{j=1,\ldots,N} \mu_j^{p,q}(x^s),$$

$p = 1, 2, \ldots, K$; $q = 1, 2, \ldots, Q$.

Крок 5.7.2.5 Знайти значення функцій приналежності розпізнаваного екземпляра x^s до p-го класу:

$$\mu_p(x^s) = \max_{q=1,2,\ldots,Q} \mu^{p,q}(x^s) \ , p = 1, 2, \ldots, K.$$

Крок 5.7.2.6 Визначити: $y^{s*} = \arg \max_{p=1,\ldots,K} \mu_p(x^s)$.

Крок 5.7.3 Розрахувати помилку для поточного розбиття екземплярів навчаючої вибірки:

$$E = \sum_{s=1}^{S} \left| y^s - y^{s*} \right| .$$

Крок 5.7.4 Якщо $E \le E_{\max}$, тоді перейти до кроку 5.5, у протилежному випадку – скасувати об'єднання кластерів C^M та C^L, установити: $R(M, L) = R(L, M)$ = RealMax, де RealMax – максимальне для розрядної сітки ЕОМ число.

Крок 5.8 Прийняти: $L = L + 1$. Перейти до кроку 5.6.

Крок 5.9 Прийняти: $M = M + 1$. Перейти до кроку 5.4.

Крок 5.10 Прийняти: $q = q + 1$. Перейти до кроку 5.2

Крок 6. Зупинення.

У результаті виконання кроків 1–6 для навчаючої вибірки отримаємо розбиття на Q кластерів C^q.

4.7.9 Нейро-нечітка кластер-регресійна апроксимація

У випадку, коли вихідна змінна при побудові моделі залежності є дійсною, пропонується використовувати кластер-регресійну апроксимацію, побудову якої можна здійснювати шляхом виконання таких кроків.

Крок 1. Ініціалізація. Задати навчаючу вибірку $<x, y>$, де $x = \{x^s\}$, $y = \{y^s\}$, $x^s = \{x^s_j\}$.

Крок 2. Дискретизувати значення цільової змінної у одним зі способів.

Спосіб 1.

Крок 2.1. Задати L – кількість інтервалів, на які розбивається діапазон значень цільової змінної y. Знайти $\min(y^s)$ та $\max(y^s)$ – мінімальне і максимальне значення цільової змінної у, відповідно.

Крок 2.2. Визначити ширину вікна дискретизації цільової змінної:

$$\Delta y = \frac{\max(y^s) - \min(y^s)}{L}.$$

Крок 2.3. Прийняти: $y^s = y^s \operatorname{div} \Delta y$, $s = 1, 2, \ldots, S$, де a div b – операція цілочисленого ділення a на b.

Спосіб 2.

Крок 2.1. Задати δy – припустиму погрішність. Знайти $\min(y^s)$ та $\max(y^s)$ – мінімальне і максимальне значення цільової змінної y, відповідно.

Крок 2.2 Відсортувати екземпляри навчаючої вибірки в порядку неспадання значень цільової змінної y.

Крок 2.3 Установити: $k = 0$, $s = 2$, $y = \min(y)$, $y^1 = 0$.

Крок 2.4 Якщо $s > S$, тоді перейти до кроку 3.

Крок 2.5 Якщо $|y^s - y_L| \le \delta y$, тоді установити: $y^s = k$, у протилежному випадку – установити: $k = k + 1$, $y^s = k$.

Крок 2.6 Установити: $s = s + 1$. Перейти до кроку 2.4.

Крок 3. Для вибірки з дискретизованою цільовою змінною за допомогою методу 4.7.1 знайти $A(j, k)$, $B(j, k)$, $K(j, k)$.

Крок 4. Видалити малоінформативні ознаки і надлишкові інтервали вхідних змінних. Задати функції приналежності екземплярів до нечітких термів (інтервалів значень ознак) $\mu_{j,k}$, де j – номер ознаки, k – номер інтервалу значень j-ої ознаки. Як функції приналежності, подібно до методу 4.7.3, пропонується використовувати трапецієподібні функції або П-образні функції.

Крок 5. Сформувати кластери – компактні області зосередження екземплярів вибірки з однаковим номером класу шляхом задавання правил виду: (q) Якщо … та $A(j, k) \le x_j \le B(j, k)$ та …, то $K(q) = K(j, k)$, де q – номер правила (кластера), $K(q)$ – номер класу, зіставлений q-му кластеру, $q = 1, 2, \ldots, Q$, Q – кількість кластерів:

$$Q = \frac{1}{2}\sum_{j=1}^{N} k_j \left(\sum_{\substack{p=1,\\ p\ne j}}^{N} k_p \right).$$

Крок 6. Задати функції приналежності екземплярів до кластерів.

$$\mu^q = \max_{\substack{j=1,2,\ldots,N;\\ k=1,2,\ldots,k_i}} w_{j,k}^q \mu_{j,k},\ q = 1, 2, \ldots, Q,\ w_{j,k}^q = \begin{cases} 1, K(q) = K(j,k), \\ 0, K(q) \ne K(j,k). \end{cases}$$

Крок 7. Для екземплярів кожного кластера побудувати часткові моделі залежностей $y_q(x^s)$, використовуючи вихідні недискретизовані значення цільової змінної:

$$y_q(x^s) = \beta_0^q + \sum_{j=1}^{N} \beta_j^q x_j^s \ ,$$

де β_j^q – коефіцієнт багатомірної лінійної регресійної моделі при j-й ознаці. При побудові часткових моделей можливо враховувати інформативність ознак, а також сам процес побудови моделей здійснювати шляхом послідовного нарощування числа ознак, що дозволить спростити часткові моделі [170].

Крок 8. Визначити метод селекції часткової моделі для кластера й одержання чіткого розрахункового значення цільової змінної:

$$y^{s*} = \sum_{q=1}^{Q} \alpha_q \mu^q(x^s) y_q(x^s),$$

де α_q – деякий ваговий коефіцієнт (у найпростішому випадку $\alpha_q = 1$).

Даний метод може мати нейро-нечітку інтерпретацію у вигляді чотиришарової нейро-нечіткої мережі (рис. 4.27).

На входи мережі подаються значення N ознак розпізнаваного екземпляра. Перший шар мережі містить нейрони, що визначають приналежності розпізнаваного екземпляра до нечітких термів (інтервалів значень ознак). Другий шар мережі містить $2Q$ нейронів: перші Q нейронів визначають приналежності розпізнаваного екземпляра до кластерів, другі Q нейронів реалізують часткові моделі $y_q(x^s)$. Третій шар містить Q нейронів, що виконують селекцію часткових моделей для кластерів. Четвертий (вихідний) шар мережі містить один нейрон, що виконує об'єднання часткових моделей і дефазифікацію цільової змінної.

Функції постсинаптичного потенціалу нейронів мережі задаються формулами:

$$\varphi^{(2,i)}(w,x) = \min_j \left(\mathrm{w}_j^{(2,i)}, \mathrm{x}_j^{(2,i)} \right),\ i = 1, 2, \ldots, Q,$$

$$\varphi^{(2,i)}(w^{(2,i)}, x^{(2,i)}) = \sum_{j=1}^{N} w_j^{(2,i)} x_j^{(2,i)} + w_0^{(2,i)},\ i = Q+1, Q+2, \ldots, 2Q,$$

$$\varphi^{(3,i)}(w^{(3,i)}, x^{(3,i)}) = \prod_{j=i,2i} w_j^{(3,i)} x_j^{(3,i)},\ i = 1, 2, \ldots, Q,$$

$$\varphi^{(4,1)}(w^{(4,1)}, x^{(4,1)}) = \sum_{j=1}^{Q} w_j^{(4,1)} x_j^{(4,1)} + w_0^{(4,1)}\ .$$

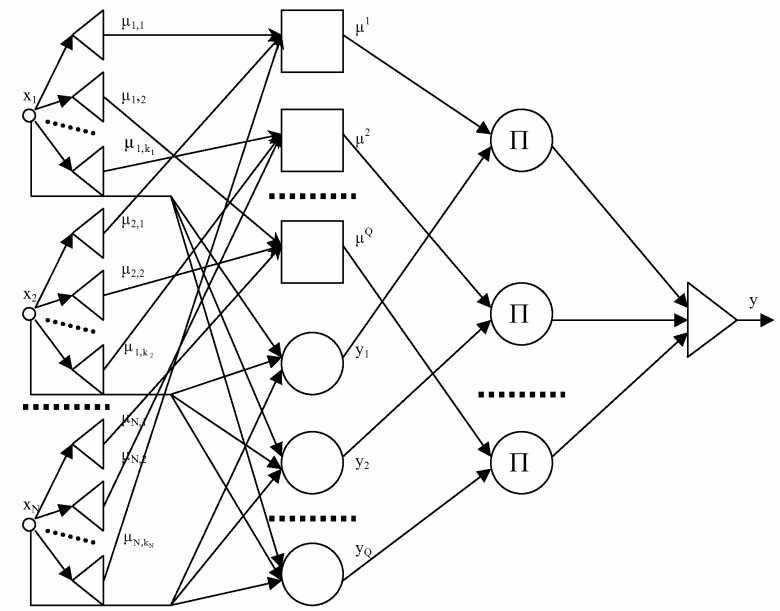


Рисунок 4.27 – Схема нейро-нечіткої мережі

Функції активації нейронів мережі задаються формулами:

$$\psi^{(2,i)}(x) = \max_j \varphi_j^{(2,i)}\left(w_j^{(2,i)}, x_j^{(2,i)}\right),\ i = 1, 2, \ldots, Q;\ \psi^{(2,i)}(x) = x,\ i = Q+1, Q+2, \ldots, 2Q;$$

$$\psi^{(3,i)}(x) = x,\ i = 1, 2, \ldots, Q;\ \psi^{(4,1)}(x) = x.$$

Вагові коефіцієнти нейронів $w_j^{(\eta,i)}$, де j – номер входу, i – номер нейрона, η – номер шару, визначають за формулою:

$$w_j^{(\eta,i)} = \begin{cases} 1, \eta = 2, i = 1, 2, ..., Q,\ j = 1, 2, ..., N, \\ \beta_j^q, \eta = 2, i = Q + q, q = 1, 2, ..., Q,\ j = 0, 1, ..., N, \\ 1, \eta = 3, i = 1, 2, ..., Q,\ j = 1, 2, ..., 2Q, \\ 0, \eta = 4, i = 1,\ j = 0, \\ \alpha_j, \eta = 4, i = 1,\ j = 1, 2, ..., Q. \end{cases}$$

Після синтезу нейро-нечіткої моделі для підвищення точності можна здійснити підбір вагових коефіцієнтів єдиного нейрона останнього шару мережі, використовуючи для цього методи багатовимірної нелінійної безумовної оптимізації [32].

4.7.10 Метод донавчання нейро-нечітких мереж

Нехай ми маємо нейро-нечітку мережу, параметри функцій приналежностей нечітких термів якої A, B, K, D, а також правила сформовані за вибіркою $<x, y>, x = \{x^s\}, x^s = \{x_j^s\}, y = \{y^s\}, s = 1, 2, \ldots, S; j = 1, 2, \ldots, N$. Нехай також є вибірка $< \hat{x}, \hat{y} >,\ \hat{x} = \{\hat{x}^s\},\ \hat{x}^s = \{\hat{x}_j^s\},\ \hat{y} = \{\hat{y}^s\},\ s = 1, 2, \ldots, \hat{S};\ j = 1, 2, \ldots, N$. Необхідно донавчити нейро-нечітку мережу на основі вибірки $< \hat{x}, \hat{y} >$, за можливістю зберігаючи при цьому знання, отримані за вибіркою $<x, y>$.

Для донавчання пропонується використовувати такий метод.

Етап 1. Донавчання нечітких термів.

Крок 1. Ініціалізація. Задати: параметри A, B, K, D, вибірку $< \hat{x}, \hat{y} >$. Прийняти: $\hat{A} = A, \hat{B} = B, \hat{K} = K, \hat{D} = D$. Установити номер поточної ознаки: $j = 1$.

Крок 2. Якщо $j > N$, тоді перейти до кроку 11.

Крок 3. Установити: $k = 1$.

Крок 4. Якщо $k > \hat{D}_j$, тоді перейти до кроку 10.

Крок 5. Установити: $s = 1$.

Крок 6. Якщо $s > \hat{S}$, тоді перейти до кроку 10.

Крок 7. Проаналізувати положення поточного екземпляра відносно поточного інтервалу поточної ознаки.

Крок 7.1 Якщо $\hat{A}(j,k) \le \hat{x}_j^s \le \hat{B}(j,k)$ та $\hat{y}^s = \hat{K}(j,k)$, тоді перейти до кроку 8.

Крок 7.2 Якщо $\hat{A}(j,k) \le \hat{x}_j^s \le \hat{B}(j,k)$ та $\hat{y}^s \ne \hat{K}(j,k)$, тоді додати два нових інтервали: $\hat{D}_j = \hat{D}_j + 2$; $\hat{A}(j,i) = \hat{A}(j,i-2)$, $\hat{B}(j,i) = \hat{B}(j,i-2)$, $\hat{K}(j,i) = \hat{K}(j,i-2)$, $i = \hat{D}_j$, $\hat{D}_j - 1$, …, $k+4$, $k+3$; $\hat{A}(j,k+2) = \hat{x}_j^s + \delta$, $\hat{B}(j,k+2) = \hat{B}(j,k)$, $\hat{K}(j,k+2) = \hat{K}(j,k)$, $\hat{A}(j,k+1) = \hat{x}_j^s$, $\hat{B}(j,k+1) = \hat{x}_j^s$, $\hat{K}(j,k+1) = \hat{y}^s$, $\hat{B}(j,k) = \hat{x}_j^s - \delta$, де δ – деяке невелике значення: $\delta \ge 0$, $\delta << |\hat{B}(j,k) - \hat{A}(j,k)|$; перейти до кроку 8.

Крок 7.3 Якщо $\hat{B}(j,k-1) < \hat{x}_j^s < \hat{A}(j,k)$ та $\hat{y}^s = \hat{K}(j,k)$, тоді прийняти $\hat{A}(j,k) = \hat{x}_j^s$, перейти до кроку 8.

Крок 7.4 Якщо $\hat{B}(j,k-1) < \hat{x}_j^s < \hat{A}(j,k)$ та $\hat{y}^s = \hat{K}(j,k-1)$, тоді прийняти: $\hat{B}(j,k-1) = \hat{x}_j^s$, перейти до кроку 8.

Крок 7.5 Якщо $\hat{B}(j,k) < \hat{x}_j^s < \hat{A}(j,k+1)$ та $\hat{K}(j,k) \ne \hat{y}^s \ne \hat{K}(j,k+1)$, тоді додати новий інтервал: $\hat{D}_j = \hat{D}_j + 1$; $\hat{A}(j,i) = \hat{A}(j,i-1)$, $\hat{B}(j,i) = \hat{B}(j,i-1)$, $\hat{K}(j,i) = \hat{K}(j,i-1)$, $i = \hat{D}_j$, $\hat{D}_j - 1$, …, $k+4$, $k+2$ $\hat{A}(j,k+1) = \hat{x}_j^s$, $\hat{B}(j,k+1) = \hat{x}_j^s$, перейти до кроку 8.

Крок 8. Установити: $s = s + 1$. Перейти до кроку 6.

Крок 9. Установити: $k = k + 1$. Перейти до кроку 4.

Крок 10. Установити: $j = j + 1$. Перейти до кроку 2.

Крок 11. Зупинення.

Етап 2. Довизначення правил. Відповідно до принципів побудови архітектури нейро-нечіткої мережі внести до неї довизначені терми, зберегти попередні правила, сформувати і додати правила для нових екземплярів.

4.7.11 Виділення нечітких термів на основі інтервалів значень ознак, що перетинаються для різних класів

У раніше розглянутих методах 4.7.4–4.7.9 нечіткі терми формувалися на основі інтервалів значень ознак, виділених за допомогою методу 4.7.1.

Метод 4.7.1 припускав, що інтервали різних класів за ознаками не перетинаються, однак для задач із взаємним проникненням і складним поділом класів це призводить до виділення великої кількості інтервалів в одномірних

проекціях на осях ознак і, отже, зниженню узагальнюючої здатності методу. Використання методу редукції нечітких термів (скорочення інтервалів) 4.7.2 дозволяє усунути ці недоліки, але лише частково.

Для забезпечення можливості вирішення задач зі складним поділом класів і побудови моделей з високою узагальнюючою здатністю пропонується за ознаками для кожного класу окремо виділяти компактні групи екземплярів, на основі значень відповідних координат яких визначати параметри інтервалів класів. Цей метод буде дозволяти за тією ж самою ознакою для різних класів виділяти цілком або частково пересічні інтервали, на основі яких можна буде більш точно визначити процедури розпізнавання.

Для знаходження параметрів інтервалів на основі запропонованого методу необхідно виконати кроки 1–21.

Крок 1. Ініціалізація. Задати навчаючу вибірку $<x, y>$. Установити номер поточної ознаки: $j = 1$.

Крок 2. Якщо $j > N$, тоді перейти до кроку 19.

Крок 3. Сформувати буферні масиви x_t та y_t: $x_t = x_j$, $y_t = y$, $\forall s$.

Крок 4. Упорядкувати пари елементів масивів x_t та y_t у порядку неспадання значень x_t. Кроки 4.1–4.7 реалізують найпростіший метод бульбашкового сортування, що на практиці може бути замінений більш швидкодіючим методом.

Крок 4.1 Установити номер поточного екземпляра: $s = 1$.

Крок 4.2 Якщо $s \le S$, тоді перейти до кроку 4.3, у протилежному випадку – перейти до кроку 5.

Крок 4.3 Установити номер поточного екземпляра: $k = s+1$.

Крок 4.4 Якщо $k \le S$, тоді перейти до кроку 4.5, у протилежному випадку – перейти до кроку 4.7.

Крок 4.5 Якщо $x^s_t > x^k_t$, тоді установити: $z = x^s_t$, $x^s_t = x^k_t$, $x^k_t = z$, $z = y^s_t$, $y^s_t = y^k_t$, $y^k_t = z$, де z – буферна змінна.

Крок 4.6 Установити: $k = k + 1$. Перейти до кроку 4.4.

Крок 4.7 Установити: $s = s + 1$. Перейти до кроку 4.2.

Крок 5. Установити номер поточного класу $q = 1$.

Крок 6. Якщо $q > K$, тоді перейти до кроку 18.

Крок 7. Знайти відстані за віссю j-ої ознаки між екземплярами вибірки, що належать до q-го класу, $\forall\ s, p = 1, 2, \ldots, S,\ s \ne p$:

$$r^q_j(s,p) = \begin{cases} |\, x^s_t - x^p_t \,|, y^s_t = y^p_t = q, \\ -1, \neg(y^s_t = y^p_t = q). \end{cases}$$

Крок 8. Знайти середню відстань за віссю j-ої ознаки між екземплярами вибірки, що належать до q-го класу:

$$\bar{r}_j^q = \frac{2}{S_q^2 - S_q} \sum_{s=1}^{S} \sum_{p=s+1}^{S} r_j^q(s,p), r_j^q(s,p) \geq 0,$$

де S_q – кількість екземплярів, що належать до q-го класу.

Крок 9. Установити номер поточного інтервалу значень j-ої ознаки екземплярів, що належать до q-го класу: $k = 0$. Установити номер поточного екземпляра $s = 1$. Установити кількість інтервалів значень j-ої ознаки для екземплярів q-го класу: $D(j, q) = 0$.

Крок 10. Якщо $s > S$, тоді перейти до кроку 17.

Крок 11. Якщо $y^s{}_t \neq q$, тоді прийняти: $s = s + 1$, перейти до кроку 10.

Крок 12. Установити: $k + 1$, $p = s + 1$, $A(j, q, k) = x^s{}_t$, $B(j, q, k) = x^s{}_t$, $n(j, q, k) = 1$, $D(j, q) = D(j, q) + 1$. Тут $A(j, q, k)$ та $B(j, q, k)$ – відповідно, ліве і праве граничні значення k-го інтервалу j-ої ознаки для екземплярів q-го класу, $n(j, q, k)$ – кількість екземплярів q-го класу, що потрапили в k-ий інтервал j-ої ознаки для екземплярів q-го класу.

Крок 13. Якщо $p > S$, тоді перейти до кроку 17.

Крок 14. Якщо $y^p{}_t \neq q$, тоді прийняти: $p = p + 1$, перейти до кроку 13.

Крок 15. Якщо $r_j^q(s,p) \leq \alpha_q \bar{r}_j^q$, де α_q – коефіцієнт, що залежить від кількості екземплярів q-го класу, тоді прийняти: $B(j, q, k) = x^p{}_t$, $n(j, q, k) = n(j, q, k)+1$, $p = p + 1$, перейти до кроку 13.

Для розрахунку α_q пропонується використовувати одну з формул:

$$\alpha_q = \frac{1}{1 + \log_2 S_q} \text{ або } \alpha_q = \frac{\alpha}{1 + S_q}, 1 \leq \alpha \leq S.$$

Крок 16. Установити: $s = p$. Перейти до кроку 10.

Крок 17. Установити: $q = q + 1$. Перейти до кроку 6.

Крок 18. Установити: $j = j + 1$. Перейти до кроку 2.

Крок 19. Для усіх виділених інтервалів значень ознак визначити $n'(j, q, k)$ – кількість влучень екземплярів, що не належать до q-го класу, у k-ий інтервал значень j-ої ознаки для екземплярів q-го класу.

Крок 19.1 Установити: $j = 1$.

Крок 19.2 Якщо $j > N$, тоді перейти до кроку 20.

Крок 19.3 Установити: $q = 1$.

Крок 19.4 Якщо $q > K$, тоді перейти до кроку 19.13.

Крок 19.5 Установити: $k = 1$, $n'(j, q, k) = 0$.

Крок 19.6 Якщо $k > D(j, q)$, тоді перейти до кроку 19.12.

Крок 19.7 Установити: $s = 1$.

Крок 19.8 Якщо $s > S$, тоді перейти до кроку 19.11.

Крок 19.9 Якщо $A(j, q, k) \leq x_j^s \leq B(j, q, k)$ та $y^s \neq q$, тоді прийняти: $n'(j, q, k) = n'(j, q, k) + 1$.

Крок 19.10 Установити: $s = s + 1$. Перейти до кроку 19.8.

Крок 19.11 Установити: $k = k + 1$. Перейти до кроку 19.6.

Крок 19.12 Установити: $q = q + 1$. Перейти до кроку 19.4.

Крок 19.13 Установити: $j = j + 1$. Перейти до кроку 19.2.

Крок 20. Визначити показники значимості (цінності) інтервалів і ознак.

Крок 20.1 Діагностичну цінність k-го інтервалу j-ої ознаки для екземплярів q-го класу пропонується визначати за формулою:

$$I(j,q,k) = \begin{cases} S_q^{-1}\left(n(j,q,k) - n'(j,q,k)\right), n(j,q,k) > n'(j,q,k), \\ 0, n(j,q,k) = n'(j,q,k), \\ -\left(S - S_q\right)^{-1}\left(n(j,q,k) - n'(j,q,k)\right), n(j,q,k) < n'(j,q,k), \end{cases}$$

$j = 1, 2,\ldots,N$; $q = 1, 2,\ldots,K$; $k = 1, 2,\ldots,D(j, q)$.

Крок 20.2 Діагностичну цінність j-ої ознаки для екземплярів q-го класу пропонується визначати за формулою:

$$I(j,q) = \frac{1}{D(j,q)} \sum_{k=1}^{D(j,q)} \frac{1}{1 + e^{-I(j,q,k)}},\ j = 1, 2,\ldots,N;\ q = 1, 2,\ldots,K.$$

Крок 20.3 Діагностичну цінність j-ої ознаки пропонується визначати за формулою:

$$I_j = \frac{1}{K} \sum_{q=1}^{K} I(j,q),\ j = 1, 2,\ldots,N.$$

Крок 21. Зупинення.

Після знаходження інтервалів й їхніх параметрів можна визначити нечіткі терми.

Як функції приналежності до нечітких термів можна використовувати трапецієподібні функції:

$$\mu_{j,q,k}(x_j^s) = \begin{cases} 0, x_j^s \le 0{,}5(A(j,q,k) + B(j,q,k-1)), \\ \dfrac{x_j^s - 0{,}5(A(j,k) + B(j,k-1))}{0{,}5(A(j,q,k) - B(j,q,k-1))},\ 0{,}5(A(j,q,k) + B(j,q,k-1)) \le x_j^s < A(j,q,k), \\ 1, A(j,q,k) \le x_j^s \le B(j,q,k), \\ \dfrac{0{,}5(A(j,q,k+1) + B(j,q,k)) - x_j^s}{0{,}5(A(j,q,k+1) - B(j,q,k))},\ B(j,q,k) \le x_j^s < 0{,}5(B(j,q,k) + A(j,q,k+1)), \\ 0,\ 0{,}5(B(j,q,k) + A(j,q,k+1)) \le x_j^s, \end{cases}$$

або П-образні функції: $\mu_{j,q,k}(x_j^s) = \mu_{j,q,k_S}(x_j^s)\mu_{j,q,k_Z}(x_j^s)$, де $\mu_{j,q,k_S}(x_j^s)$ – S-образна функція, а $\mu_{j,q,k_Z}(x_j^s)$ – Z-образна функція:

$$\mu_{i,q,k_S}(x_j^s)=\begin{cases}0, x_j^s<0{,}5(A(j,q,k)+B(j,q,k-1)),\\ \dfrac{1}{2}+\dfrac{1}{2}\cos\left(\dfrac{x_j^s-A(j,q,k)}{0{,}5(A(j,q,k)-B(j,q,k-1))}\pi\right), 0{,}5(A(j,q,k)+B(j,q,k-1))\le x_j^s\le A(j,q,k),\\ 1, x_j^s>A(j,q,k);\end{cases}$$

$$\mu_{j,q,k_Z}(x_j^s)=\begin{cases}1, x_j^s<B(j,q,k),\\ \dfrac{1}{2}+\dfrac{1}{2}\cos\left(\dfrac{x_j^s-B(j,q,k)}{0{,}5(A(j,q,k+1)-B(j,q,k))}\pi\right), B(j,q,k)\le x_j^s\le 0{,}5(B(j,q,k)+A(j,q,k+1)),\\ 0, x_j^s>0{,}5(B(j,q,k)+A(j,q,k+1)).\end{cases}$$

Також можливо використовувати гаусіан:

$$\mu_{j,q,k}(x_j^s)=\exp\left(-\frac{(x_j^s-0{,}5(B(j,q,k)-A(j,q,k)))^2}{2(\alpha_q)^2}\right).$$

Приналежність *s*-го екземпляра x^s до *q*-го класу за *j*-ою ознакою пропонується оцінювати за формулою:

$$\mu_{q,j}(x^s)=\max_{k=1,2,\ldots,D(j,q)}(I(j,q,k)\mu_{q,j,k}(x^s)),$$

$$q = 1, 2, \ldots, K;\ j = 1, 2, \ldots, N.$$

Приналежність *s*-го екземпляра x^s до *q*-го класу пропонується оцінювати за формулою:

$$\mu_q(x^s)=\max_{j=1,2,\ldots,N}(I(j,q)\mu_{q,j}(x^s))$$

$$\text{або } \mu_q(x^s)=\frac{1}{N}\sum_{j=1}^{N}I(j,q)\mu_{q,j}(x^s) \text{ або } \mu_q(x^s)=f\left(\sum_{j=1}^{N}I(j,q)\mu_{q,j}(x^s)\right),$$

де $f(x) = 1/(1+e^{-x})$, $q = 1, 2, \ldots, K$.

Використовуючи запропонований метод і формули для визначення приналежностей розпізнаваного екземпляра, можна модифікувати відповідним чином методи синтезу нейро-нечітких мереж 4.7.4–4.7.9.

4.7.12 Синтез вейвлет-нейро-нечітких діагностичних моделей

У задачах діагностики об’єктів, що характеризуються набором однотипних ознак, які є часовими відрахунками сигналів, доцільним для побудови моделей є виділення не окремих ознак, а груп суміжних ознак-відрахунків як інформативних показників і поєднувати їх за допомогою певних згорток.

Тому для подібних задач дуже ефективним може бути розкладання вихідного сигналу на складові за допомогою вейвлет-перетворення.

Вейвлет-апроксимація дискретизованого сигналу x, поданого впорядкованим у часі набором відрахунків $\{x_j\}$, $j = 1, 2,\ldots,N$, де N – кількість відрахунків, являє собою розкладання сигналу в системі базисних вейвлет-функцій:

$$x_j = \sum_g \alpha_g \psi_g\left(j, a_g, b_g\right), \; j = 1, 2, \ldots, N,$$

де g – номер базисної вейвлет-функції ψ_g, α_g – масштабний коефіцієнт для g-ої базисної функції, a_g та b_g – параметри, що визначають зсув і розтягання по часовій вісі для g-ої базисної функції (як правило, a_g та b_g задають межі інтервалу для базисної вейвлет-функції ψ_g).

Задачею побудови вейвлет-апроксимації є знаходження значень коефіцієнтів α_g, a_g та b_g. для функції ψ_g заданого виду.

Коефіцієнти для вейвлет-функцій α_g, a_g та b_g можна визначити одним з таких способів.

1. На основі багатомірної нелінійної мінімізації функціонала

$$Err = \sum_{j=1}^{N}\left(x_j - \sum_g^{\mathrm{G}} \alpha_g \psi_g\left(j, a_g, b_g\right)\right)^2 \to \min$$

у (3G)-мірному просторі, утвореному G парами керованих змінних $<\alpha_g, a_g, b_g>$, де G – задана кількість базисних функцій. Для здійснення пошуку значень керованих змінних доцільно застосовувати градієнтні методи: Левенберга-Марквардта, Ньютона, найшвидшого спуску.

Перевагою градієнтних методів оптимізації є універсальність процедур пошуку щодо можливості використання різних базових вейвлет-функцій.

До недоліків даного способу варто віднести те, що результати пошуку на основі градієнтних методів сильно залежать від вибору початкової точки пошуку, процес пошуку є високо ітеративним і повільним за часом, вимагає великих витрат обчислювальних ресурсів ЕОМ і не гарантує одержання оптимального рішення, оскільки заснований на методах локального пошуку, а також вимагає обчислення похідних функціонала, що оптимізується.

2. На основі еволюційного пошуку шляхом максимізації фітнес-функції

$$F = \frac{1}{1 + \mathrm{Err}} \to \max$$

у ($2^{D+1}G$)- мірному просторі бінарних керованих змінних, де D – кількість інтервалів на які розбиваються діапазони значень параметрів вейвлета.

До переваг даного способу можна віднести те, що еволюційний пошук є глобальним і теоретично здатний одержати найкраще рішення, а також не вимагає обчислення похідних функціонала, що оптимізується.

До недоліків даного способу варто віднести те, що він є високо ітеративним, вимагає великих витрат пам'яті й обчислювальних ресурсів ЕОМ, вимагає дискретизації значень керованих змінних, що може привести до втрати точності й істотного збільшення розмірності простору пошуку.

3. Шляхом послідовної апроксимації залишків сигналу на основі пропонованого методу, що полягає у виконанні кроків 1–5.

Крок 1. Задати максимально припустиму кількість базисних вейвлет-функций G ($G << N$), а також максимально припустиму погрішність $\varepsilon > 0$. У якості базисної вейвлет-функції задати:

$$\psi_g(j, c_g, r_g) = e^{-\left(\frac{j-c}{r}\right)^2},$$

де c_g – центр, а r_g – радіус вейвлет-функції.

Крок 2. Серед всіх екземплярів навчаючої вибірки знайти мінімальне і максимальне значення сигналів $x_{\min}$ та $x_{\max}$:

$$x_{\min} = \min_{\substack{s=1,2,...,S \\ j=1,2,...,N}} x_j^s, \quad x_{\max} = \max_{\substack{s=1,2,...,S \\ j=1,2,...,N}} x_j^s.$$

Крок 3. Пронормувати ознаки екземплярів навчаючої вибірки за формулою:

$$x_j^s = \frac{x_j^s - x_{\min}}{x_{\max} - x_{\min}}, \; s = 1, 2, \ldots, S, \; j = 1, 2, \ldots, N.$$

Крок 4. Визначити параметри базисних вейвлет-функцій для кожного екземпляра навчаючої вибірки:

$$x_j^s = \sum_{g=1}^{G} \alpha_g \psi_g\left(j, c_g, r_g\right) + \varepsilon_j^s, \; s = 1, 2, \ldots, S, \; j = 1, 2, \ldots, N,$$

де, ε^s – погрішність апроксимації, що допускається для s-го екземпляра після підсумовування не більше G базисних функцій, ε_j^s – погрішність апроксимації, що допускається для j-го відліку s-го сигналу після підсумовування не більше G базисних функцій.

Крок 4.1. Установити $s = 1$.

Крок 4.2. Якщо $s > S$, тоді перейти до кроку 5, інакше – перейти до кроку 4.3.

Крок 4.3. Установити: $g = 1$.

Крок 4.4. Виконати перевірку на закінчення процесу розкладання сигналу.

Крок 4.4.1. Визначити погрішності для відрахунку сигналу:

$$\varepsilon_j^s = x_j^s - \sum_{g'=1}^{g} \alpha_{g'} \psi_{g'}\left(j, c_{g'}, r_{g'}\right).$$

Крок 4.4.2. Визначити загальну погрішність вейвлет-апроксимації сигналу ε^s:

$$\varepsilon^s = \sum_{j=1}^{N} \left|\varepsilon_j^s\right| \text{ або } \varepsilon^s = \frac{1}{2} \sum_{j=1}^{N} \left(\varepsilon_j^s\right)^2 \text{ або } \varepsilon^s = \frac{1}{N} \sum_{j=1}^{N} \left|\varepsilon_j^s\right| \text{ або } \varepsilon^s = \max_{j=1,2,\ldots,N} \left|\varepsilon_j^s\right|.$$

Крок 4.4.3. Якщо $g > G$ пбо $\varepsilon^s < \varepsilon$, тоді перейти до кроку 4.9.

Крок 4.5. Знайти $x_j^{s\max}$ – максимальне значення для сигналу x^s та його номер j: $x_j^{s\max} = \max\limits_{j=1,2,\ldots,N} x_j^s$.

Крок 4.6. Установити: $\alpha_g = x_j^{s\max}$, $c_g = j$.

Крок 4.7. Знайти відстань r_g.

Крок 4.7.1. Установити: $r_g = 1$.

Крок 4.7.2. Якщо $r < \min(c_g,\ N - c_g)$ та $x_q^s \le x_j^s \le x_p^s, q = j - r_g, p = j + r_g$, тоді: прийняти: $r_g = r_g + 1$, перейти до кроку 4.7.2; у протилежному випадку – прийняти: $r_g = r_g - 1$.

Крок 4.7.3. Якщо $r_g = 0$, тоді прийняти: $r_g = 1/6$.

Крок 4.8. Прийняти: $x_j^s = x_j^s - \alpha_g \psi_g\left(j, c_g, r_g\right)$, $g = g + 1$. Перейти до кроку 4.4.

Крок 4.9. Установити: $s = s + 1$.

Крок 4.10. Якщо погрішність ε^s досягне значення ε при $g < G$, тоді для всіх $g' = g + 1,\ldots,G$, прийняти: $\alpha_{g'} = 0$, $c_{g'} = 0$, $r_{g'} = 1$.

Крок 4.11. Перейти до кроку 4.2.

Крок 5. Зупинення.

У результаті виконання кроків 1–4 для кожного пронормованого сигналу x^s одержимо його розкладання в системі базисних функцій. За знайденими значеннями параметрів базисних функцій c_g та r_g можна визначити значення меж інтервалів a_g та b_g: $a_g = c_g - r_g$, $b_g = c_g + r_g$.

Поєднуючи запропоновані вище методи опишемо процес синтезу вейвлет-нейро-нечіткої діагностичної моделі як послідовність кроків 1–4.

Крок 1. Ініціалізація. Задати навчаючу вибірку $<x = \{x^s\}, y = \{y^s\}>$, де $x^s = \{x_j^s\}$ – s-ий сигнал, поданий набором упорядкованих у часі відрахунків x_j^s, y^s – фактичний номер класу для s-го екземпляра.

Крок 2. Побудувати вейвлет-апроксимацію для сигналу кожного екземпляра x^s у системі не більше G базисних функцій.

Крок 3. На основі параметрів базисних функцій вейвлет-апроксимації сигналів сформувати нову навчаючу вибірку: $<x = \{x^s\}, y = \{y^s\}>$, де $x^s = \{<\alpha_g, c_g, r_g>\}$, $g = 1, 2, \ldots, G$.

Крок 4. Для нової навчаючої вибірки побудувати нейро-нечітку модель залежності $y(x)$.

Процес діагностики з використанням побудованої вейвлет-нейронечіткої моделі буде полягати у виконанні кроків 1–3.

Крок 1. Задати розпізнаваний екземпляр x^*, а також модель залежності $y(x)$.

Крок 2. Побудувати вейвлет-апроксимацію сигналу розпізнаваного екземпляра на основі G базисних функцій.

Крок 3. На основі параметрів вейвлет-апроксимації визначити за нейро-нечіткою моделлю $y(x)$ розрахункове значення номера класу y^* для розпізнаваного екземпляра x^s.

4.8 Програмні засоби для синтезу нейро-нечітких моделей

Модуль Fuzzy Logic Toolbox поряд із засобами для побудови нечітких моделей містить засоби для синтезу нейро-нечітких мереж.

ANFIS-редактор дозволяє автоматично синтезувати з експериментальних даних нейро-нечіткі мережі. Завантаження ANFIS-редактора здійснюється командою `anfisedit`. ANFIS-редактор містить меню: File, View (аналогічні меню FIS-редактору) та Edit, а також області керування створенням нейро-нечіткої мережі (рис. 4.28).

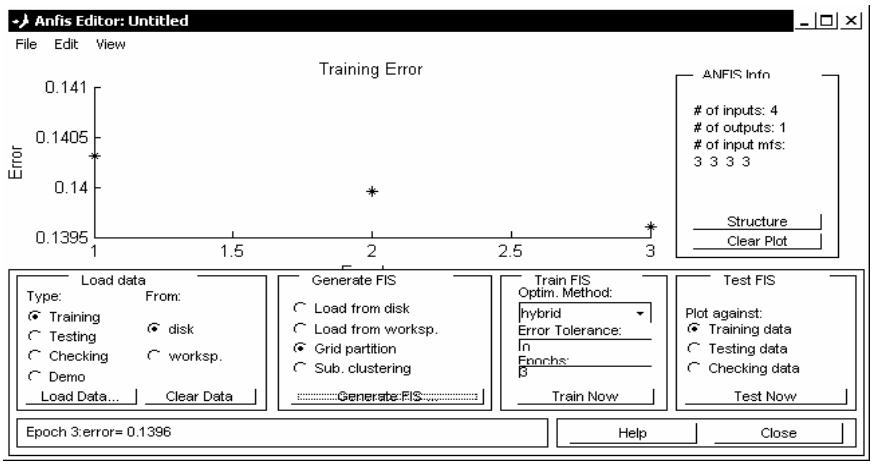


Рисунок 4.28 – ANFIS-редактор

Меню Edit. Команда Undo скасовує раніше зроблену дію. Команда FIS Properties відкриває FIS-редактор. Команда Membership Functions відкриває редактор функцій приналежності. Команда Rules відкриває редактор бази знань. Команда Anfis відкриває ANFIS-редактор.

Область візуалізації містить два типи інформації: при навчанні системи – крива навчання у вигляді графіка залежності помилки навчання від порядкового номера ітерації; при завантаженні даних і тестуванні системи – експериментальні дані і результати моделювання. Експериментальні дані і результати моделювання виводяться як множина точок у двовимірному просторі. При цьому по вісі абсцис відкладається порядковий номер рядка даних у вибірці (навчаючої, тестової або контрольної), а по осі ординат – значення вихідної змінної для даного рядка вибірки. Використовуються такі маркери: блакитна крапка (**.**) – тестова вибірка; блакитна окружність (o) – навчаюча вибірка; блакитний плюс (+) – контрольна вибірка; червона зірочка (*) – результати моделювання.

В *області властивостей* (ANFIS info) виводиться інформація про кількість вхідних і вихідних змінних, про кількість функцій приналежності для кожної вхідний змінний, а також про кількість рядків у вибірках. Натискання кнопки Structure відчиняє нове графічне вікно, у якому система нечіткого логічного виведення представляє у виді нейро-нечіткої мережі. Натискання кнопки Clear Plot дозволяє очистити область візуалізації.

В *області завантаження даних* (Load data) розташовані: меню вибору типу даних *Type*, що містить альтернативи: Traning – навчаюча вибірка, Testing – тестова вибірка, Checking – контрольна вибірка, Demo – демонстраційний приклад, меню вибору джерела даних From, що містить альтернативи: disk – диск, worksp. – робоча область Matlab; кнопка завантаження даних Load Data, по натисканню якої з'являється діалогове вікно вибору файлу, якщо завантаження даних відбувається з диска, або вікно введення ідентифікатора вибірки, якщо завантаження даних походить з робочої області; кнопка очищення даних Clear Data.

В *області генерування FIS* (Generate FIS) розташовані меню вибору способу створення вихідної системи нечіткого логічного виведення. Меню містить альтернативи: Load from disk – завантаження FIS з диска, Load from worksp. – завантаження системи з робочої області Matlab, Grid partition – генерування системи за методом ґрат (без кластеризації), Sub. clustering – генерування системи за методом субкластеризації. В області також розташована кнопка Generate, по натисканню якої генерується вихідна система нечіткого логічного виведення.

При виборі Grid partition з'являється вікно введення параметрів методу ґрат, у якому потрібно вказати кількість термів для кожної вхідної змінної і тип функцій приналежності для вхідних і вихідної змінних.

При виборі Sub. clustering з'являється вікно введення параметрів методу субкластеризації: Range of influence – рівні впливу вхідних змінних, Squash

factor – коефіцієнт пригнічення, Accept ratio – коефіцієнт, що встановлює у скільки разів потенціал даної точки повинний бути вище потенціалу центра першого кластера для того, щоб центром одного з кластерів була призначена розглянута точка, Reject ratio – коефіцієнт, що встановлює у скільки разів потенціал даної точки повинний бути нижче потенціалу центра першого кластера, щоб розглянута точка була виключена з можливих центрів кластерів.

В *області навчання* (Train FIS) розташовані: Optim. method – меню вибору методу оптимізації, Error tolerance – поле введення необхідної точності навчання, Epochs – поле введення кількості ітерацій навчання та кнопка Train Now, натискання якої запускає режим навчання. Проміжні результати навчання виводяться в область візуалізації й у робочу область Matlab. У ANFIS-редакторі реалізовані два методи навчання: backprora – метод зворотного поширення помилки та hybrid – гібридний метод, що поєднує метод зворотного поширення помилки з методом найменших квадратів.

В *області тестування* (Test FIS) розташовані меню вибору вибірки і кнопка *Test Now*, по натисканню якої відбувається тестування нечіткої системи з виведення результатів в область візуалізації.

Поряд із функціями для побудови систем нечіткого виведення модуль Fuzzy Logic Toolbox пакету Matlab містить *функції для створення й використання нейро-нечітких мереж*: *anfis* – функція навчання для системи Сугено, функція *genparam* – генерує початкові параметри передумов для навчання ANFIS, функція *evalfis* – виконує обчислення для вибірки екземплярів за допомогою вказаної нейро-нечіткої мережі.

Вибірки даних та нейро-нечіткі мережі, з якими працюють описані функції повинні зберігатися в робочій області Matlab (в оперативній пам'яті). В стовпцях матриць, відповідних вибіркам, зберігаються значення входів (ознак) та виходів, в рядках – екземпляри вибірки.

Бібліотека діагностичних функцій «Diaglab», розроблена автором, доповнює пакет Matlab набором функцій для синтезу розпізнаючих нейромережевих та нейро-нечітких діагностичних моделей багатовимірних нелінійних об'єктів та процесів. Наведемо опис деяких функцій «Diaglab».

Функція `[D,A,B,K]=fetinf(x,y)` на основі навчаючої вибірки <*x*, *y*> визначає кількість (*D*) та параметри (*A*, *B*, *K*) інтервалів зміни номеру класу для ознак екземплярів.

Функція `[g]=fetgroup(x,y,alfa,betta,gamma,width)` на основі навчаючої вибірки <*x*, *y*> розбиває ознаки на групи та повертає масив *g*, рядки якого відповідають групам ознак, а елементи містять номери ознак, що входять до груп. Параметри alfa, betta, gamma, width відповідають параметрам α, β, γ та width методу синтезу нейро-нечітких мереж з групуванням ознак.

Функція `[FNN]=FNNnew(x,y,model,options)` на основі навчаючої вибірки <*x*, *y*> створює та навчає нейро-нечітку мережу, яку повертає у змінній

FNN. Тип моделі нейро-нечіткої мережі має бути визначеній як вхідний параметр model, який може приймати значення: '3class' – тришарова мережа для класифікації, '4class' – чотиришарова мережа для класифікації, 'hierarchy' – ієрархічна мережа, 'fetgroup' – нейромережа з групуванням ознак, 'fetgroupconv' – мережа з групуванням та згорткою ознак, 'clreg' – нейро-нечітка кластер-регресійна апроксимація, 'wavelet' – вейвлет-нейро-нечітка мережа. Параметр options є масивом значень параметрів, власних для кожного методу.

Функція `[y]=FNNsim(FNN,x)` емулює роботу нейро-нечіткої мережі FNN для екземплярів вибірки *x* та повертає обчислені значення виходу у змінній *y*.

4.9 Приклади та ілюстрації

Приклад 1. Використання функцій створення нейро-нечітких мереж модуля Fuzzy Logic Toolbox пакету Matlab.

Апроксимувати залежність: $y = \frac{\sin 2x}{e^{x/5}}$, $x \in [0, 10]$, $\Delta x = 0{,}1$.

```
x = (0:0.1:10)'; % задаємо масив x
y = sin(2.*x)./exp(x./5); % визначаємо значення елементів масиву y
trnData = [x y]; % створюємо масив навчаючої вибірки
numMFs = 5; % кількість функцій приналежності
mfType = 'gbellmf'; % тип функцій приналежності
epoch_n = 20;% кількість ітерацій
% створюємо FIS-структуру типу Сугено
in_fismat = genfis1(trnData,numMFs,mfType);
% навчаємо FIS-структуру
out_fismat = anfis(trnData,in_fismat,20);
% будуємо графіки:
% цільової залежності – синій колір, суцільна лінія
% апроксимуючої залежності – зелений колір, пунктир
plot(x,y,x,evalfis(x,out_fismat));
% підписуємо легенду до графіків
% Training Data – навчаюча вибірка – цільові дані
% ANFIS Output – вихід ANFIS – розрахункові дані
legend('Training Data','ANFIS Output');
```

Приклад 2. Використання функції fetinf бібліотеки «Diaglab».

Нехай ми маємо масив *x*, стовпці якого відповідають екземплярам навчаючої вибірки, а рядки – їхнім ознакам, та масив *y*, елементи якого відповідають номерам класів, співставленим екземплярам навчаючої вибірки *x*.

У командному вікні пакету Matlab введемо команду:

```
[D,A,B,K]=fetinf(x,y);
```

У результаті виконання цієї команди отримаємо кількість (*D*) та параметри (*A*, *B*, *K*) інтервалів зміни номеру класу для ознак екземплярів.

Приклад 3. Використання функції побудови та емуляції нейро-нечітких мереж бібліотеки «Diaglab». Нехай, як у прикладі 2, ми маємо масиви *x* та *y*.

У командному вікні пакету Matlab введемо команду:

```
[FNN]=FNNnew(x,y,'3class');
```

У результаті виконання цієї команди отримаємо у змінній FNN тришарову нейро-нечітку мережу для класифікації.

У командному вікні пакету Matlab введемо команду:

```
[y1]=FNNsim(FNN,x);
```

У результаті виконання цієї команди отримаємо у змінній *y*1 результати розпізнавання екземплярів вибірки x нейро-нечіткою мережею FNN.

? 4.10 Контрольні питання

1. Порівняйте властивості нечітких систем та нейронних мереж.
2. Що таке нейро-нечітка мережа?
3. Які властивості мають нейро-нечіткі мережі?
4. Які існують типи нейро-нечітких мереж?
5. Як формують базу знань нейро-нечіткої мережі?
6. З яких елементів складаються нейро-нечіткі мережі?
7. У чому полягають особливості паралельних нейро-нечітких мереж?
8. Що таке нейро-нечтітка асоціативна пам'ять Коско?
9. Як відбувається виділення нечітких правил за допомогою карт, що самоорганізуються?
10. Опишіть особливості систем, що здатні навчати нечіткі множини.
11. У чому полягають особливості конкурентних нейро-нечітких систем?
12. У чому полягають особливості інтегрованих нейро-нечітких систем?
13. Опишіть архітектуру та методи навчання нейро-нечітких мереж: апроксиматора Мамдані, FALCON, мережі Такагі-Сугено-Канга, мережі Ванга-Менделя, ANFIS, NNFLC, SONFIN, нечіткого персептрона, NEFCON, NEFCLASS, NEFPROX, NNDFR, GARIC, FINEST, FUN, EFuNN, dmEFuNN, гібридної мережі з нечіткою самоорганізацією.
14. У чому полягають основні принципи використання методу зворотного поширення помилки для навчання нейро-нечітких мереж?
15. Які існують основні етапи гібридного методу навчання мереж Такагі-Сугено-Канга?
16. Як визначити апріорну інформацію про навчаючу вибірку?
17. У чому полягає метод редукції нечітких термів?
18. Як і для чого можна виконати об'єднання суміжних термів по ознаках?
19. Як побудувати тришарову розпізнаючу нейро-нечітку модель у неітеративному режимі?

20. Як побудувати чотиришарову розпізнаючу нейро-нечітку модель у неітеративному режимі?

21. Як і для чого можна побудувати ієрархічну логічно прозору нейро-нечітку модель?

22. Як побудувати нейро-нечіткі моделі з групуванням та згорткою ознак?

23. Як виконати об'єднання нечітких термів у кластери?

24. Як побудувати нейро-нечітку кластер-регресійну апроксимацію?

25. Як можна донавчити нейро-нечіткі моделі, побудовані із використанням інтервалів значень ознак?

26. Як побудувати вейвлет-нейро-нечітку модель?

27. Опишіть функції синтезу нейро-нечітких моделей бібліотеки діагностичних функцій «Diaglab».

28. Опишіть функції пакету Matlab для створення нейро-нечітких мереж.

29. Опишіть редактор anfisedit.

30. Який метод кластер-аналізу призводить до отримання нейро-нечіткої мережі меншої складності?

29. Як впливає задана кількість циклів навчання на точність навчання?

30. Як впливає задана точність навчання на тривалість навчання?

31. Які вимоги мають пред'являтися до навчаючої вибірки та як це вплине на процес навчання?

4.11 Практичні завдання

Завдання 1. Згідно з номером індивідуального варіанта студента за журналом згенерувати навчаючу та тестову вибірки даних.

Нехай V – номер студента за журналом, а rand – функція, що генерує псевдовипадкові числа у діапазоні [0; 1]. Визначимо кількість екземплярів у навчаючій вибірці $S_{н.}$, кількість екземплярів у тестовій вибірці $S_{т.}$, кількість вхідних змінних (ознак) вибірок N, значення ознак вибірок: навчаючої – $x_{н.}$ та тестової – $x_{т.}$, а також значення цільових ознак для вибірок: навчаючої – $y_{н.}$ та тестової – $y_{т.}$:

$$N = \begin{cases} V, \text{якщо } V < 15, \\ V - 14, \text{якщо } V > 15; \end{cases}$$

$$S_{н.} = \begin{cases} 10\,\text{V}, \text{якщо } V < 5, \\ 5V, \text{якщо } 5 \le V < 10, \\ 2V, \text{якщо } 10 \le V < 20, \\ V, \text{інакше}; \end{cases} \quad S_{т.} = \begin{cases} 8V, \text{якщо } V < 5, \\ 6V, \text{якщо } 5 \le V < 10, \\ 3V, \text{якщо } 10 \le V < 20, \\ 2V, \text{інакше}; \end{cases}$$

$$x_{н.} = \{x_j^s\},\ y_{н.} = \{y^s\},\ s = 1, 2,...,S_{н.};\ x_{т.} = \{x_j^s\},\ y_{т.} = \{y^s\},\ s = 1, 2,..., S_{т.};$$

$$x_j^s = \frac{V}{s \cdot j} \cdot \text{rand},\ j = 1,2,...,N,$$

$$y^s = \begin{cases} 0{,}1(x^s{}_1 + x^s{}_2), x^s{}_1 < V, \\ 0{,}3(x^s{}_1 - x^s{}_2), x^s{}_1 > V, \\ 0{,}5x^s{}_1, x^s{}_1 = V. \end{cases}$$

Для згенерованих вибірок за допомогою редактору anfisedit побудувати нейро-нечітку модель. Спробувати використати різні методи кластеризації, різні кількості функцій приналежності для входів, різні кількості циклів навчання та різні методи навчання. Протестувати побудовану модель. Проаналізувати отримані результати та відповісти на питання: який метод кластер-аналізу призводить до отримання мережі меншої складності; як впливає задана кількість циклів навчання на точність навчання; як вливає задана точність навчання на тривалість навчання; які вимоги мають пред'являтися до навчаючої вибірки та як це вплине на процес навчання.

Завдання 2. Виконати завдання 1 у режимі командного вікна без застосування редактору anfisedit, використовуючи функції модуля Fuzzy Logic Toolbox.

Завдання 3. Виконати завдання 1 у режимі командного вікна, використовуючи функції бібліотеки «Diaglab».

Завдання 4. На алгоритмічній мові програмування високого рівня (наприклад, Сі, Паскалі, мові пакету Matlab) створити програму, що реалізує одну з моделей нейро-нечітких мереж.

Завдання 5. Підготувати реферат на одну з таких тем.

1. Подання нечітких знань та формування нечітких баз знань.
2. Класифікація та властивості моделей нейро-нечітких мереж.
3. Моделі нейро-нечітких мереж, заснованих на виведенні Мамдані.
4. Моделі нейромереж, заснованих на виведенні Такагі-Сугено-Канга.
5. Використання характеристик вибірок даних для створення та навчання нейро-нечітких мереж.
6. Застосування нейро-нечітких мереж в задачах діагностики.
7. Програмні засоби для синтезу нейро-нечітких моделей.

4.12 Література до розділу

Моделі, методи синтезу і навчання, а також застосування нейронних мереж описано в [49, 52–55, 58, 68–70, 97, 98, 107, 132, 148, 149, 151, 164, 170, 171, 180, 182, 189, 203, 205, 206]; моделі, методи синтезу і навчання, а також застосування нейро-нечітких мереж розглянуто в [2, 4–7, 21, 27, 39, 49, 50, 59, 66, 67, 81, 97, 98, 101, 105, 113, 114, 122, 129, 132, 156, 164, 172–175,

177–179]; описи програмних засобів синтезу нейро-нечітких моделей наведено у [2, 21, 49, 56–58, 62, 98, 105, 205].

Навчально-методичні матеріали до розділу доступні на сайті автора в мережі Інтернет за адресою: `http://csit.narod.ru`.

ЛІТЕРАТУРА

1. Breiman L., Shang N. Distribution based trees are more accurate [Electronic resource]. – Access mode: ftp://ftp.stat.berkeley.edu.

2. Fuzzy Logic Toolbox for use with Matlab User's guide. – Natick: Mathworks, 1998. – 235 p.

3. Hoffmann A. G. Paradigms of Artificial Intelligence: a methodological and computational analysis. – Singapore: Springer-Verlag, 1998. – 234 p.

4. Kruse R., Nauck D. Learning methods for fuzzy systems // Proceedings of Third German GI-Workshop «Fuzzy-Neuro-Systeme'95». – Darmstadt (Germany), 1995. – P. 7–22.

5. Kruse R., Nauck D. NEFCLASS – a neuro-fuzzy approach for the classification of data // Applied computing: Proc. of the 1995 ACM Syphosium on Applied Computing. – Nashville: ACM Press, New York, 1995. – P. 461–465.

6. Nauck D., Klawonn F. Neuro-fuzy classification initialized by fuzzy clustering // Proc. Fourth European Congress on Intelligent Techniques and Soft Computing (EUFIT'96). – Aachen: Verlag und Druck Mainz GmbH, 1996. – P. 1551–1555.

7. Nauck D., Kruse R., Stellmach R. New learning algorithms for the neuro-fuzzy environment NEFCON-I // Proceedings of Third German GI-Workshop « Neuro-Fuzzy-Systeme'95». – Darmstadt, 1995. – P. 357–364.

8. Айвазян С. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д. Прикладная статистика. Классификация и снижение размерности. – М.: Финансы и статистика, 1989. – 607 с.

9. Алиев Р. А. Интеллектуальные роботы с нечеткими базами знаний. – М. – Радио и связь, 1995. – 177 с.

10. Алиев Р. А., Абдикеев Н. М., Шахназаров М. М. Производственные системы с искусственным интеллектом. – М.: Радио и связь, 1990. – 264 с.

11. Алиев Р. А., Церковный А. Э., Мамедова Г. А. Управление производством при нечеткой исходной информации. – М.: Энергоатомиздат, 1991. – 238 с.

12. Андреев И. Деревья решений – CART математический аппарат // «Exponenta Pro (математика в приложениях)» 2004. – № 3–4.

13. Андрейчиков А. В., Андрейчикова О. Н. Интеллектуальные информационные системы: Учебник. – М.: Финансы и статистика, 2004. – 424 с.

14. Архангельский В. И., Богаенко И. Н., Грабовский Г. Г., Рюмшин Н. А. Системы фуцци-управления. – К.: Тэхника, 1997. – 208 с.

15. Асанов А. А., Кочин Д. В. Метод выявления решающих правил в задачах экспертной классификации // Искусственный интеллект. – 2002. – № 2. – С. 20–31.

16. Асельдеров З. М., Гузенко Д. О., Делеган М. М. Побудова баз знань за допомогою автоматичного висування гіпотез // Искусственный интеллект. – 2006. – № 4. – С. 118–128.

17. Бакаев А. А., Гриценко В. И., Козлова Д. Н. Методы организации и обработки баз знаний. – К.: Наукова думка, 1993. – 150 с.

18. Барсегян А. А. Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP: Уч. пос. – СПб: BHV, 2007. – 384 с.

19. Белецкий Б. А., Вагис А. А., Васильев С. В., Гупал Н. А. Сложность байесовской процедуры индуктивного вывода. Дискретный случай // Проблемы управления и информатики. – 2006. – № 6. – С. 55–70.

20. Беллман Р., Заде Л. Принятие решений в расплывчатых условиях – В сб.: Вопросы анализа и процедуры принятия решений.– М: Мир, 1976.– С.172–215.

21. Бодянский Е. В., Кучеренко Е. И., Михалев А. И. Нейро-фаззи сети Петри в задачах моделирования сложных систем. – Днепропетровск: Системные технологии. – 2005. – 311 с.

22. Бондарев В. Н., Аде Ф. Г. Искусственный интеллект. – Севастополь: СевНТУ, 2002. – 615 с.

23. Бондаренко М. Ф., Гребенюк В. О., Кайкова О. Б., Терзиян В. Я. Теория многоуровневых семантических сетей. Учебное пособие. – Харьков: ХГТУРЭ. – 1997. – 76 с.

24. Борисов А. Н., Алексеев А. В., Крумберг О. А. и др. Модели принятия решений на основе лингвистической переменной. – Рига: Зинатне, 1982. – 256 с.

25. Борисов А. Н., Алексеев А. В., Меркурьева Г. В. и др. Обработка нечеткой информации в системах принятия решений.– М: Радио и связь, 1989. – 304 с.

26. Борисов А. Н., Крумберг О. А., Федоров И. П. Принятие решений на основе нечетких моделей. Примеры использования.– Рига: Зинатне, 1990. – 184 с.

27. Борисов В. В., Круглов В. В., Федулов А. С. Нечеткие модели и сети. – М.: Горячая линия-Телеком. – 2007. – 284 с.

28. Бочарников В. П. Fuzzy technology: Модальности и принятие решений в маркетинговых коммуникациях. – К.: Эльга-Ника-Центр, 2002. – 224 с.

29. Брянцев И. Н. Data Mining. Теория и практика. – М.: БДЦ-Пресс, 2006. – 208 с.

30. Будько В. И., Беленков В. Г., Синицын И. Н., Рыков А. С. Алгоритмы обработки экспертной информации // Информационные технологии. – 2003. – № 10. – С. 56–60.

31. Булкин В. И., Шаронова Н. В. Формальное представление знаний в продукционных системах // Искусственный интеллект. – 2006. – № 1. – С. 147–157.

32. Бурдаев В. П. Структура базы знаний для выбора мобильного телефона // Искусственный интеллект. – 2006. – № 4. – С. 681–688.

33. Веревка О. В., Парасюк И. Н. Линейная интерполяция в нечетком информационном пространстве // Кибернетика и системный анализ. – 2006. – № 2. – С. 55–68.

34. Веревка О. В., Парасюк И. Н. Математические основы построения нечетких байесовских механизмов вывода // Кибернетика и системный анализ. – 2002. – № 1. – С. 105–117.

35. Вороной С. М., Бабин Д. В., Малащук Е. В. Повышение эффективности извлечения знаний на основе интеллектуального анализа и структурирования информации // Искусственный интеллект. – 2005. – № 3. – С. 259–264.

36. Галуев Г. А. Интеллектуальные среды нового поколения: состояние проблемы и перспективы решения // Искусственный интеллект. – 2004. – № 3. – С. 523–533.

37. Герасимов Б. М., Тарасов В. А., Токарев И. В. Человеко-машинные системы принятия решений с элементами искусственного интеллекта. – К.: Наукова думка, 1993. – 180 с.

38. Герман О. В. Получение выводов в противоречивых системах // Кибернетика и системный анализ. – 2005. – № 5. – С. 29–41.

39. Гимаров В. А. Нейро-нечеткий идентификатор // Нейрокомпьютеры: разработка, применение. – 2003. – № 2. – С. 29–34.

40. Гнатчук Е. Г. Обработка нечеткой информации в нечеткой экспертной системе диагностирования компьютерных средств // Искусственный интеллект. – 2006. – № 4. – С. 526–533.

41. Гостев В. И. Синтез нечетких регуляторов систем автоматического управления. – К.: Радиоаматор, 2003. – 488 с.

42. Гречихин Л.И., Шумский И.П. Многоуровневая автоматическая система принятия решений на основе нечетких исходных данных // Искусственный интеллект. – 2004. – № 3. – С. 287–292.

43. Григорьев А. В. Пути создания интеллектуальных САПР при различных уровнях квалификации экспертов // Искусственный интеллект. – 2005. – № 3. – С. 758–763.

44. Гуляев В. А., Бугаев А. Е., Аль-Хадиди М. Применение нечеткой логики в управляющих и диагностических устройствах. – Электронное моделирование. – 1993. – Т. 15, № 4. – С. 69–73.

45. Гусев Л. А., Смирнова И. М. Размытые множества. Теория и приложения (обзор) // Автоматика и телемеханика. – 1973. – № 5. – С.66–85.

46. Джарратано Д., Райли Г. Экспертные системы: принципы разработки и программирование. – М.: Вильямс, 2007. – 1152 с.

47. Джексон П. Введение в экспертные системы: Пер с англ. – М.: Вильямс, 2001. – 624 с.

48. Джонс М. Т. Программирование искусственного интеллекта в приложениях / Пер. с англ. А. И. Осипов. – М.: ДМК Пресс, 2004. – 312 с.

49. Дли М. И. Нечеткая логика и искусственные нейронные сети. – М.: Физматлит, 2003. – 225 с.

50. Дли М. И., Стоянова О. В. Метод прогнозирования экономических показателей на основе самоорганизующихся нейронечетких сетей. // Нейрокомпьютеры: разработка, применение. – 2005. – № 6. – С. 26–29.

51. Досин Д. Г., Даревич Р. Р., Пугач Т. М., Литвин В. В. Модель представления знаний посредством объектов для построения интеллектуальных систем поддержки принятия решений // Искусственный интеллект. – 2004. – № 3. – С. 343–349.

52. Дубровін В. І., Субботін С. О. Методи оптимізації та їх застосування в задачах навчання нейронних мереж: Навчальний посібник. – Запоріжжя: ЗНТУ, 2003. – 136 с.

53. Дубровин В. И., Субботин С. А. Алгоритм классификации с оценкой значимости признаков // Радіоелектроніка. Інформатика. Управління. – 2001. – № 2. – С. 145–150.

54. Дубровин В. И., Субботин С. А. Алгоритм кластер-регрессионной аппроксимации и его нейросетевая интерпретация // Информатика и системы управления. – 2002. – № 2. – С. 63–69.

55. Дубровин В. И., Субботин С. А. Алгоритм синтеза и настройки весов многослойной нейронной сети // Сборник трудов научно-технической конференции «Нейроинформатика-2003», 29-31 января 2003 г. – М.: МИФИ, 2003. – Ч.1. – С. 68–76.

56. Дубровин В. И., Субботин С. А. Библиотека дагностических функций DiagLab // Моделирование неравновесных систем: Материалы VI Всероссийского семинара, 24–26 октября 2003 г. / Под ред. В. И. Быкова. – Красноярск: ИВМ СО РАН, 2003. – С. 61–63.

57. Дубровин В. И., Субботин С. А. Библиотека диагностических функций DIAGLAB для пакета MATLAB // Проектирование и технология электронных средств. – 2003. – № 3. – С. 40–46.

58. Дубровин В. И., Субботин С. А., Богуслаев А. В., Яценко В. К. Интеллектуальные средства диагностики и прогнозирования надежности авиадвигателей: Монография. – Запорожье: ОАО «Мотор-Сич». – 2003. – 279 с.

59. Дудкин А. А., Ваткин М. Е. Нечеткая нейронная сеть для классификации объектов на полутоновых изображениях // Искусственный интеллект. – 2005. – № 3. – С. 272–281.

60. Дюбуа Д., Прад А. Теория возможностей. Приложения к представлению знаний в информатике: Пер. с фр. – М.: Радио и связь, 1990. – 288 с.

61. Дюк В., Самойленко А. Data mining: учебный курс. – СПб.: Питер, 2001. – 368 с.

62. Дьяконов В. MATLAB 6: учебный курс. – СПб.: Питер, 2001. – 592 с.

63. Дьяконов В. П. Вейвлеты. От теории к практике. – М.: Солон-Р, 2002. – 448 с.

64. Жамбю М. Иерархический кластер-анализ и соответствия. – М.: Финансы и статистика, 1988. – 342 с.

65. Жернаков С. В. Активная экспертная система комплексного мониторинга и управления эксплуатацией авиационных двигателей // Информационные технологии. – 2001. – № 12. – С. 27–31.

66. Жернаков С. В. Диагностика и контроль параметров маслянной системы ГТД гибридными нейронечеткими экспертными системами // Информационные технологии. – 2000. – № 3. – С. 31–36.

67. Жернаков С. В. Комплексная диагностика и контроль параметров ГТД в условиях неопределенности на базе нечеткой экспертной системы TILShell 3.0+ // Информационные технологии. – 2000. – № 8. – С. 33–40.

68. Жернаков С. В. Нейросетевая экспертная система комплексного мониторинга и управления эксплуатацией авиационных двигателей // Нейрокомпьютеры: разработка, применение. – 2001. – № 6. – С. 33–40.

69. Жернаков С. В. Применение динамических экспертных систем с нейросетевыми базами знаний в процессе эксплуатации авиационных двигателей // Информационные технологии. – 2001. – № 6. – С. 42–47.

70. Жернаков С. В. Применение экспертных систем с нейросетевыми базами знаний к диагностике и контролю устройств аиационных двигателей // Информационные технологии. – 2000. – № 12. – С. 37–44.

71. Жуковская О. А., Файнзильберг Л. С. Формальная оценка квалификации эксперта на основе байесовской модели и методов интервального анализа // Проблемы управления и информатики. – 2005. – № 3. – С. 103–115.

72. Заводнов Е. С., Лютикова Л. А. Логический подход к моделированию и минимизации баз знаний // Искусственный интеллект. – 2005. – № 3. – С. 158–161.

73. Заде Л. Понятие лингвистической переменной и его применение к принятию приближенных решений. – М.:Мир, 1976. – 165 с.

74. Заде Л. А. Основы нового подхода к анализу сложных систем и процессов принятия решений. – В кн.: Математика сегодня. – М.:Знание, 1974. – С. 5–49.

75. Заде Л. А. Понятие лингвистической переменной и его применение к принятию приближенных решений. – М.: Мир, 1976. – 165 с.

76. Заде Л. А. Размытые множества и их применение в распознавании образов и кластер-анализе. – В сб.: Классификация и кластер. – М: Мир, 1980, С. 208–247.

77. Зайченко Ю. П. Основи проектування інтелектуальних систем. Навчальний посібник. – К.: Слово, 2004. – 352 с.

78. Зайченко Ю. П., Заец И. О., Камоцкий О. В., Павлюк О. В. Исследование разных видов функций принадлежности параметров нечетких прогнозирующих моделей в нечетком методе группового учета аргументов // Управляющие системы и машины. – 2003. – № 2. – С. 56–67.

79. Зубов Д. А. Программное обеспечение гибридной экспертной системы управления технологическими процессами углеобогащения // Радіоелектроніка. Інформтика. Управління. – 2002. – № 2. – С. 87–92.

80. Зуенков М. А. Приближение характеристических функций нечетких множеств // Автоматика и телемеханика.– 1984.– № 10.– С. 138–149.

81. Ильясов Б. Г., Исмагилова Л. А., Валенева Р. Г., Сергеева И. Г. Применение нейро-нечетких моделей в управлении производством // Нейрокомпьютеры: разработка, применение. – 2001. – № 4–5. – С. 36–43.

82. Каменева И. П. Вероятностные модели репрезентации знаний в интеллектуальных системах принятия решений // Искусственный интеллект. – 2005. – № 3. – С. 399–409.

83. Кандель А., Байатт У.Дж. Нечеткие множества, нечеткая алгебра, нечеткая статистика // Труды американского общества инженеров-радиоэлектроников. – 1978. – Т. 66. – №12. – С. 37–61.

84. Капитонова Ю. В. Общие принципы построения знание-компьютерных систем // Кибернетика и системный анализ. – 2006. – № 4. – С. 81–101.

85. Кирсанова Е. В., Субботин С. А. Обобщенный метод кластер-регрессионной аппроксимации в задаче моделирования показателя здоровья детей // Радіоелектроніка. Інформатика. Управління. – 2004. – № 1. – С. 62–67.

86. Классификация и кластер / под ред. Дж. Вэн Райзина. – М.: Мир, 1980. – 392 с.

87. Клещев А. С., Смагин С. В. Распараллеливание вычислений при решении задачи индуктивного формирования баз знаний // Искусственный интеллект. – 2006. – № 3. – С. 421–428.

88. Коваль В. Н., Кук Ю. В. Извлечение и анализ знаний // Искусственный интеллект. – 2004. – № 3. – С. 293–304.

89. Коваль В. Н., Кук Ю. В. Методика приобретения новых знаний в семантических сетях «объект – предикат» // Искусственный интеллект. – 2005. – № 3. – С. 25–36.

90. Козырев Г. И., Назаров А. В., Лоскутов А. И. Экспертная система прогнозирования в пространстве параметров предметной области // Нейрокомпьютеры: разработка, применение. – 2001. – № 11.– С. 64 –68.

91. Кокорева Л. В., Перевозчикова О. Л., Ющенко Е. Л. Диалоговые системы и представление знаний. – К.: Наукова думка, 1993. – 444 с.

92. Коломейко В. В. Методологические аспекты построения человеко-машинных систем поддержки принятия решений // Искусственный интеллект. – 2002. – № 3. – С. 101–106.

93. Комп'ютерна програма «Бібліотека діагностичних функцій DiagLab»: Свідоцтво про реєстрацію авторського права на твір № 7865 / В. І. Дубровін, С. О. Субботін. – Держ. департамент інтелектуальної власності, Зареєстр. 27.06.2003.

94. Корхин А. С., Мизерный В. Н. Использование нечеткой априорной информации для оценивания параметров регрессии // Проблемы управления и информатики. – 2003. – № 1. – С. 49–61.

95. Коршунов Ю. М. Математические основы кибернетики. – М.: Энергоатомиздат, 1987. – 497 с.

96. Кофман А. Введение в теорию нечетких множеств.– М.: Радио и связь, 1982. – 432 с.

97. Кричевский М. Л. Интеллектуальные методы в менеджменте. – СПб.: Питер, 2005. – 304 с.

98. Круглов В. В., Борисов В. В. Искуственные нейронные сети. Теория и практика. – М.: Горячая линия – Телеком, 2001. – 382 с.

99. Кузин Е. С. Представление знаний и решение информационно-сложных задач в компьютерных системах // Информационные технологии. – 2004. – № 4. – Приложение к журналу. – 32 с.

100. Кузнецов Д. А. Интеллектуальная система поддержки принятия решений прогнозирования заболеваний на основе нечёткой логики // Искусственный интеллект. – 2004. – № 3. – С. 337–342.

101. Куссуль Н.Н. Обучение нейронных сетей с использованием метода нечетких эллипсоидальных оценок // Проблемы управления и информатики. – 2001. – № 1. – С. 72–78.

102. Кучеренко Е.И., Павлов Д.А. Некоторые аспекты анализа развития нечетких онтологий // Искусственный интеллект. – 2005. – № 3. – С. 162–169.

103. Ларин С. Применение ассоциативных правил для стимулирования продаж [Электронный ресурс]. – М.: Basegroup, 2003. – Режим доступа: http://www.basegroup.ru/practice/salepromotion.htm.

104. Левин Р., Дранг Д., Эдельсон Б. Практическое введение в технологию искусственного интеллекта и экспертных систем с иллюстрациями на бейсике / Пер. с англ. и предисл. М. Л.Сальникова. – М.: Финансы и статистика, 1991. – 237 с.

105. Леоненков А. В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. – СПб.: БХВ-Петербург, 2003. – 736 с.

106. Лорьер Ж.-Л. Системы искусственного интеллекта: Пер. с франц. – М.: Мир, 1991. – 568 с.

107. Люгер Дж.Ф. Искусственный интеллект: стратегии и методы решения сложных проблем / Пер. с англ. – М. – Вильямс, 2005. – 864 с.

108. Малышев Н. Г., Бернштейн Л. С., Боженюк А. В. Нечеткие модели для экспертных систем в САПР. – М.: Энергоиздат, 1991. – 136 с.

109. Мамедова М. Г., Джабраилова З. Г. Применение нечеткой логики в демографическом прогнозе // Информационные технологии. – 2004. – № 3. – С. 45–53.

110. Мелихов А. Н., Берштейн Л. С., Коровин С. Я. Ситуационные советующие системы с нечеткой логикой.– М.: Наука, 1990.– 272 с.

111. Мешалкин В. П. Экспертные системы в химической технологии. – М.: Химия, 1995. – 368 с.

112. Миронов А. С. Представление и обработка знаний в одном семействе интеллектуальных информационных систем // Информационные технологии. – 2004. – № 3. – С. 39–45.

113. Митюшкин Ю. И., Мокин Б. И., Ротштейн А. П. Soft Computing: идентификация закономерностей нечеткими базами знаний. – Винница: УНИВЕРСУМ-Винница, 2002 . – 145 с.

114. Михаль О. Ф., Руденко О. Г., Халайбех Зияд Моделирование системы нечеткого регулирования средствами нечетких сетей Петри // Управляющие системы и машины. – 2005. – № 4. – С. 3–7.

115. Мочалов И. А., Петрунин Н. Г., Редькин А. С., Цегельский С. В. Нечеткие методы теории вероятностей и математической статистики // Информационные технологии, Приложение. – 2003. – № 4. – С. 1–24.

116. Мюллер Й.-А. Извлечение знаний из данных на основе МГУА // Управляющие системы и машины. – 2003. – № 2. – С. 13–20.

117. Насибов Э. Н. К вопросу агрегации нечеткой информации на базе длекомпозиционного представления // Кибернетика и системный анализ. – 2005. – № 2. – С. 176–186.

118. Нейлор К. Как построить свою экспертную систему. – М.: Энергоатомиздат, 1991. – 286 с.

119. Некрашевич С. П., Божко Д. В. Представление данных в Интернет на основе семантических сетей // Искусственный интеллект. – 2006. – № 1. – С. 57–65.

120. Несенюк А. П. Неопределенные величины в задачах управления с неполной информацией // Автоматика .– 1979. – № 2. – С. 55–64.

121. Нетрадиционные модели и системы с нечеткими знаниями / под ред. А.Ф. Блишуна. – М.: Энергоатомиздат, 1991. – 144 с.

122. Нечаев Ю. Н., Дубров С. Н. Нечеткие нейросетевые модели представления и обработки знаний в интеллектуальных обучающих системах // Искусственный интеллект. – 2002. – № 4. – С. 734–741.

123. Нечеткие множества в моделях управления и искусственного интеллекта / А. Н. Аверкин, И. З. Батыршин, А. Ф. Блишун, В. Б. Силов, В. Б. Тарасов. Под ред. Д. А. Поспелова. – М.: Наука, 1986. – 312 с.

124. Нечеткие множества и теория возможностей. Последние достижения / Под ред. Р. Р. Ягера. – М.: Радио и связь, 1986. – 408 с.

125. Нильсон Н. Принципы искусственного интеллекта / Пер. с англ. – М.: Радио и связь, 1985. – 376 с.

126. Нильсон Н. Дж. Искусственный интеллект. Методы поиска решений. – М.: Мир, 1973. – 270 с.

127. Новак М., Перфильева И., Мочкорж И. Математические принципы нечеткой логики / Пер с англ.; под ред. А. Н. Аверкина. – М.: Физматлит. – 2006. – 352 с.

128. Новоселова Н. А. Построение нечеткой модели классификации с использованием многокритериального генетического алгоритма // Искусственный интеллект. – 2006. – № 3. – С. 613–622.

129. Новоселова Н. А., Том И. Э., Красько О. В. Нечеткое нейросетевое моделирование для получения интерпретируемого набора классифицирующих правил // Искусственный интеллект. – 2006. – № 2. – С. 211–214.

130. Орлов А. И. Задачи оптимизации и нечеткие переменные. – М.: Знание, 1980. – 64 с.

131. Орловский С. А. Проблемы принятия решений при нечеткой исходной информации. – М.: Радио и связь, 1981. – 286 с.

132. Осовский С. Нейронные сети для обработки информации. – М.: Финансы и статистика, 2004. – 344 с.

133. Осуга С. Обработка знаний. – М.: Мир, 1989. – 293 с.

134. Пелещишин А. М., Шаховская Н. Б. Использование аппарата нечетких множеств для описания аудитории веб-сайта // Искусственный интеллект. – 2005. – № 3. С. 524–528.

135. Поспелов Г. С. Искусственный интеллект – основа новой информационной технолгии. – М.: Наука, 1988. – 280 с.

136. Поспелов Г. С., Поспелов Д. А. Искусственный интеллект – прикладные системы. – М.: Знание, 1985. – 48 с.

137. Поспелов Д. А. Большие системы. Ситуационное управление. – М.: Знание. – 1975. – 64 с.

138. Поспелов Д. А. Логико-лингвистические модели в системах управления. – М.: Энергоатомиздат, 1981. – 232 с.

139. Поспелов Д. А. Моделирование рассуждений. Опыт анализа мыслительных актов. – М.: Радио и связь, 1989. – 184 с.

140. Поспелов Д. А. Ситуационное управление: теория и практика. – М.: Наука, 1986. – 288 с.

141. Построение экспертных систем / Под ред. Ф. Хейес-Рота, Д. Уотермена, Д. Лената. – М.: Мир, 1987. – 441 с.

142. Представление и использование знаний / Под ред. Х. Уэно, М. Исидзука. – М.: Мир, 1989. – 220 с.

143. Пригожев А. С. Особенности разработки и применения планирующей экспертной системы для обучения // Искусственный интеллект. – 2005. – № 3. – С. 529–540.

144. Прикладные нечеткие системы / Асаи К., Ватада Д., Иваи С. и др. / Под ред. Т. Тэрано, К. Асаи, М. Сугено.– М.: Мир, 1993. – 368 с.

145. Приобретение знаний / Под ред. С. Осуги, Ю. Саэки. – М.: Мир, 1990. – 304 с.

146. Прокопчук Ю. А. Разработка структуры базы знаний медицинской интеллектуальной системы на основе формализма // Искусственный интеллект. – 2006. – № 4. – С. 469–474.

147. Протасов В. И. Генерация новых знаний сетевым человекомашинным интеллектом. Постановка проблемы. // Нейрокомпьютеры: разработка, применение. – 2001. – № 7–8. – С. 94–103.

148. Рассел С., Норвиг П. Искусственный интеллект: современный подход, 2-е и зд.: Пер с англ. – М. – Вильямс, 2006. – 1408 с.

149. Рідкокаша А. А., Голдер К. К. Основи систем штучного інтелекту. Навчальний посібник. – Черкаси: «ВІДЛУННЯ-ПЛЮС», 2002. – 240 с.

150. Робототехника и гибкие автоматизированные производства. В 9-ти кн. Кн. 6. Техническая имитация интеллекта: Учебное пособие для втузов / В. М. Назаретов, Д. П. Ким; Под ред. И. М. Макарова. – М.: Высшая школа, 1986. – 144 с.

151. Ротштейн А. П. Интеллектуальные технологии идентификации: нечеткая логика, генетические алгоритмы, нейронные сети. – Винница: УНИВЕРСУМ-Винница, 1999. – 320 с.

152. Ротштейн А. П. Медицинская диагностика на нечеткой логике. – Винница: Континент-ПРИМ, 1996. – 132 с.

153. Ротштейн А. П., Кательников Д. И. Идентификация нелинейных объектов нечёткими базами знаний // Кибернетика и системный анализ. – 1998. – №5. – С. 53–61.

154. Ротштейн А. П., Лойко Е. Е., Кательников Д. И. Прогнозирование количества заболеваний на основе экспертно-лингвистической информации // Кибернетика и системный анализ. – 1999. – №2. – С. 178–185.

155. Ротштейн А. П., Митюшкин Ю. И. Извлечение нечетких баз знаний из экспериментальных данных с помощью генетических алгоритмов // Кибернетика и системный анализ. – 2001. – № 4. – С. 45–53.

156. Ротштейн А. П., Познер М., Ракитянская А. Б. Прогнозирование результатов футбольных матчей на основе нечеткой модели с генетико-нейронной настройкой // Кибернетика и системный анализ. – 2005. – № 4. – С. 171–184.

157. Ротштейн А. П., Ракитянская А. Б. Решение задачи диагностики на основе нечетких отношений и генетического алгоритма // Кибернетика и системный анализ. – 2001. – № 6. – С. 162–170.

158. Ротштейн А. П., Штовба С. Д. Влияние методов дефаззификации на скорость настройки нечеткой модели // Кибернетика и системный анализ. – 2002. – № 5. – С. 169–176.

159. Ротштейн А. П., Штовба С. Д. Идентификация нелинейной зависимости нечеткой базой знаний с нечеткой обучающей выборкой // Кибернетика и системный анализ. – 2006. – № 2. – С. 17–24.

160. Ротштейн А. П., Штовба С. Д. Нечеткая надежность алгоритмических процессов. – Винница: Континент-ПРИМ, 1997. – 142 с.

161. Ротштейн А. П., Штовба С. Д. Прогнозирование надежности алгоритмических процессов при нечетких исходных данных // Кибернетика и системный анализ. – 1998. – № 4. – С. 85–93.

162. Ротштейн О. П., Мітюшкин Ю. П. Нейро-лінгвістична ідентифікація нелінійних залежностей // Вимірювальна та обчислювальна техніка в технологічних процесах . –1998. – № 4. – С. 5–12.

163. Ротштейн О. П., Ракитянська Г. Б. Діагностика на базі нечітких відношень в умовах невизначеності. – Вінниця: Універсум-Вінниця, 2006. – 275 с.

164. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер с польск. – М.: Горячая линия-Телеком, 2004. – 452 с.

165. Сергиенко И. В., Гупал А. М. Принципы построения процедур индуктивного вывода // Кибернетика и системный анализ. – 2006. – № 4. – С. 51–63.

166. Сергиенко И. В., Парасюк И. Н., Каспшицкая М. Ф. Об одной нечеткой задаче многопараметрического выбора оптимальных решений // Кибернетика и системный анализ. – 2003. – № 2. – С. 3–15.

167. Сетлак Г. Интеллектуальная система поддержки решений в нечеткой среде // Искусственный интеллект. – 2002. – № 3. – С. 428–438.

168. Сойер Б., Фостер Д. Л. Программирование экспертных систем на Паскале: Пер с англ. – М.: Финансы и статистика, 1990. – 191 с.

169. Стиранка А. И. Решение кластерной задачи большой размерности в нечеткой постановке. – Кибернетика. – 1991. – № 1. – С. 116–121.

170. Субботін С. О. Алгоритми кластер-регресійної апроксимації та їх нейромережеві інтерпретації // Радіоелектроніка. Інформатика. Управління. – 2003. – № 1. – С. 114–121.

171. Субботін С. О., Богуслаєв О. В. Методи та алгоритми синтезу нейромережевих діагностичних і прогнозуючих моделей // Вісник двигунобудування. – 2005. – № 1. – С. 145–149.

172. Субботин С. А. Синтез распознающих нейро-нечетких моделей с учетом информативности признаков // Нейрокомпьютеры: разработка, применение. – 2006. – № 10. – С. 50–56.

173. Субботин С. А. Метод формирования баз знаний для нейро-нечетких моделей // Нейроинформатика и ее приложения: Материалы XIV Всероссийского семинара, 6–8 октября 2006 г. / Под ред. А. Н. Горбаня, Е. М. Миркеса. Отв. За выпуск Г. М. Садовская. – Красноярск: ИВМ СО РАН, 2006. – С. 116–118.

174. Субботин С. А. Методы синтеза нейро-нечетких классификаторов для случая нескольких классов // Информационные технологии. – 2006. – № 11. – С. 31–36.

175. Субботин С. А. Неитеративный синтез и редукция нейро-нечетких моделей // Искусственный интеллект. – 2006. – № 3. – С. 323–330.

176. Субботин С. А. Подсистема моделирования семантических сетей // Материалы IX Всероссийского семинара «Моделирование неравновесных систем-2006», 13–15 октября 2006 г. / Под ред. В. В. Слабко. Отв.за выпуск М. Ю. Сенашова. – Красноярск: ИВМ СО РАН, 2006. – С. 172–174.

177. Субботин С. А. Синтез вейвлет-нейро-нечетких моделей для диагностики деталей авиадвигателей // Вісник двигунобудування. – 2006. – № 2. – С. 163–168.

178. Субботин С. А. Синтез нейро-нечетких моделей для выделения и распознавания объектов на сложном фоне по двумерному изображению // Комп'ютерне моделювання та інтелектуальні системи: Збірник наукових праць / За ред. Д. М. Пізи, С. О. Субботіна. – Запоріжжя: ЗНТУ, 2007. – С. 68–91.

179. Субботин С. А. Синтез нейро-нечетких сетей с группировкой признаков // Программные продукты и системы. – 2006. – № 4. – С. 3–7.

180. Субботин С. А., Кирсанова Е. В. Синтез многослойной нейросети на основе кластер-регрессионной аппроксимации в задаче моделирования показателя здоровья детей // Нейроинформатика и ее приложения: Материалы XII Всероссийского семинара, 1–3 октября 2004 г. / Под ред. А. Н. Горбаня, Е. М. Миркеса. Отв. за выпуск Г. М. Садовская. – Красноярск: ИВМ СО РАН, 2004. – С. 136–137.

181. Субботин С. А., Клименко В. А. Модуль поддержки принятия решений библиотеки «Diaglab» // Моделирование неравновесных систем // Материалы VIII Всероссийского семинара, 14–16 октября 2005 г. / Под ред. В. В. Слабко. Отв. за выпуск М. Ю. Сенашова. – Красноярск: ИВМ СО РАН, 2005. – С. 184–185.

182. Тарков М. С. Нейрокомпьютерные системы. Учебное пособие М.: Интуит, 2006. – 142 с.

183. Таунсенд К., Фохт Д. Проектирование и программная реализация экспертных систем на персональных ЭВМ. – М.: Финансы и статистика, 1990. – 320 с.

184. Тимофеев А. В. Роботы и искусственный интеллект. – М.: Наука, 1978. – 192 с.

185. Ткач С. Н. Настройка параметров адаптивного контроллера с использованием нечеткой нейронной сети // Программные продукты и системы. – 2001. – № 3. – С. 9–16.

186. Тоценко В. Г., Егорова Е. А. Метод поддержки принятия решений в условиях нечеткости структуры базы знаний // Электронное моделирование. – 2005. – Т. 27, № 6. – С. 53–61.

187. Уотермен Д. Руководство по экспертным системам / Пер. с англ. В. Л. Стефанюка. – М.: Мир, 1989. – 388 с.

188. Усков А. А., Круглов В. В. Нечеткий дополняющий алгоритм идентификации динамических объектов // Информационные технологии. – 2003. – № 3. – С. 32–34.

189. Усков А. А., Кузьмин А. В. Интеллектуальные технологии управления. Искусственные нейронные сети и нечеткая логика. – М.: Горячая линия-Телеком, 2004. – 143 с.

190. Хант Э. Искусственный интеллект / под ред. Стефанюка. – М.: Мир, 1978. – 558 с.

191. Черноруцкий И. Г. Методы принятия решений. – СПб.: БХВ-Петербург, 2005. – 416 с.

192. Чубукова И. А. Data Mining. Учебное пособие. – М.: Интуит, 2006. – 382 с.

193. Шапцев В. А., Новицкая Е. Н. Решение проблемы экспертных знаний семейными экспертными системами // Искусственный интеллект. – 2006. – № 2. – С. 423–425.

194. Шахиди А. Apriori – масштабируемый алгоритм поиска ассоциативных правил [Электронный ресурс]. – М.: Basegroup, 2002. – Режим доступа: http://www.basegroup.ru/rules/apriori.htm.

195. Шахиди А. Введение в анализ ассоциативных правил [Электронный ресурс]. – М.: Basegroup, 2002. – Режим доступа: http://www.basegroup.ru/rules/intro.htm.

196. Шеннон К. Работы по теории информации и кибернетике. – М.: Иностранная литература, 1963. – 830 с.

197. Штовба С. Д. Нечеткая идентификация на основе регрессионных моделей параметрической функции принадлежности // Проблемы управления и информатики. – 2006. – № 6. – С. 38–44.

198. Штовба С. Д. Обучение нечеткой базы знаний по выборке нечетких данных // Искусственный интеллект. – 2006. – № 4. – С. 560–570.

199. Шушура А. Н., Аниканов В. С. Кластеризация данных с использованием нечетких отношений // Искусственный интеллект. – 2006. – № 1. – С. 139–145.

200. Шушура А. Н., Кондратьев В. В. Разработка механизма обратного логического вывода на основе теории вопросников // Искусственный интеллект. – 2005. – № 1. – С. 138–144.

201. Экспертные системы. Принципы работы и примеры: Пер. с англ. / А. Брукинг, П. Джонс, Ф. Кокс и др.; Под ред. Р. Форсайта. – М.: Радио и связь, 1987. – 224 с.

202. Элти Дж., Кубмс М. Экспертные системы: концепции и примеры. – М.: Финансы и статистика, 1987. – 191 с.

203. Яковлев В. Л., Яковлева Г. Л., Малиевский Д. А. Нейросетевая экспертная система управления портфелем банка // Нейрокомпьютеры: разработка, применение. – 2000. – № 1. – С. 83–85.

204. Ямпольський Л. С., Лавров О. А. Штучний інтелект у плануванні та управлінні виробництвом. – К.: Вища школа, 1995. – 255 с.

205. Ярушкина Н. Г. Основы теории нечетких и гибридных систем. – М.: Финансы и статистика, 2004. – 320 с.

206. Яхъяева Г. Э. Нечеткие множества и нейронные сети. Учебное пособие. – М.: Интуит, 2006. – 316 с.

207. Ященко В. А. Бионический подход к представлению знаний в интеллектуальных системах. I // Кибернетика и системный анализ. – 1998. – №1. – С. 3–26.

АЛФАВІТНО-ПРЕДМЕТНИЙ ПОКАЖЧИК

К

Л

М

Н

О

П

Р

С

Т

У

Ф

Ц, Ч, Ш

Навчальне видання

СУББОТІН Сергій Олександрович

ПОДАННЯ Й ОБРОБКА ЗНАНЬ У СИСТЕМАХ ШТУЧНОГО ІНТЕЛЕКТУ ТА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

Навчальний посібник

Надруковано у авторській редакції

Верстання Рибалка Ірина Сергіївна

Підп. до друку 15.09.2008. Формат 60×84/16. Бум. офс.
Різогр. друк. Ум. друк. арк. 19,8. Обл.-вид. арк. 18.
Тираж 100. Зам. № 1376.

Запорізький національний технічний університет
Україна, 69063 Запоріжжя, вул. Жуковського, 64
Тел.: 8 (0617) 69–82–96, 220–12–14.

Свідоцтво про внесення суб'єкта видавничої справи
до державного реєстру видавців, виготівників
і розповсюджувачів видавничої продукції
від 27.12.2005 р., серія ДК № 2394