МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАСИЛЯ СТУСА

Т. В. Нескородєва, Є. Є. Федоров, Т. В. Січко, А. Р. Нескородєва

ЕКСПЕРТНІ ТА РЕКОМЕНДАЦІЙНІ СИСТЕМИ

Навчальний посібник

Вінниця
2023

УДК 004.891(075.8)
Е457

Рекомендовано до друку Вченою радою ДонНУ імені Василя Стуса
(протокол № 3 від 28 жовтня 2022 р.)

Автори: *Нескородєва Т. В.,* д-р техн. наук, доцент, завідувач кафедри інформаційних технологій Донецького національного університету імені Василя Стуса;
Федоров Є. Є., д-р техн. наук, професор, професор кафедри інформаційних технологій Донецького національного університету імені Василя Стуса;
Січко Т. В., канд. техн. наук, доцент, доцент кафедри інформаційних технологій Донецького національного університету імені Василя Стуса;
Нескородєва А. Р., лаборант міжкафедральної лабораторії машинного навчання та інтелектуального аналізу даних Донецького національного університету імені Василя Стуса.

Рецензенти: *Бондаренко М. О.,* д-р техн. наук, професор, завідувач кафедри приладобудування, мехатроніки та комп'ютерних технологій Черкаського державного технологічного університету;
Шушура О. М., д-р техн. наук, доцент, професор кафедри автоматизації проєктування енергетичних процесів і систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського»;
Штовба С. Д., д-р техн. наук, професор, професор кафедри інформаційних технологій Донецького національного університету імені Василя Стуса.

Т. В. Нескородєва, Є. Є. Федоров, Т. В. Січко, А. Р. Нескородєва

Е457 Експертні та рекомендаційні системи: навч. посіб. для здобувачів вищої освіти спеціальностей 122 «Комп'ютерні науки», 125 «Кібербезпека», 113 «Прикладна математика» / Т. В. Нескородєва, Є. Є. Федоров, Т. В. Січко, А. Р. Нескородєва. Вінниця: ДонНУ імені Василя Стуса, 2023. 224 с.

Навчальний посібник призначений для знайомства з експертними та рекомендаційними системами, що базуються на методах штучного інтелекту, та спрямований на формування і розвиток компетентностей та програмних результатів навчання у здобувачів вищої освіти спеціальностей галузі 122 «Комп'ютерні науки», 125 «Кібербезпека», 113 «Прикладна математика».

УДК 004.891(075.8)

ЗМІСТ

1. КЛАСИФІКАЦІЯ ТА СТРУКТУРА ЕКСПЕРТНИХ СИСТЕМ 6
1.1. Визначення експертних систем 6
1.2. Критерії відбору завдань, які вирішуються експертною системою ..6
1.3. Структура експертних систем 7
1.4. Класифікація експертних систем 8
1.4.1. Класифікація за типом розв'язуваної задачі 8
1.4.2. Класифікація за зв'язками з реальним часом 11
1.4.3. Класифікація за обчислювальною потужністю комп'ютера ..11
1.4.4. Класифікація за ступенем інтеграції з іншими програмами ...12
1.5. Інструментальні засоби побудови експертних систем 12
1.5.1. Традиційні мови програмування 12
1.5.2. Мови штучного інтелекту 12
1.5.3. Спеціальний програмний інструментарій 13
1.5.4. «Оболонки» 13

2. ТЕХНОЛОГІЯ ПРОЄКТУВАННЯ ТА РОЗРОБКИ ЕКСПЕРТНОЇ СИСТЕМИ 14
2.1. Етапи розробки ЕС 14
2.2. Вибір відповідної проблеми 14
2.3. Швидке прототипування 15
2.4. Розвиток прототипу до промислової ЕС 18
2.5. Оцінка системи 19
2.6. Стиковка системи 19
2.7. Підтримка системи 19

3. МОДЕЛІ ПРЕДСТАВЛЕННЯ ДАНИХ ТА ЗНАНЬ 20
3.1. Дані та знання 20
3.2. Моделі представлення даних 21
3.2.1. Ієрархічна модель 21
3.2.2. Мережева модель 22
3.2.3. Реляційна модель 23
3.3. Моделі представлення знань 28
3.3.1. Продукційна модель 29
3.3.2. Семантичні мережі 30
3.3.3. Фрейми 32
3.3.4. Сценарії 34
3.3.5. Формальні логічні моделі 35
3.3.6. Реляційні моделі 36

4. ПОПОВНЕННЯ ЗНАНЬ НА ОСНОВІ ПСЕВДОФІЗИЧНИХ ЛОГІЧНИХ МОДЕЛЕЙ 38
4.1. Логіка часу 38
4.2. Логіка простору 41
4.3. Каузальна логіка 43
4.4. Логіка дій 45

5. ВИВЕДЕННЯ НА ЗНАННЯХ 47
5.1. Виведення на знаннях у вигляді продукційної моделі 47
5.1.1. Машина виведення 47
5.1.2. Стратегії керування виведенням 49
5.1.2.1. Пряме та зворотне виведення 49
5.1.2.2. Методи пошуку в глибину та ширину 51
5.2. Виведення на знаннях у вигляді формальної логічної моделі 52
5.2.1. Принцип резолюції 52
5.2.2. Алгоритм уніфікації 52
5.2.3. Приклади принципу резолюції 54
5.3. Виведення на знаннях у вигляді фреймової моделі 56
5.4. Виведення на знаннях у вигляді моделі семантичної мережі 56

6. ВИВЕДЕННЯ НА НЕЧІТКИХ ЗНАННЯХ 57
6.1. Введення в нечіткі множини та нечітку логіку 57
6.2. Структура нечіткого виведення 65
6.2.1. Формування бази правил 65
6.2.2. Фазифікація 67
6.2.3. Агрегування умов 67
6.2.4. Активізація заключень 68
6.2.5. Агрегування заключень 70
6.2.6. Дефазифікація 70
6.3. Основні алгоритми нечіткого виведення 71

7. ВИВЕДЕННЯ НА НЕТОЧНИХ ЗНАННЯХ 76
7.1. Введення у виведення на неточних знаннях 76
7.2. Байєсівський підхід 77
7.2.1. Механізм виведення в ЕС PROSPECTOR 77
7.2.2. Схема Перла (байєсівські мережі довіри) 82
7.3. Метод коефіцієнтів впевненості 83
7.4. Теорія Демпстера−Шефера (ТДШ) 85
7.5. Механізм виведення INFERNO 88
7.6. Обчислення інцидентів 89

8. ВИЗНАЧЕННЯ, ЦІЛІ, ЗАВДАННЯ, ПРИКЛАДИ ТА КЛАСИФІКАЦІЯ РЕКОМЕНДАЦІЙНИХ СИСТЕМ 90
8.1. Визначення та цілі рекомендаційних систем 90
8.2. Завдання рекомендаційних систем 91
8.3. Приклади рекомендаційних систем 91
8.4. Класифікація рекомендаційних систем 93
8.5. Класифікація та розподіл рейтингів 94

9. ОСНОВНІ МЕТОДИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ 96
9.1. Методи колаборативної фільтрації 96
9.1.1. Методи на основі пам'яті або сусідства 97
9.1.2. Методи на основі моделей класифікаторів або апроксиматорів 101
9.1.3. Методи на основі моделей латентних факторів 122
9.2. Методи контентної фільтрації 128
9.3. Методи на основі знань 131
9.4. Контекстно-залежні методи 133

ЛАБОРАТОРНІ РОБОТИ 135

ЛІТЕРАТУРА 168

ДОДАТКИ 169
ДОДАТОК А. Відношення і правила псевдофізичної логіки 169
ДОДАТОК Б. Основні поняття та функції мови CLIPS 181
ДОДАТОК В. Основні класи модуля CLIPSPY, що використовуються для чіткого виведення 196
ДОДАТОК Г. Основні функції модуля SCIKIT-FUZZY, що використовуються для нечіткого висновку 207
ДОДАТОК Д. Основні класи та функції модуля SURPRISE, що використовуються для обчислення рейтингу 211

1. КЛАСИФІКАЦІЯ ТА СТРУКТУРА ЕКСПЕРТНИХ СИСТЕМ

1.1. Визначення експертних систем

Одним із найбільш поширених видів інтелектуальних систем є експертні системи.

Експертна система (ЕС) – це інтелектуальна система, що акумулює знання експерта в конкретній предметній галузі та здатна на їх основі формувати рішення на рівні експерта.

Тільки у США щорічний дохід від продажів інструментальних засобів розробки ЕС становить 300−400 млн доларів, а від застосування ЕС – 80–90 млн доларів.

Сьогодні 25 % користувачів використовують ЕС, а 25 % мають намір придбати ЕС. Найбільш популярні ЕС у виробництві, бізнесі, медицині.

ЕС характеризуються такими властивостями:

– відкритість (користувач може перевірити рішення, прийняті ЕС, на будь-якому етапі виконання програми, завдяки підсистемі пояснень);

– гнучкість (простота модифікації бази знань у разі декларативних знань);

– недетермінованість рекомендацій (використовуються евристичні методи).

1.2. Критерії відбору завдань, які вирішуються експертною системою

Щоб розробка ЕС була можлива, необхідне виконання таких вимог:

– мають бути експерти, які ефективно вирішують проблеми у цій галузі;

– оцінки правильності рішень, зроблені різними експертами, мають переважно збігатися;

– експерти повинні вміти пояснювати методи, які вони використовують під час вирішення задачі;

– завдання має вирішуватися на основі виведення на знаннях;

– розв’язуване завдання не повинно бути надто складним, а його вирішення має здійснюватися експертом за кілька годин;

– завдання має ставитися до вже добре дослідженої області.

Застосування ЕС для вирішення завдання може бути виправдане такими факторами:

– вирішення завдання створює значний економічний ефект;

– залучення експертів до вирішення завдання неможливе через їх нечисленність та високу оплату праці;

– звільнення персоналу організації призводить до значного зниження рівня компетенції у цій організації;

– необхідність прийняття рішень у шкідливих умовах, що виключають присутність людини.

Використання ЕС доречно, якщо вирішуване завдання має такі характеристики:

– вирішення завдання здійснюється шляхом маніпулювання символьними структурами, а не числами;

– процес вирішення завдання має евристичний характер;

– для підготовки експерта потрібно багато років навчання;

– завдання має становити практичний інтерес і бути вузьким, щоб допускати рішення через виведення на знаннях.

1.3. Структура експертних систем

Узагальнена структура ЕС представлена на рис. 1.1.

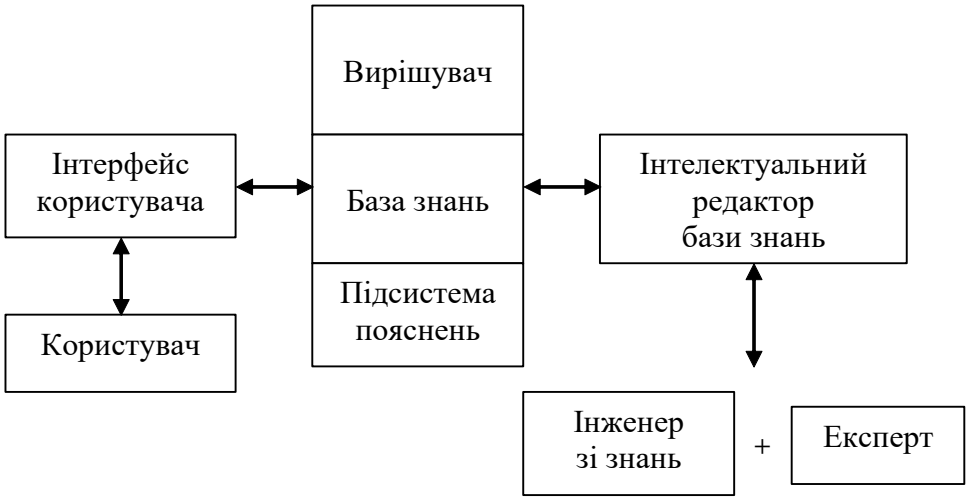


Рис 1.1 − Структура експертної системи

Слід врахувати, що реальні ЕС можуть мати більш складну структуру, однак блоки, зображені на рис. 1.1, неодмінно присутні в будь-якій дійсній ЕС.

Загалом процес функціонування ЕС можна зобразити так: користувач, який хоче отримати необхідну інформацію, через призначений інтерфейс для користувача надсилає запит до ЕС; розв’язувач, користуючись БЗ, генерує та видає користувачу відповідну рекомендацію.

Визначимо основні терміни в області розробки ЕС.

Користувач – фахівець предметної області, для якого призначена система. Зазвичай його кваліфікація недостатньо висока і тому він потребує допомоги з боку ЕС.

Інженер зі знань – фахівець у галузі штучного інтелекту (ШІ), який виступає в ролі проміжного буфера між експертом і БЗ. Синоніми: *когнітолог, інженер-інтерпретатор, аналітик.*

Інтерфейс користувача – комплекс програм, що реалізують діалог користувача з ЕС і на стадії введення інформації, і під час отримання результатів.

База знань (БЗ) – ядро ЕС, сукупність знань предметної області, записана на машинний носій у формі, зрозумілій експерту і користувачу (зазвичай мовою, наближеною до природної). Паралельно існує БЗ у машинному поданні.

Розв'язувач – програма, що моделює перебіг міркувань експерта на підставі знань, наявних у БЗ. Синоніми: *дедуктивна машина, машина логічного виведення, блок логічного виведення*.

Підсистема пояснень – програма, що дає змогу користувачу отримати відповіді на питання: «Як була отримана та чи інша рекомендація?» та «Чому система прийняла таке рішення?». Відповідь на питання «як» − це трасування всього процесу отримання рішення із зазначенням використаних фрагментів БЗ, тобто всіх кроків ланцюга умовиводів. Відповідь на питання «чому?» – посилання на умовивід, – безпосередньо передувала отриманому рішенню, тобто відхід на один крок назад. Розвинені підсистеми пояснень підтримують і інші типи питань.

Інтелектуальний редактор БЗ – програма, що надає інженеру зі знань можливість створювати БЗ у діалоговому режимі. Вміщує в себе систему вкладених меню, шаблонів мови представлення знань, підказок та інших сервісних засобів, що полегшують роботу з базою.

Промислові прикладні ЕС можуть бути істотно складнішими і додатково включати бази даних, інтерфейси обміну даними з різними пакетами прикладних програм, електронними бібліотеками тощо.

1.4. Класифікація експертних систем

Класифікація ЕС наведена на рис. 1.2.

1.4.1. Класифікація за типом розв'язуваної задачі

За завданням ЕС класифікуються:

1. Інтерпретація даних. Це одне з традиційних завдань для ЕС. Під інтерпретацією розуміється процес визначення сенсу даних, результати якого

повинні бути узгодженими і коректними. Зазвичай передбачається багатоваріантний аналіз даних. Наприклад, виявлення та ідентифікація різних типів океанських суден за результатами аерокосмічного сканування; визначення основних властивостей особистості за результатами тестування.

2. Діагностика. Під діагностикою розуміється процес співвідношення об'єкта з деяким класом об'єктів та/або виявлення несправності в деякій системі. Несправність – це відхилення від норми. Важливою специфікою тут є необхідність розуміння функціональної структури («анатомії») системи, що діагностує. Наприклад, діагностика і терапія звуження коронарних судин; діагностика помилок в апаратурі і математичному забезпеченні комп'ютера.

3. Моніторинг. Основне завдання моніторингу – безперервна інтерпретація даних у реальному масштабі часу і сигналізація про вихід тих або інших параметрів за допустимі межі. Головні проблеми – «пропуск» тривожної ситуації та інверсне завдання «помилкового» спрацьовування. Складність цих проблем у розмитості симптомів тривожних ситуацій і необхідність обліку часового контексту. Наприклад, контроль за роботою електростанцій; допомога диспетчерам атомного реактора; контроль аварійних датчиків на хімічному заводі.

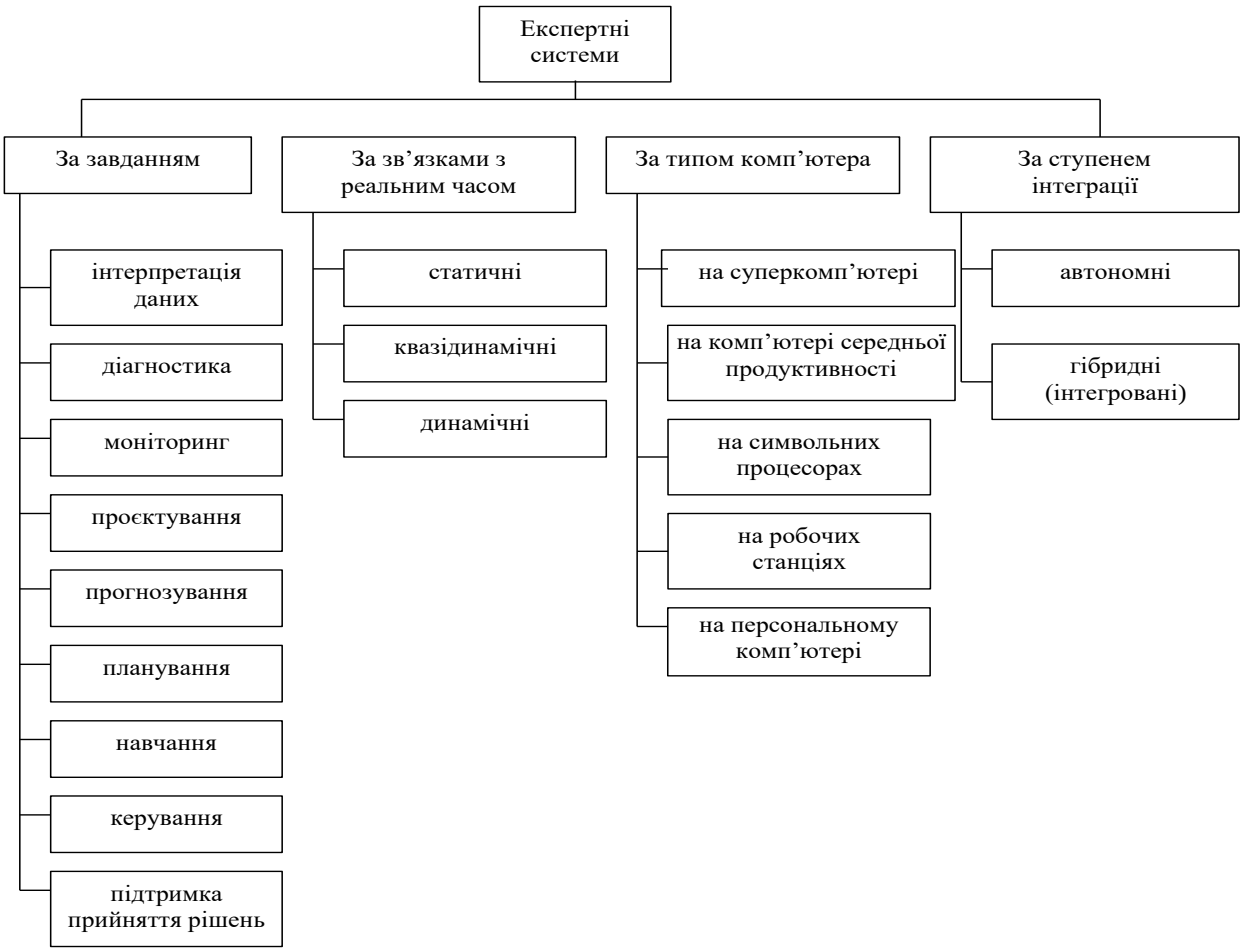


Рис. 1.2 − Класифікація експертних систем

4. Проєктування. Проєктування полягає в підготовці специфікацій на створення «об'єктів» із заздалегідь визначеними властивостями. Під специфікацією розуміється весь набір необхідних документів – креслення, пояснювальна записка та ін. Основні проблеми тут – отримання чіткого структурного опису знань про об'єкт і проблема «сліду». Для організації ефективного проєктування і перепроєктування необхідно формувати не тільки самі проєктні рішення, але і мотиви їх прийняття. Отже, у завданнях проєктування тісно зв'язуються два основні процеси, які виконуються в межах відповідної ЕС: процес логічного виводу рішення і процес пояснення (наприклад, проєктування конфігурації комп'ютера і БІС; синтез електричних мереж).

5. Прогнозування. Прогнозування дає змогу передбачати наслідки деяких подій або явищ на підставі аналізу наявних даних. Системи, що прогнозують, логічно виводять ймовірні наслідки із заданих ситуацій. У прогнозуючій системі зазвичай використовується параметрична динамічна модель, у якій значення параметрів «підганяються» під задану ситуацію. Наслідки, які логічно виведені з цієї моделі, складають основу для прогнозів зі ймовірними оцінками. Наприклад, прогноз погоди; оцінки майбутнього врожаю; прогнози в економіці.

6. Планування. Під плануванням розуміється знаходження планів дій, що належать до об'єктів, здатних виконувати деякі функції. У таких ЕС використовуються моделі поведінки реальних об'єктів для того, щоб логічно вивести наслідки планованої діяльності (наприклад, планування поведінки робота; планування промислових замовлень; планування експерименту).

7. Навчання. Під навчанням розуміється використання комп'ютера для навчання певної дисципліни або предмета. Системи навчання діагностують помилки під час вивчення якої-небудь дисципліни за допомогою комп'ютера і підказують правильні рішення. Вони акумулюють знання про гіпотетичного «учня» та його характерні помилки, потім у роботі вони здатні діагностувати слабкості в пізнаннях учнів і знаходити відповідні засоби для їх ліквідації. До того ж вони планують акт спілкування з учнем залежно від успіхів учня з метою передачі знань (наприклад, вивчення мови програмування LISP; вивчення мови Паскаль).

8. Керування. Під керуванням розуміється функція організованої системи, що підтримує певний режим діяльності. Подібні ЕС здійснюють керування поведінкою складних систем відповідно до заданих специфікацій (наприклад, допомога в керуванні газовою котельнею; керування системою календарного планування).

9. Підтримка прийняття рішень. Підтримка прийняття рішення – це сукупність процедур, що забезпечує особу, яка приймає рішення, необхідною

інформацією та рекомендаціями, що полегшують процес прийняття рішення. Ці ЕС допомагають фахівцям вибрати та/або сформувати потрібну альтернативу серед множини виборів під час прийняття відповідальних рішень (наприклад, вибір стратегії виходу фірми з кризової ситуації; допомога у виборі страхової компанії або інвестора).

Загалом усі системи, засновані на знаннях, можна поділити на *системи, які вирішують завдання аналізу,* і *системи, які вирішують завдання синтезу.* Основна відмінність завдань аналізу від завдань синтезу полягає в тому, що якщо у завданнях аналізу множина рішень може бути перерахованою і включеною в систему, то в завданнях синтезу множина рішень потенційно необмежена і будується з рішень компонент або підпроблем. Завданнями аналізу є: інтерпретація даних, діагностика, підтримка ухвалення рішення; до завдань синтезу належать проєктування, планування, керування. Комбіновані: навчання, моніторинг, прогнозування.

1.4.2. Класифікація за зв'язками з реальним часом

1. Статичні ЕС розробляються у предметних областях, у яких БЗ та інтерпретовані дані не змінюються в часі. Вони стабільні (наприклад, діагностика несправностей в автомобілі).

2. Квазідінамічні ЕС інтерпретують ситуацію, яка змінюється з деяким фіксованим інтервалом часу (наприклад, мікробіологічні ЕС, в яких знімаються лабораторні вимірювання з технологічного процесу один раз на 4–5 годин і аналізується динаміка отриманих показників щодо попереднього виміру).

3. Динамічні ЕС працюють у сполученні з датчиками об'єктів у режимі реального часу і з безперервною інтерпретацією надходять у систему даних (наприклад, керування гнучкими виробничими комплексами, моніторинг у реанімаційних палатах; програмний інструментарій для розробки динамічних систем).

1.4.3. Класифікація за обчислювальною потужністю комп'ютера

1) ЕС для унікальних стратегічно важливих завдань на суперкомп'ютері;
2) ЕС на комп'ютері середньої продуктивності;
3) ЕС на символьних процесорах і робочих станціях;
4) ЕС на міні- і супермінікомп'ютері;
5) ЕС на персональних комп'ютерах.

1.4.4. Класифікація за ступенем інтеграції з іншими програмами

1. Автономні ЕС працюють безпосередньо в режимі консультацій із користувачем для специфічно «експертних» завдань, для вирішення яких не потрібно залучати традиційні методи обробки даних (розрахунки, моделювання).

2. Гібридні ЕС являють програмний комплекс, який агрегує стандартні пакети прикладних програм (ППП) (наприклад, математичну статистику, лінійне програмування або системи керування БД) і засоби маніпулювання знаннями. Це може бути інтелектуальна надбудова над ППП або інтегроване середовище для вирішення складного завдання з елементами експертних знань.

Незважаючи на зовнішню привабливість гібридного підходу, необхідно зазначити, що розробка таких систем є складнішою задачою, ніж розробка автономної ЕС.

1.5. Інструментальні засоби побудови експертних систем

До інструментальних засобів належать традиційні мови програмування, мови штучного інтелекту, спеціальний програмний інструментарій, «оболонки».

1.5.1. Традиційні мови програмування

У цю групу інструментальних засобів входять традиційні мови програмування (C, C++, Basic, SmallTalk, Fortran і т. д.), орієнтовані здебільшого на чисельні алгоритми і слабо підходять для роботи з символьними і логічними даними. Тому створення систем штучного інтелекту на основі цих мов вимагає великої роботи програмістів. Однак великою перевагою цих мов є висока ефективність, пов’язана з їх близькістю до традиційної машинної архітектури. До того ж використання традиційних мов програмування дає змогу вміщати інтелектуальні підсистеми (наприклад, інтегровані експертні системи) у великі програмні комплекси загального призначення. Серед традиційних мов найбільш зручними вважаються об’єктно-орієнтовані (C++, C#, Java, Python). Це пов’язано з тим, що парадигма об’єктно-орієнтованого програмування тісно пов’язана з фреймовою моделлю представлення знань. До того ж традиційні мови програмування використовуються для створення інших класів інструментальних засобів штучного інтелекту.

1.5.2. Мови штучного інтелекту

Це насамперед ЛІСП (LISP) і Пролог (Prolog) − найбільш поширені мови, призначені для вирішення завдань штучного інтелекту. Універсальність цих мов менша, ніж традиційних, але втрату мови штучного інтелекту компенсують

розширеними можливостями роботи з символьними і логічними даними, що вкрай важливо для задач штучного інтелекту. На основі мов штучного інтелекту створюються спеціалізовані комп'ютери (наприклад, Лісп-машини), призначені для вирішення завдань штучного інтелекту. Недолік цих мов – непридатність для створення гібридних експертних систем.

1.5.3. Спеціальний програмний інструментарій

У цю групу програмних засобів штучного інтелекту входять спеціальні інструментарії загального призначення. Зазвичай це бібліотеки та надбудови над мовою штучного інтелекту Лісп: KEE (Knowledge Engineering Environment), FRL (Frame Representation Language), KRL (Knowledge Represantation Language), ARTS та інші, що дають змогу користувачам працювати із заготовками експертних систем на більш високому рівні, ніж це можливо у звичайних мовах штучного інтелекту. Також використовуються спеціальні модулі, орієнтовані на експертні системи CLIPS.

1.5.4. «Оболонки»

Під «оболонками» (shells) розуміють «порожні» версії наявних експертних систем, тобто готові експертні системи без бази знань. Прикладом такої оболонки може бути експертна система в галузі медицини EMYCIN (Empty MYCIN – порожній MYCIN), яка являє собою незаповнену експертну систему MYCIN. Перевага оболонок у тому, що вони зовсім не вимагають роботи програмістів для створення готової експертної системи. Потрібні тільки фахівці у предметній області для заповнення бази знань. Однак якщо деяка предметна область погано вкладається в модель, яка використовується в деякій оболонці, заповнити базу знань у цьому разі досить непросто.

2. ТЕХНОЛОГІЯ ПРОЄКТУВАННЯ ТА РОЗРОБКИ ЕКСПЕРТНОЇ СИСТЕМИ

2.1. Етапи розробки ЕС

Процес розробки промислової ЕС з опертям на традиційні технології практично для будь-якої предметної області можна розділити на шість послідовних етапів.

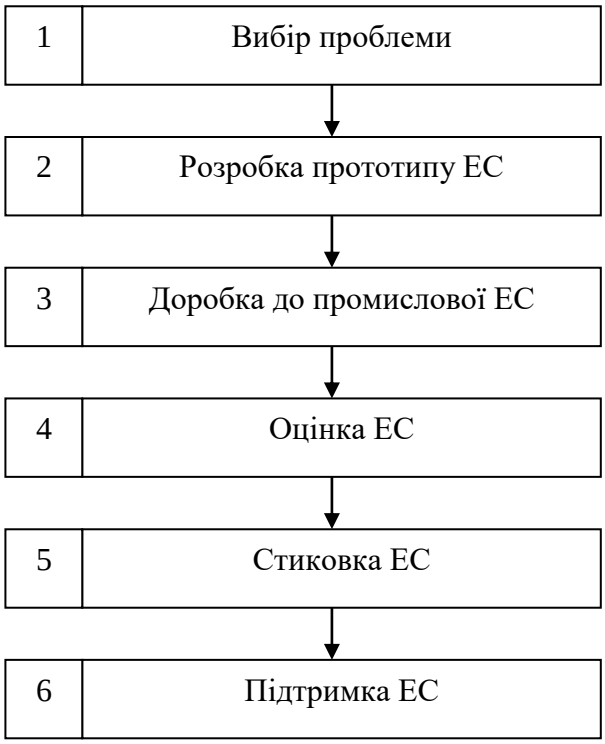


Рис 2.1 – Етапи розробки ЕС

2.2. Вибір відповідної проблеми

Цей етап визначає діяльність, що передує рішенню почати розробляти конкретну ЕС. Він вміщує такі дії інженера зі знань:

- визначення проблемної області та завдання;
- знаходження експерта, який хоче співпрацювати для вирішення проблеми, і призначення колективу розробників;
- визначення попереднього підходу до вирішення проблеми;
- аналіз витрат і прибутків від розробки;
- підготовка докладного плану розробки.

2.3. Швидке прототипування

Прототипування системи є усіченою версією ЕС, спроєктованою для перевірки правильності кодування фактів, зв'язків і стратегій міркування експерта. Вона також дає можливість інженеру зі знань залучити експерта до активної участі в процесі розробки експертної системи і до прийняття ним зобов'язання докласти всіх зусиль для створення системи в повному обсязі.

Обсяг прототипу – кілька десятків знань, представлених правилами, фреймами та ін. На рис. 2.2 зображено шість стадій розробки прототипу і мінімальний колектив розробників, зайнятих на кожній стадії.

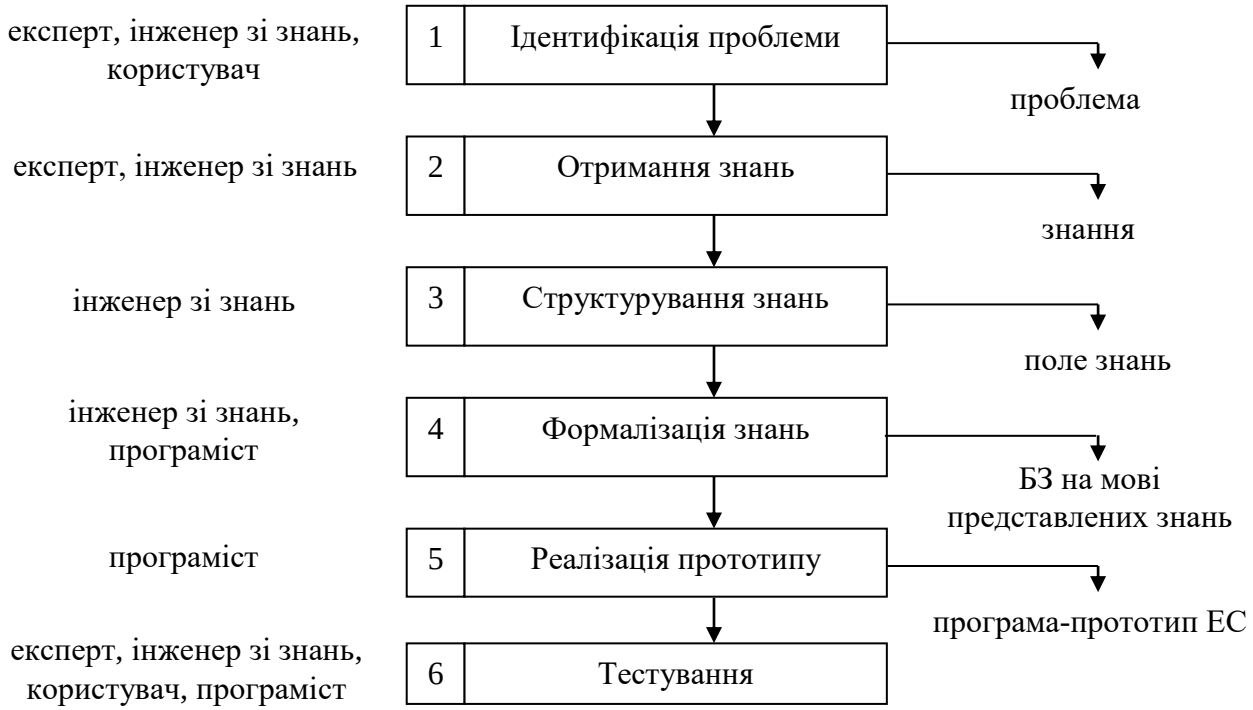


Рис. 2.2 – Стадії розробки прототипу ЕС

Ідентифікація проблеми – це знайомство і навчання членів колективу розробників, а також створення неформального формулювання проблеми.

На цій стадії ускладнюється завдання, планується перебіг розробки ЕС, визначаються:

- необхідні ресурси (час, люди, комп'ютери та ін.);
- джерела знань (книги, додаткові експерти, методики);
- наявні аналогічні ЕС;
- цілі (поширення досвіду, автоматизація рутинних дій та ін.);
- класи розв'язуваних завдань.

Середня тривалість – 1–2 тижні.

Видобуток знань – це отримання інженером зі знань найбільш повного з можливих уявлень про предметну область і способи прийняття рішення в ній.

На цій стадії відбувається перенесення компетентності від експерта до інженера зі знань із використанням різних методів (рис. 2.3).

Під час спостереження просто фіксуються дії експерта; під час протоколу «думок вголос» вони коментуються; під час «мозкового штурму» висловлюються будь-які ідеї; під час круглого столу відбувається дискусія; під час інтерв'ю, на відміну від анкетування, частина питань може змінюватися залежно від ситуації; під час діалогу здійснюється вільна бесіда між експертом та інженером зі знань.

Середня тривалість – 1–3 місяці.

Структурування (концептуалізація) знань – це розробка неформального опису знань про предметну область у вигляді графа, таблиці, діаграми або тексту, який відображає основні концепції і взаємозв'язки між поняттями предметної області.

На цій стадії виявляється структура отриманих знань про предметну область, тобто визначаються:

- термінологія;
- список основних понять та їх властивостей;
- відношення між поняттями;
- структура вхідної та вихідної інформації;
- стратегія прийняття рішень;
- обмеження стратегій.

Такий опис називається полем знань. Середня тривалість – 2–4 тижні.

Формалізація знань – це розробка бази знань мовою представлення знань, яка, з одного боку, відповідає структурі поля знань, а з іншого – дає змогу реалізувати прототип системи на наступній стадії програмної реалізації.

На цій стадії будується формалізоване уявлення концепцій предметної області на основі вибраної мови представлення знань. Традиційно на цьому етапі використовуються:

- продукційні моделі;
- семантичні мережі;
- фрейми;
- сценарії;
- формальні логічні моделі;
- реляційні моделі.

Усе частіше на цій стадії використовується симбіоз мов представлення знань. Середня тривалість – 1–2 місяці.

Реалізація – це розробка програмного комплексу, який демонструє життєздатність підходу загалом. Найчастіше перший прототип відкидається на етапі реалізації діючої ЕС.

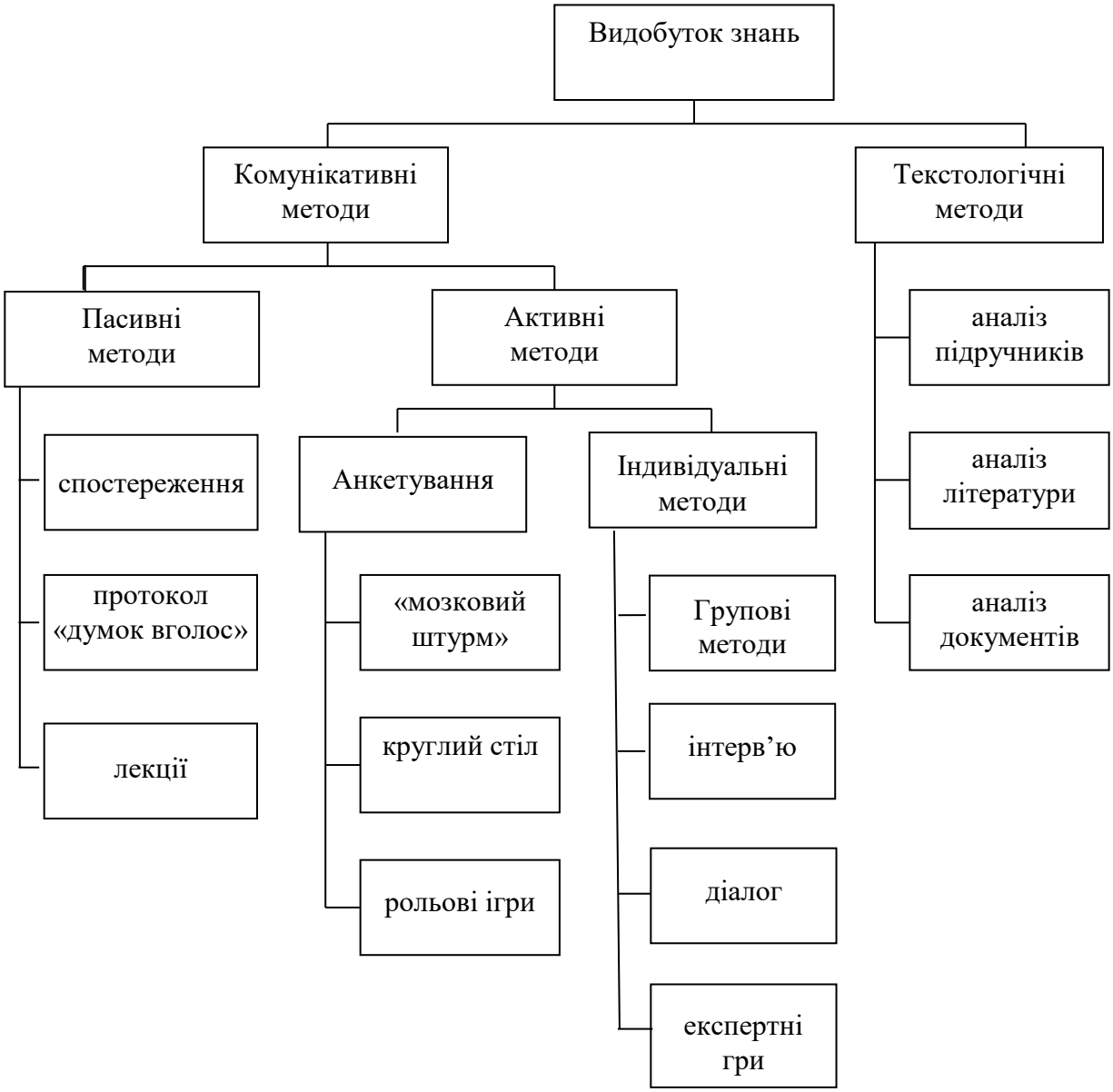


Рис. 2.3 – Класифікація методів видобутку знань

На цій стадії створюється прототип ЕС, що вміщує базу знань та інші блоки за допомогою одного з таких способів:

- програмування традиційними мовами типу С++;
- програмування на спеціалізованих мовах, застосовуваних у задачах штучного інтелекту, типу LISP, FRL, SMALLTALK;
- використання інструментальних засобів розробки ЕС типу СПЕІС, ПІЕС, G 2.

Середня тривалість – 1–2 місяці.

Тестування – це виявлення помилок у підході та реалізації прототипу і вироблення рекомендацій із доведення системи до промислового варіанта.

На цій стадії оцінюється та перевіряється робота програм прототипу з метою приведення у відповідність із реальними запитами користувачів. Прототип перевіряється на:

- зручність і адекватність інтерфейсів вводу / виводу (характер питань у діалозі, зв'язність виведеного тексту, результату та ін.);
- ефективність стратегії керування (порядок перебору, використання нечіткого виведення та ін.);
- кількість перевірочних прикладів;
- коректність бази знань (повнота та несуперечність правил).

Середня тривалість – 2–4 тижні.

2.4. Розвиток прототипу до промислової ЕС

У разі незадовільного функціонування прототипу експерт та інженер зі знань мають можливість оцінити, що саме буде вміщено в розробку остаточного варіанта ЕС. Якщо спочатку вибрані об'єкти або властивості виявляються невідповідними, їх необхідно змінити. Можна зробити оцінку загальної кількості евристичних правил, необхідних для створення остаточного варіанта ЕС. Етапи переходу від прототипу до промислової ЕС представлені в табл. 2.1.

Основна робота полягає в істотному розширенні бази знань, тобто збільшенні глибини ЕС. Після встановлення основної структури ЕС інженер зі знань приступає до розробки та адаптації інтерфейсів, за допомогою яких система буде спілкуватися з користувачем та експертом. На цьому етапі експерти можуть самі вводити в систему нові правила.

Таблиця 2.1

Перехід від прототипу до промислової ЕС

Система	Опис
Демонстраційний прототип ЕС	Система вирішує частину завдань, демонструючи життєздатність підходу
Дослідницький прототип ЕС	Система вирішує більшість завдань, але нестійка в роботі і не повністю перевірена
Діючий прототип ЕС	Система надійно вирішує всі завдання на реальних прикладах, але для складного завдання вимагає багато часу і пам'яті
Промислова ЕС	Система забезпечує високу якість рішень за умови мінімізації необхідного часу і пам'яті; реалізується з використанням більш ефективних засобів представлення знань
Комерційна ЕС	Промислова система придатна до продажу, тобто добре документована і забезпечена сервісом

2.5. Оцінка системи

Після завершення етапу розробки промислової ЕС необхідно провести її тестування щодо критеріїв ефективності. Оцінка здійснюється відповідно до таких критеріїв:

- критерії користувачів (зрозумілість роботи системи, зручність інтерфейсів та ін.);
- критерії запрошених експертів;
- критерії колективу розробників (ефективність реалізації; продуктивність; час відгуку; дизайн; широта охоплення предметної області; несуперечливість БЗ; кількість тупикових ситуацій, коли система не може прийняти рішення; аналіз чутливості програми до незначних змін в уявленні знань і вагових коефіцієнтів, що застосовуються в механізмах логічного виведення та ін.).

2.6. Стиковка системи

На цьому етапі здійснюється стикування ЕС з іншими програмними засобами в середовищі, в якому вона буде працювати, і навчання людей, яких вона буде обслуговувати. Іноді це вимагає істотних змін.

Коли ЕС готова, інженер зі знань повинен переконатися, що експерти та користувачі можуть з нею працювати. Стиковка вміщує в себе забезпечення зв’язку ЕС із наявними базами даних та іншими системами на підприємстві, а також за необхідності з вимірювальними приладами.

2.7. Підтримка системи

Під час перекодування системи на мову, подібну С, підвищується її швидкодія і збільшується переносимість, але гнучкість водночас зменшується. Це прийнятно, якщо знання системи не будуть змінюватися в найближчому майбутньому.

3. МОДЕЛІ ПРЕДСТАВЛЕННЯ ДАНИХ ТА ЗНАНЬ

3.1. Дані та знання

Під час вивчення ЕС традиційно виникає питання, що таке знання і чим вони відрізняються від звичайних даних, десятиліттями оброблюваних комп'ютером.

Дані – це окремі факти, що характеризують об'єкти, процеси і явища предметної області, а також їх властивості. Під час обробки на комп'ютері дані трансформуються, умовно проходячи такі етапи:

1) D 1 – дані як результат вимірювань і спостережень;

2) D 2 – дані на матеріальних носіях інформації (таблиці, протоколи, довідники);

3) D 3 – моделі (структури) даних у вигляді діаграм, графіків, функцій;

4) D 4 – дані в комп'ютері мовою опису даних;

5) D 5 – БД на машинних носіях інформації.

Знання засновані на даних, отриманих емпіричним шляхом. Вони є результатом розумової діяльності людини, спрямованої на узагальнення її досвіду, отриманого внаслідок практичної діяльності.

Знання – це інформація, оброблена (відображена) свідомістю людини.

У процесі обробки на комп'ютері знання трансформуються аналогічно даним:

1) Z 1 – знання в пам'яті людини як результат мислення,

2) Z 2 – матеріальні носії знань (підручники, методичні посібники),

3) Z 3 – *поле знань* – умовний опис основних об'єктів предметної області, їх атрибутів і закономірностей, які їх зв'язують,

4) Z 4 – знання, описані на мовах подання знань (продукційні мови, семантичні мережі, фрейми),

5) Z 5 – БЗ на машинних носіях інформації.

Для зберігання даних використовуються БД.

База даних (БД) – це поіменована сукупність структурованих даних, що належать до певної предметної області.

Для зберігання знань використовуються БЗ. БЗ – основа будь-якої ЕС.

База знань (БЗ) – сукупність знань предметної області, записана на машинний носій у формі, яка є зрозумілою користувачу (зазвичай мовою, наближеною до природної). Паралельно існує БЗ у машинному поданні.

3.2. Моделі представлення даних

Зовнішня та концептуальна (але не внутрішня) моделі БД ІС можуть бути ієрархічними, мережними або реляційними моделями (будуть розглянуті в наступних розділах).

3.2.1. Ієрархічна модель

Ієрархічна модель являє собою деревоподібну структуру. Деревоподібна структура (дерево) – це зв'язний орієнтований граф, який не містить циклів (рис. 3.1). «Зв'язний» означає, що всі вершини у графі з'єднані дугами.

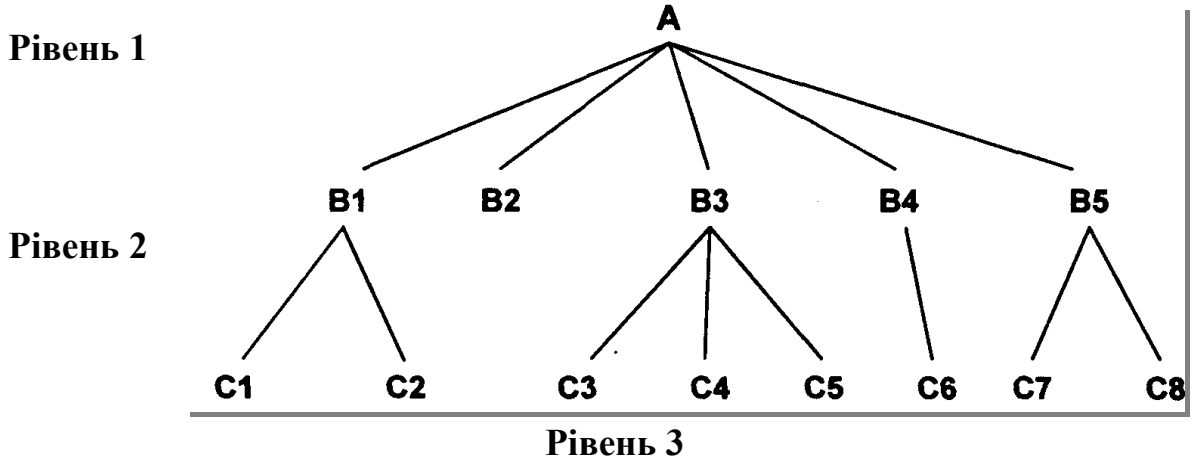


Рис. 3.1 – Графічне зображення ієрархічної моделі

До основних понять ієрархічної структури належать: рівень, елемент (вузол), зв'язок. Вузол (вершина) використовується для опису властивостей об'єкта, а дуга – для опису зв'язків між ними. На схемі ієрархічного дерева вузли представляються вершинами графа. Кожен вузол на більш низькому рівні пов'язаний лише з одним вузлом, що знаходиться на більш високому рівні, який називається вихідним. Ієрархічне дерево має тільки одну вершину (корінь дерева), що не підпорядкована ніякий іншій вершині і знаходиться на самому верхньому (першому) рівні. Підлеглі (породжені) вузли знаходяться на другому, третьому та ін. рівнях.

До кожного підлеглого вузла існує тільки один ієрархічний шлях від кореневого вузла. Наприклад, як бачимо з рис. 3.1, для вузла С4 шлях проходить через вузол А та В3.

Наприклад, на рис. 3.2 представлена ієрархічна модель для студентів.

Рис. 3.2 – Приклад ієрархічної моделі

Переваги ієрархічної моделі:

1. Простота розуміння і використання моделі.
2. Забезпечення певного рівня незалежності даних.
3. Простота оцінки операційних характеристик завдяки заздалегідь заданим взаємозв'язкам.
4. Наявність успішних СУБД, що використовують цю модель.

Недоліки ієрархічної моделі:

1. Взаємозв'язки «багато до багатьох» реалізуються штучно, що призводить до громіздких структур і надмірності даних.
2. Через сувору ієрархічну впорядкованість об'єктів моделі значно ускладняються операції включення та видалення з моделі.
3. Видалення вихідних об'єктів тягне видалення породжених об'єктів.
4. Процедурна мова маніпулювання даними.
5. Доступ до будь-якого залежного вузла можливий тільки через кореневий вузол.

3.2.2. Мережева модель

Мережева модель також базується на графовій формі представлення. Але, на відміну від ієрархічної моделі, породжений вузол може мати більше одного вихідного вузла (рис. 3.3).

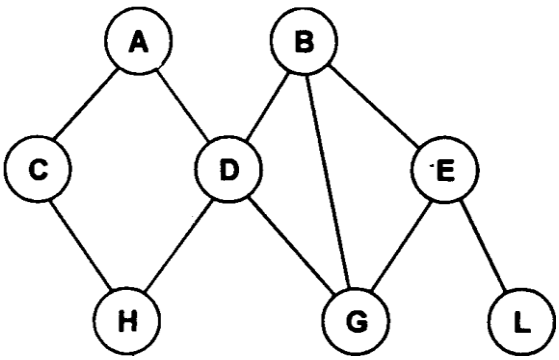


Рис. 3.3 – Графічне зображення мережевої моделі

Наприклад, на рис. 3.4 представлена мережева модель.

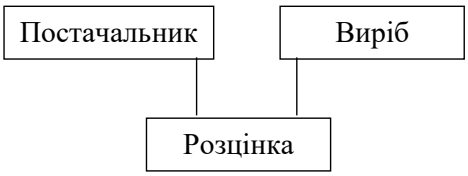


Рис. 3.4 – Приклад мережевої моделі

Цикли (наприклад, на рис. 3.3) і петлі в мережевій моделі можна не допустити шляхом введення надмірності. Уведена надмірність стосується не всього логічного запису, а тільки його ключового елемента даних.

Переваги мережевої моделі:

1. Простота реалізації взаємозв'язку «багато до багатьох».
2. Наявність успішних СУБД, що використовують цю модель.

Недоліки мережевої моделі:

1. Складність моделі (складність навігації (проходження зв'язками) в БД).
2. Втрата незалежності даних під час реорганізації БД.
3. Складність написання прикладних програм.

3.2.3. Реляційна модель

Поняття реляційний (англ. *relation* – відношення) пов'язане з розробками відомого американського фахівця в області БД і співробітника фірми IBM Кодда.

Ці моделі характеризуються простотою структури даних, зручним для користувача табличним представленням і можливістю використання формального апарату реляційної алгебри і реляційного числення для обробки даних. Це найпоширеніша модель.

Реляційна модель орієнтована на організацію даних у вигляді реляційних таблиць. Кожна реляційна таблиця являє собою прямокутну таблицю і має такі властивості:

- кожна таблиця має унікальне ім’я (відповідає об’єкту предметної області);
- кожен елемент таблиці – один елемент даних (конкретне значення властивості об’єкта предметної області);
- усі стовпці в таблиці однорідні, тобто всі елементи у стовпці мають однаковий тип (числовий, символьний і под.) і довжину;
- кожен стовпець має унікальне ім’я (відповідає властивості об’єкта предметної області);
- однакові рядки в таблиці відсутні;
- порядок проходження рядків і стовпців може бути довільним.

Слід зазначити, що зовнішня і концептуальна моделі є реляційними, а внутрішня модель не є реляційною. Для цієї моделі використовується стандартна високорівнева мова SQL, яка виступає як мова маніпулювання даними, мова опису даних і мова запитів.

Наприклад, на рис. 3.5 представлена реляційна модель.

СТУДЕНТ

№ особової справи	Прізвище	Ім’я	По батькові	Дата народження	Група
16493	Сергєєв	Петро	Михайлович	01.01.76	111
16593	Петрова	Ганна	Володимирівна	15.03.75	112
16693	Анохін	Андрій	Борисович	14.04.76	111

СЕСІЯ

№ особової справи	Оцінка за предмет 1	Оцінка за предмет 2	Оцінка за предмет 3	Оцінка за предмет 4	Результат
16493	3	4	3	4	3.5
16593	4	5	4	5	4.5
16693	5	5	5	5	5

СТИПЕНДІЯ

Результат	Відсоток
3.5	50
4.5	75
5	100

Рис. 3.5 – Приклад реляційної моделі

Дамо основні визначення, пов'язані з реляційною моделлю:

Домен – це пойменована кінцева множина значень одного типу.

Відношення – це підмножина декартового добутку доменів.

Відношення складається із заголовка та тіла.

Заголовок відношення – це множина *атрибутів* відповідних доменів. Позначається як множина пар виду $\{< A1 : D1 >,...,< An : Dn >\}$, де A_j – ім'я атрибута, D_j – ім'я домену, n – кількість доменів.

Тіло відношення – це множина *кортежів.* Кожен i-й кортеж складається із множини значень, що відповідають атрибутам. Позначається як множина пар виду $\{< A1 : Vi1 >,...,< An : Vin >\}$, де V_{ij} – значення атрибута з ім'ям A_j з домену з ім'ям D_j.

Кардинальне число – кількість кортежів.

Ступінь – кількість атрибутів.

Базове відношення – це пойменоване відношення, що не є похідним. На практиці базове відношення – це відношення, яке з огляду на свою важливість поіменовано і є частиною БД.

Похідне відношення – це відношення, визначене за допомогою реляційного вираження через інші зазначені відношення і в кінцевому підсумку через базові відношення.

Наприклад, похідним відношенням є результат запиту.

Представлення – це поіменоване похідне відношення, виражене виключно через інші зазначені відношення. Створення представлення схоже на створення запиту, але, на відміну від результату запиту, він названий і не містить даних (порожня реляційна таблиця).

Знімок – це поіменоване похідне відношення, виражене не тільки через інші зазначені відношення, а й своїми даними. Створення знімка схоже на створення запиту, але, на відміну від результату запиту, він названий і оновлюється.

Неіменоване відношення – це результат обчислення деякого реляційного відношення. На відміну від іменованого відношення (базового відношення, представлення, знімка) неіменоване відношення існує недовго і множина його елементів (кортежів) не змінюється. Наприклад, неіменованим відношенням є проміжні та кінцеві результати запитів.

Відношення можна представити у вигляді реляційної таблиці (наприклад, на рис. 3.5 відношеннями є СТУДЕНТ, СЕСІЯ, СТИПЕНДІЯ). Тут дамо такі визначення:

1. *Атрибут* – це поле (заголовок стовпчика) реляційної таблиці.

Для опису атрибута використовуються характеристики:

- ім’я (наприклад, Прізвище, Ім’я, По батькові, Дата народження);
- тип (наприклад, символьний, числовий, календарний);
- довжина (наприклад, 4 байти), яка визначається максимально можливою кількістю символів;
- точність для числових даних (наприклад, два десяткові знаки для відображення дробової частини числа).

2. *Домен* – це область допустимих значень одного атрибута.

3. *Кортеж* – це рядок реляційної таблиці. Кортеж відповідає примірнику логічного запису.

Наприклад, на рис. 3.5 кортежем відношення СТУДЕНТ буде такий рядок:

16493	Сергєєв	Петро	Михайлович	01.01.76	111

Рис. 3.6 – Кортеж з прикладу реляційної моделі на рис. 3.5

4. *Суперключ* – це підмножина множини атрибутів базового відношення, якому притаманна властивість унікальності (немає двох різних кортежів з однаковим значенням суперключа), тобто суперключ єдиним способом ідентифікує кортеж базового відношення.

5. *Потенційний ключ* (ключ-кандидат, унікальний ключ) – це підмножина множини атрибутів базового відношення, яка має властивість унікальності (немає двох різних кортежів з однаковим значенням потенційного ключа) і властивість ненадлишкових (жодна з підмножин потенційного ключа не має властивості унікальності). Кожне базове відношення має принаймні один потенційний ключ, оскільки комбінація всіх атрибутів має властивість унікальності, адже кортежі унікальні. Потенційний ключ є окремим випадком суперключа.

6. *Первинний ключ* – це обраний потенційний ключ базового відношення. Атрибути первинного ключа базового відношення не містять неіснуючих значень (NULL-значення).

7. *Альтернативні ключі* – це невибрані потенційні ключі базового відношення. Атрибути альтернативного ключа базового відношення можуть містити неіснуючі значення (NULL-значення).

8. *Вторинний ключ* – це підмножина множини атрибутів базового відношення, яка не є первинним ключем. Загалом вторинний ключ може не мати властивості унікальності.

9. *Зовнішній ключ* – це підмножина множини атрибутів одного базового відношення, яке або відповідає потенційному ключу іншого базового відношення, або містить тільки неіснуючі значення (NULL-значення). Але

зворотне необов’язково, тобто не всі значення потенційного ключа одного базового відношення мають відповідати значенню зовнішнього ключа іншого базового відношення. Зовнішній ключ використовується, щоб зв’язати два базові відношення. NULL-значення використовуються, якщо для значення зовнішнього ключа одного базового відношення відсутній кортеж іншого базового відношення, що містить відповідне значення потенційного ключа. Зовнішній ключ завжди є вторинним ключем, але зворотне необов’язково.

10. Простий ключ – це ключ, що складається з одного атрибута.

11. Складний ключ – це ключ, що складається з більше ніж одного атрибута.

Потенційний ключ, суперключ, зовнішній ключ, первинний ключ, альтернативний ключ і вторинний ключ можуть бути і простими, і складними.

Приклад

Згідно з рис. 3.5, таблиці СТУДЕНТ і СЕСІЯ мають співпадаючі первинні прості ключі («№ особової справи»), що дає можливість легко організувати зв’язок між ними, а таблиця СТИПЕНДІЯ має первинний простий ключ «Результат». Таблиця СТУДЕНТ містить простий потенційний ключ «№ особової справи» та складний потенційний ключ «ПІБ», таблиця СЕСІЯ містить простий потенційний ключ «№ особової справи», таблиця СТИПЕНДІЯ містить простий потенційний ключ «Результат». Таблиця СЕСІЯ містить зовнішній ключ «Результат», який забезпечує її зв’язок із таблицею СТИПЕНДІЯ. Будь-який ключ у таблицях СТУДЕНТ, СЕСІЯ і СТИПЕНДІЯ, крім первинного ключа, є вторинним.

Із використанням поняття потенційного, первинного і зовнішнього ключа формулюються правила цілісності реляційної моделі:

1. Цілісність об’єктів – жоден атрибут первинного ключа базового відношення не може містити неіснуючих значень (NULL-значення).

2. Посилальна цілісність – БД повинна містити тільки узгоджені значення зовнішніх ключів, тобто кожне значення зовнішнього ключа одного базового відношення має або відповідати значенню потенційного (зокрема первинного) ключа іншого базового відношення, або містити неіснуюче значення (NULL-значення).

3. Цілісність атрибутів – значення кожного атрибута беруться з відповідного домену.

Переваги реляційної моделі:

1. Простота моделі (використання прямокутних таблиць для представлення структур даних – найпростіший спосіб роботи з БД).

2. Проста і гнучка непроцедурна мова маніпулювання даними на основі реляційної алгебри або реляційного обчислення, за допомогою якого обробляють відношення.

3. Незалежність даних.

4. Теоретичне обґрунтування моделі на основі формального апарату реляційної алгебри і реляційного числення для обробки даних.

5. Спрощення контролю секретності (для кожного відношення задається правомірність доступу; засекречені атрибути відношення можна виділити у спеціальні відношення з вимогою перевірки прав доступу, і якщо це право не порушено, то ці атрибути можна приєднати до інших атрибутів вихідного відношення).

6. Ясність логічної моделі БД.

7. Фізичне розміщення прямокутних таблиць простіше, ніж розміщення дерев і мереж.

8. Найбільша кількість успішних СУБД, що використовують цю модель.

3.3. Моделі представлення знань

Знання можуть бути класифіковані за такими категоріями:

- *поверхневі* – знання про видимі взаємозв'язки між окремими подіями і фактами у предметній області;
- *глибинні* – абстракції, аналогії, схеми, що відображають структуру і природу процесів, які протікають у предметній області. Ці знання пояснюють явища і можуть використовуватися для прогнозування поведінки об'єктів.

Приклад

Поверхневі знання: «Якщо натиснути на кнопку дзвінка, пролунає звук. Якщо болить голова, то слід прийняти аспірин». Глибинні знання: «Принципова електрична схема дзвінка та проводки. Знання фізіології і лікарів високої кваліфікації про причини, види головних болів і методи їх лікування».

Сучасні ЕС працюють в основному з поверхневими знаннями. Це пов'язано з тим, що наразі немає універсальних методик, що дадуть змогу виявляти глибинні структури знань і працювати з ними.

До того ж знання ділять на *процедурні* та *декларативні.* Історично первинними були процедурні знання, тобто знання, «розчинені» в алгоритмах. Вони керували даними. Для їх зміни потрібно змінювати програми. Однак із розвитком ШІ пріоритет даних поступово змінювався, і все більша частина знань зосереджувалася у структурах даних (таблиці, списки, абстрактні типи даних), тобто збільшувалася роль декларативних знань.

Сьогодні знання набули суто декларативної форми, тобто знаннями вважаються пропозиції, записані мовами подання знань, наближених до природної і зрозумілих нефахівцеві.

Існують десятки моделей (або мов) подання знань для різних предметних областей. Більшість із них може бути зведена до таких класів:

1) мережеві моделі:
   - семантичні мережі;
   - фрейми;
   - сценарії.
2) немережеві моделі:
   - продукційні моделі;
   - формальні логічні моделі;
   - реляційні моделі.

3.3.1. Продукційна модель

Продукційна модель, або модель, заснована на правилах, дає змогу уявити знання про предметну область у вигляді пропозицій типу «Якщо (умова), то (дія)».

Загалом продукція розуміється як вираз такого виду:

$$(i): Q; P; A \Rightarrow B; N,$$

де (i) – ім'я продукції;

Q – передумова продукції (описує предметну область знання, яку представляє окрема продукція, у своїй декомпозиції предметної області на окремі незалежні області може значно підвищити ефективність міркувань у продукційній системі);

P – умова застосування ядра продукції (зазвичай у вигляді предиката);

$A \Rightarrow B$ – ядро продукції;

A – умова ядра продукції (антецедент) (зразок, яким здійснюється пошук у БЗ);

B – заключення ядра продукції (консеквент) (може бути проміжним і виступати далі як умова, або термінальним (цільовим) і завершувати роботу системи);

N – постумова продукції (описує дії та процедури, які необхідно виконати у разі реалізації ядра продукції).

На практиці зазвичай використовують такий окремий випадок продукції:

ПРАВИЛО k : ЯКЩО умова, k ТО заключення k,

де умова k – сукупність умов (висловлювань), сполучених бінарною операцією (зазвичай кон'юнкцією «І» або диз'юнкцією «АБО»),

заключення k – сукупність підзаключень (висловлювань), з'єднаних бінарною операцією (зазвичай кон'юнкцією «І» чи диз'юнкцією «АБО»).

Найчастіше виведення на такій БЗ буває *прямим* (від даних до пошуку мети) або *зворотним* (від цілі для її підтвердження – до даних). Дані – це початкові факти, що зберігаються в базі фактів, на підставі яких запускається машина виведення або інтерпретатор правил, що перебирає правила продукційної БЗ.

Продукційна модель найчастіше застосовується в ЕС. Вона привертає розробників своєї наочністю, високою модульністю, легкістю внесення доповнень і змін, простотою механізму логічного виведення. Є велика кількість програмних засобів, що реалізують продукційний підхід, а також промислових ЕС на його основі.

3.3.2. Семантичні мережі

Термін «семантична» означає «смислова», а сама семантика – це наука, що встановлює відношення між символами і предметами, які вони позначають, тобто наука, яка визначає зміст знаків.

Семантична мережа є інформаційною моделлю предметної області і має вигляд орієнтованого графа, вершини якого – поняття, а дуги – відношення між ними.

Як поняття зазвичай виступають предмети реального світу, як відношення – зв'язки типу «клас–підклас» («АКО», яке позначає «a kind of»), «клас–екземпляр класу» («is_a»), «ціле–частина» («a_part_of» або «has»), «є власністю» («належить»), «подобається», «колір» та ін.

Можна запропонувати кілька класифікацій семантичних мереж, пов'язаних із типами відношень між поняттями.

За кількістю типів відношень розрізняють:

- однорідні (з єдиним типом відношень);
- неоднорідні (з різними типами відношень).

За типами відношень:

- бінарні (в яких відношення пов'язують два об'єкти);
- n-арні (в яких є спеціальні відношення, що зв'язують більше двох понять).

Найбільш часто в семантичних мережах використовуються такі відношення:

- теоретико-множинні (клас–підклас, клас–примірник класу, ціле–частина тощо);

- функціональні (є власністю, подобається тощо);
- кількісні зв’язки (більше, менше, дорівнює та ін.);
- просторові зв’язки (далеко від, близько від, за, під, над та ін.);
- тимчасові зв’язки (раніше, пізніше, протягом);
- атрибутивні зв’язки (колір, розмір, форма);
- логічні зв’язки (І, АБО, НЕ);
- квантифіковані (квантори загальності та існування);
- відмінкові (агент, об’єкт, інструмент, місце, час тощо).

Проблема пошуку рішення в БЗ типу семантичної мережі зводиться до задачі пошуку фрагмента мережі, відповідного до деякої підмережі, що відбиває поставлений запит до бази.

Приклад

На рис. 3.7 зображена семантична мережа. Як вершини тут виступають поняття «Людина», «Степаненко», «Автомобіль», «Волга», «Двигун», «Червоний». Дугами є «АКО», «is_a», «a_part_of» або «has», «належить», «подобається», «колір».

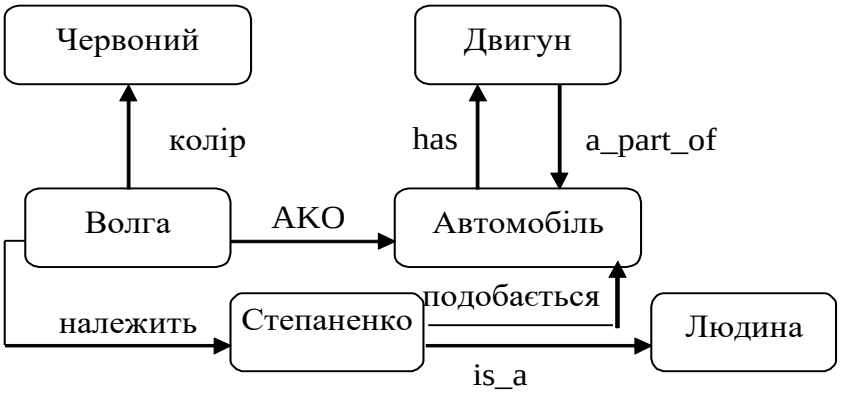


Рис. 3.7 – Семантична мережа

Ця модель подання знань була запропонована американським психологом Куілліаном. Її основна перевага в тому, що вона більше за інші відповідає сучасним уявленням про організацію довготривалої пам’яті людини. Недоліком цієї моделі є складність організації процедури пошуку виведення на семантичній мережі.

Для реалізації семантичних мереж існують спеціальні мережеві мови, наприклад, NET, мова реалізації систем SIMER + MIR та ін. Широко відомі ЕС, що використовують семантичні мережі як мови представлення знань – PROSPECTOR, CASNET, TORUS.

3.3.3. Фрейми

Термін *фрейм* (від англ. *frame*, що означає «каркас» або «рамка») був запропонований Марвіном Мінським, одним із піонерів ШІ в 70-і роки для позначення структури знань під час сприйняття просторових сцен. Ця модель, як і семантична мережа, має глибоке психологічне обґрунтування. *Фрейм* являє собою інформаційну модель предметної області і має вигляд локальної семантичної мережі. Але на відміну від звичайних семантичних мереж він забезпечує перетворення інформації всередині себе і зв'язок з іншими фреймами.

Наприклад, поняття «кімната» можна представити у вигляді такого опису: «житлове приміщення з чотирма стінами, підлогою, стелею, вікнами та дверима». З цього опису нічого не можна прибрати (наприклад, прибравши вікна, ми отримаємо вже не кімнату, а комору), але в ньому є незаповнені ділянки – «слоти». Слот відповідає властивості об'єкта, яка може змінювати своє значення (наприклад, кількість вікон, колір стін, висота стелі та ін.). У теорії фреймів такий опис кімнати називається фреймом кімнати. Фреймом також називається і формалізована модель для відображення опису кімнати.

Розрізняють *фрейми-зразки,* або *прототипи,* що зберігаються в БЗ, і *фрейми-екземпляри,* які створюються для відображення реальних фактичних ситуацій на основі даних, що надходять. Модель фрейма є досить універсальною, оскільки дає змогу відобразити все різноманіття знань про світ через:

- *фрейми-структури*, що використовуються для позначення об'єктів і понять (позика, застава, вексель);
- *фрейми-ролі* (менеджер, касир, клієнт);
- *фрейми-сценарії* (банкрутство, збори акціонерів, святкування іменин);
- *фрейми-ситуації* (тривога, аварія, режим пристрою) та ін.

Традиційно структура фрейму може бути представлена як список властивостей:

(Ім'я фрейма:
(Ім'я 1-го слота: значення 1-го слота),
...
(Ім'я N-го слота: значення N-го слота)),

Цей запис можна представити у вигляді таблиці, доповнивши її двома стовпцями.

У табл. 3.1 додаткові стовпці (спосіб отримання значення, приєднана процедура) призначені для опису способу отримання слотом його значення і можливого приєднання до того чи іншого слота спеціальних процедур, що допускається в теорії фреймів. Як значення слота може виступати ім'я іншого фрейма. Так утворюються мережі фреймів.

Таблиця 3.1

Структура фрейму

ім’я фрейму			
ім’я слота	**значення слота**	**спосіб отримання значення**	**приєднана процедура**
…	…	…	…

Існує кілька способів отримання слотом значень у фреймі-екземплярі:

- за замовчуванням від фрейму-зразка (Default-значення);
- через успадкування властивостей від фрейму, зазначеного у слоті АКО (A - Kind - Of = це);
- за формулою, зазначеною у слоті;
- через приєднану процедуру;
- явно з діалогу з користувачем;
- з БД.

Найважливішою властивістю теорії фреймів є запозичення з теорії семантичних мереж – так зване *успадкування властивостей*. І у фреймах, і в семантичних мережах спадкування відбувається за умови АКО-зв’язків. Слот АКО вказує на фрейм більш високого рівня ієрархії, звідки неявно успадковуються, тобто переносяться, значення аналогічних слотів.

Приклад

У мережі фреймів на рис. 3.8 поняття «учень» успадковує властивості фреймів «дитина» і «людина», які знаходяться на більш високому рівні ієрархії. Зокрема, на запитання «чи люблять учні солодке?» слід відповісти «так», тому що це властиво всім дітям, що зазначено у розіграші «дитина». Спадкування властивостей може бути частковим, адже вік для учнів не успадковується з фрейму «дитина», оскільки зазначений явно у своєму власному фреймі.

Основною перевагою фреймів як моделі подання знань є те, що вона відображає концептуальну основу організації пам’яті людини, а також її гнучкість і наочність.

Спеціальні мови представлення знань у мережах фреймів FRL (Frame Representation Language), KRL (Knowledge Representation Language), фреймова «оболонка» Карра та ін. Програмні засоби дають змогу ефективно будувати промислові ЕС. Широко відомі такі фрейм-орієнтовані ЕС: ANALYST, МОДІС, TRISTAN, ALTERID.

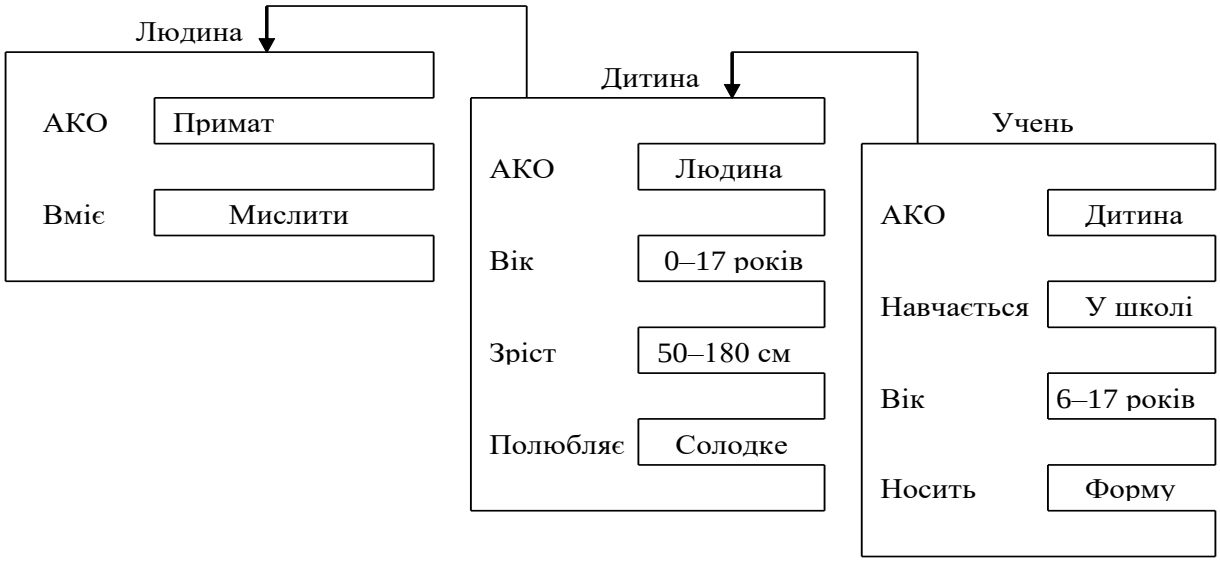


Рис. 3.8 – Мережа фреймів

3.3.4. Сценарії

Сценарій являє собою інформаційну модель предметної області у вигляді формалізованого опису стандартної послідовності взаємопов'язаних фактів.

Уперше поняття сценарію було введено Р. Шенком і Р. Абельсоном під час розробки нових засобів розуміння історії. Сценарії в їх системі представлялися у вигляді фреймоподібних структур та використовувалися для зв'язування подій історії. Кожен сценарій складався з набору слотів за їх значеннями, що описують ролі, причини і послідовності сцен, які були послідовністю певних дій. Слот «роль» ставив виконавців сценарію, слот «мета» – мотивування дій, що виконуються. Будь-яка попередня дія створює умови для здійснення подальшої дії.

Приклад

< Сценарій: ресторану
ролі: відвідувач, офіціант
мета: прийняття їжі, щоб насититися й отримати задоволення
сцена 1: вхід у ресторан
увійти в ресторан
оглянути місця
вибрати вільне місце
пройти до вільного столика
сісти
сцена 2: замовлення
взяти меню

прочитати меню
вирішити, що замовити
замовлення меню офіціантові

сцена 3: прийом їжі

отримання їжі
споживання їжі

сцена 4: догляд

прохання розрахувати
отримання чека
передача грошей офіціанту
вихід з ресторану>

3.3.5. Формальні логічні моделі

Формальна логічна модель дає змогу представити знання про предметну область у вигляді формул обчислення предикатів *i*-го порядку. У промислових ЕС використовуються різні модифікації цієї логічної моделі.

Визначення обчислення предикатів першого порядку.

1. Алфавіт:

константи: a, b, c, d, ...,
змінні: t, u, v, w, x, y, z, ...,
предикати: A, B, C, D, ...,
логічні оператори: $\neg$(не), $\wedge$(і), $\vee$(або), $\rightarrow$(якщо ... то ...),
квантори: $\forall$ (квантор загальності), $\exists$ (квантор існування).

2. Аксіоми:

$A \rightarrow (B \rightarrow A)$,
$(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (B \rightarrow C))$,
$(\neg A \rightarrow \neg B) \rightarrow (A \rightarrow B)$,
$\forall x\, F(x) \rightarrow F(y)$.

3. Правила виведення:

а) правило виведення (Modus Ponens): якщо A та $A \rightarrow B$ – формули, які логічно виведені, то B виведена: $\dfrac{A, A \rightarrow B}{B}$;

б) правила узагальнення: $\dfrac{F \rightarrow G(x)}{F \rightarrow \forall x G(x)}$, $\dfrac{G(x) \rightarrow F}{\exists x G(x) \rightarrow F}$.

Теорема дедукції

Нехай Г – деякий список формул. Тоді якщо Г, $\frac{\text{A}}{\text{B}}$, то $\frac{\Gamma}{\text{A} \to \text{B}}$,

або

якщо формула В виведена зі списку гіпотез Г, доповненого формулою А, то формула А $\to$В виведена зі списку гіпотез Г.

Приклад

Дано таке знання: «Кожна деталь повинна спочатку пройти фрезерування торців, а потім токарну обробку».

Перефразуємо це знання у вигляді: «Для кожної деталі *x* існують операції *y* і *z*, такі, що кінець операції *y* настає не пізніше початку операції *z*, операція *y* є фрезерування торців, операція *z* – токарна обробка, і в обох операціях бере участь одна і та сама деталь *x*». Відповідна формула буде такою:

$\forall x \in$Деталь $\exists y, z \in$Операція(кін(y)≤поч(z)$\wedge$дет(y)=$x\wedge$дет(y)=$x\wedge$фрез_торц(y) $\wedge$ток_обр (z)).

3.3.6. Реляційні моделі

Реляційна модель дає змогу представити знання про предметну область у вигляді формул реляційної алгебри або реляційного числення. В основі цього подання знань лежить уявлення інформаційних зв'язків між поняттями у вигляді відношень і таблиць, що складаються з рядків (кортежів) і стовпців (атрибутів).

Основні операції реляційної алгебри:

а) об'єднання відношень R_1, R_2

$R = R_1 \cup R_2$,

б) різниця відношень R_1, R_2

$R = R_1 - R_2$,

в) декартовий добуток відношень R_1, R_2

$R = R_1 \times R_2$,

г) проєкція відношення R_1 на компоненти (номери стовпців) $i_1,...,i_r$

$R = \pi_{i_1,...,i_r}(R_1)$,

д) селекція відношення R_1 за формулою F

$R = \text{o}_F(R_1)$, де формула F утворена:

- операндами, які є номерами стовпців;

- логічними операторами ¬(не), ∧(і), ∨(або);
- арифметичними операторами порівняння <, =,>, ≤, ≥, ≠.

Інші операції реляційної алгебри (перетин відношень, приватна відношень, з'єднання відношень, природне з'єднання) виходять за допомогою основних

Основні операції обчислення висловлювань:

а) об'єднання відношень R_1, R_2

$\{t \mid R_1(t) \vee R_2(t)\}$,

б) різниця відношень R_1, R_2

$\{t \mid R_1(t) \wedge \neg R_2(t)\}$,А

в) декартовий добуток відношень R_1, R_2

$$\{t^{(m+n)} \mid (\exists u)(\exists v)(R_1(u) \wedge R_2(v) \wedge t[1] = u[i_1] \wedge ... \wedge t[m] = u[m] \wedge$$
$$\wedge t[m+1] = v[1] \wedge ... \wedge t[m+n] = v[n])\},$$

г) проєкція відношення R_1 на компоненти (номери стовпців) $i_1,...,i_r$

$\{t^{(r)} \mid (\exists u)(R_1(u) \wedge t[1] = u[i_1] \wedge ... \wedge t[r] = u[i_r])\}$,

д) селекція відношення R_1 за формулою F

$\{t \mid R_1(t) \wedge F'\}$, де вираз F' є вираз F, у якому кожен операнд, що позначає компонент i, замінений на $t[i]$.

Приклад відношення R_1 наведено в табл. 3.2.

Таблиця 3.2

Таблиця відношень

Продавець	**Покупець**	**Купівля**	**Властивість (вік) покупки**	**Час купівлі**
Том	Джон	автомобіль	новий	вчора
Білл	Гаррі	мотоцикл	старий	сьогодні
...	...	...	...	...

Запит до реляційної БЗ «який продавець якому покупцю продав яку покупку» виконаний за допомогою:

а) реляційної алгебри

$R = \pi_{i_1, i_2, i_3}(R_1)$;

б) реляційного числення

$\{t^{(3)} \mid (\exists u)(R_1(u) \wedge t[1] = u[i_1] \wedge t[2] = u[i_2] \wedge t[3] = u[i_3])\}$.

4. ПОПОВНЕННЯ ЗНАНЬ НА ОСНОВІ ПСЕВДОФІЗИЧНИХ ЛОГІЧНИХ МОДЕЛЕЙ

В ЕС під час вирішення завдань розуміння природної мови виникає проблема поповнення знань. Описуючи ситуацію, людина зазвичай не повідомляє всі подробиці, тому, щоб повністю зрозуміти ситуацію, потрібні механізми відновлення пропущеної інформації про зовнішній світ. Одним із таких механізмів є псевдофізична логіка (ПФЛ), яка дає змогу породжувати нові знання. ПФЛ вміщує в себе ПФЛ часу, ПФЛ простору, каузальну ПФЛ, ПФЛ дій (Додаток А).

4.1. Логіка часу

Логіка часу являє собою систему, що описує знання про часові відношення між подіями і містить механізм поповнення цих знань.

Загальна структура логіки часу показана на рис. 4.1.

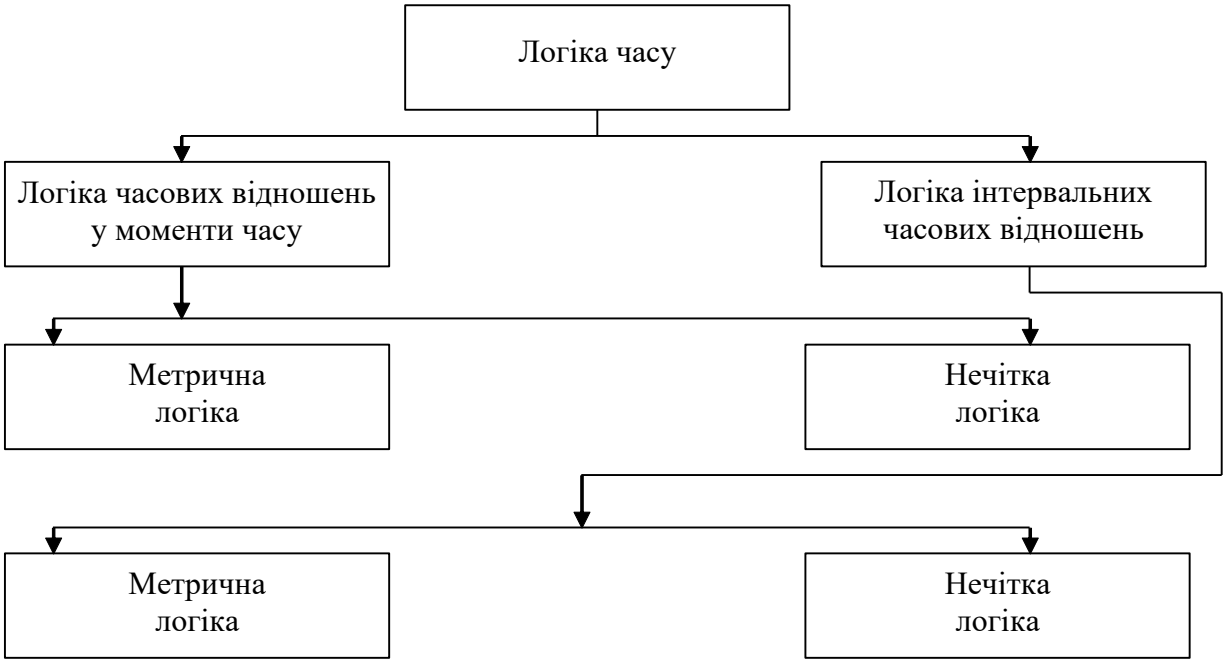


Рис. 4.1 – Загальна структура логіки часу

У метричній логіці вказується відстань між точками з точкою відліку (наприклад, «о 15 годині 20 хв 5 жовтня 2003 р.») Або просто відстань між точками (наприклад, «о 15 годині 20 хв»). У нечіткій логіці вказується лише ставлення порядку між впорядкованими подіями (наприклад, «раніше»).

У всіх моделях часу важливу роль відіграє поняття *події*. Під подією розуміється певний факт, пов’язаний із діяльністю якогось суб’єкта; зміною

властивостей об'єктів; діями, вчиненими на основі законів природи. Наприклад, «Робот забрав вантаж». Події бувають двох типів: точкові та інтервальні. Тип події характеризується тим, як вона проєктується на шкалі часу. Тобто якщо тривалість події займає одну поділку шкали, вона точкова, а якщо кілька поділок – то *інтервальна*. Наприклад, якщо масштаб шкали часу одна хвилина, а подія займає кілька хвилин, то вона інтервальна.

Метрична логіка точкових часових відношень використовує часові відношення типу «раніше на *n* одиниць». Нечітка логіка точкових часових відношень використовує часові відношення типу «одночасно», «раніше» або «раніше приблизно на *n* одиниць», при цьому лінгвістичною змінною виступає змінна *X = точкове часове відношення*, а її значенням T_1 *= раніше*, T_2 *= раніше приблизно на n одиниць*, $T_1, T_2 \in \Theta(X)$.

Метрична логіка інтервальних часових відношень використовує часові відношення типу «строго передує на *n* одиниць». Нечітка логіка точкових часових відношень використовує часові відношення типу «строго передує» або «строго передує приблизно на *n* одиниць», при цьому лінгвістичною змінною виступає змінна *X = інтервальне часове відношення*, а її значенням T_1 *= строго передує,* T_2 *= суворо передує приблизно на n одиниць,* $T_1, T_2 \in \Theta(X)$.

Для ПФЛ часу вводяться:

а) поняття (моменти часу і події) і відношення між ними;

б) правила виведення.

Для опису понять і відношень уведемо такі множини:

$T = \{t_i\}$, де t_i – моменти часу,

$P = \{p_j\}$, де p_j – події,

$R^1 = \{R_k^1\}$, де R_k^1 – точкові часові відношення,

$R^2 = \{R_k^2\}$, де R_k^2 – інтервальні часові відношення.

Приклад

$R_{3,M}^1(n, \omega)$ – подія p відбувається раніше події q на $n\omega$ одиниць за шкалою L, де ω – одиниця виміру шкали T, n = 1, 2, 3, ...,

$R_{4,M}^1(t^*)$ – подія p реалізується в момент t^* на шкалі T,

R_0^1 – подія p відбувається одночасно з подією q,

R_1^2 – подія p строго передує q,

R_2^2 – подія p строго слідує за q.

Інтервальну подію можна задати парою точкових подій – маркерів початку $(\mu_п)$ та кінця $(\mu_к)$. У цьому разі відношення між парами інтервальних

подій можливо задавати через сукупності відношень між точковими подіями, які слугують їх маркерами.

Приклад

$\mu_к(p_i)R^1_{4,M}(t)$ – інтервальна подія p_i завершилася в момент часу *t*.

Узагальнені правила виведення представлені у вигляді:

$\alpha \to \beta$.

Приклад

$p_i R^1_0 p_j, p_j R^1_{4,M}(t_1) \to p_i R^1_{4,M}(t_1)$,

$\mu_к(p_1)R^1_{3,M}(n,\omega)\mu_п(p_2) \to p_1 R^2_1 p_2$,

$p_i R^2_1 p_j \to p_j R^2_2 p_i$,

$p_i R^2_2 p_j \to p_j R^2_1 p_i$,

$p_i R^2_1 p_j, p_j R^2_1 p_k \to p_i R^2_1 p_k$.

Використовуючи модель часу, можна вирішувати різні завдання. Наприклад, визначення часу реалізації подій, встановлення їх взаємного розташування в часі, проєктування подій на шкали часу, співвіднесення шкал між собою, аналіз часових залежностей між подіями і виведення нових залежностей. На вхід системи, яка використовує модель часу, надходить текст природною мовою, що описує ситуацію зовнішнього світу. Ситуації можуть являти собою різні дії суб'єкта над об'єктами у просторово-часовому континуумі. Потрібно визначити всі можливі часові відношення між подіями.

Приклад

Дано такий текст: «Літак рейсом Баку–Київ здійснив посадку о 15 годині 20 хв. Через 10 хв був поданий трап і потім пасажири покинули літак. Прийшов автобус і через 2 хв пасажири були вже в будівлі аеровокзалу». У тексті виділяються події:

p_1 – літак, що прямує рейсом Баку–Київ, здійснив посадку,

p_2 – подача трапа,

p_3 – вихід пасажирів,

p_4 – подача автобуса,

p_5 – поява пасажирів у будівлі аеровокзалу.

Структура тексту представлена у вигляді

$ST = (\mu_к(p_1)R^1_{4,M}(t)) \wedge (\mu_к(p_1)R^1_{3,M}(10, \text{хв})\mu_п(p_2)) \wedge (p_3 R^2_2 p_2) \wedge (p_5 R^2_2 p_4) \wedge$

$\wedge (\mu_к(p_4)R^1_{3,M}(2, \text{хв})\mu_п(p_5))$, t = 15 годин 20 хв.

Поповнимо структуру *ST* на основі правил виведення новими фактами

1. $(\mu_к(p_1)R^1_{4,M}(t)),(\mu_к(p_1)R^1_{3,M}(10,\text{хв})\mu_п(p_2))\to\mu_п(p_2)R^1_4(t')),$
   t'=15 годин 30 хв,
2. $(\mu_к(p_1)R^1_{3,M}(10,\text{хв})\mu_п(p_2))\to p_1R^2_1p_2,$
3. $p_1R^2_1p_2\to p_2R^2_2p_1,$
4. $p_3R^2_2p_2\to p_2R^2_1p_3,$
5. $p_1R^2_1p_2, p_2R^2_1p_3\to p_1R^2_1p_3,$
6. $(\mu_к(p_4)R^1_{3,M}(2,\text{хв})\mu_п(p_5))\to p_4R^2_1p_5.$

Приєднавши виведені факти до *ST*, отримаємо

$$ST^*=ST\cup\{\mu_п(p_2)R^1_{4,M}(t')\wedge(p_1R^2_1p_2)\wedge(p_2R^2_2p_1)\wedge(p_2R^2_1p_3)\wedge(p_1R^2_1p_3)\wedge(p_4R^2_1p_5)\}.$$

4.2. Логіка простору

Логіка простору являє собою систему, що описує знання про взаємне розташування об’єктів та місцезнаходження об’єктів у просторі і містить механізм поповнення цих знань. Загальна структура логіки простору показана на рис. 4.2.

У метричній логіці простору вказується відстань (або напрям) між точками з точкою відліку (наприклад, «перебувати в 5 кілометрах від Північного полюса» або «перебувати під кутом 30° від Північного полюса») або просто відстань (або напрям) між точками (наприклад, «перебувати в 5 кілометрах від мене» або «перебувати під кутом 30°»). У нечіткій логіці вказується лише відношення порядку між впорядкованими об’єктами (наприклад, «близько»).

Метрична логіка взаємного розташування об’єктів використовує просторові відношення типу «перебувати під кутом *A*». Нечітка логіка взаємного розташування об’єктів використовує просторові відношення типу «перебувати праворуч» або «перебувати приблизно під кутом *A*», при цьому лінгвістичною змінною виступає змінна *X* = *взаємне розташування об’єктів*, а її значенням T_1 = *перебувати праворуч*, T_2 = *перебувати приблизно під кутом A*, $T_1,\ T_2\in\Theta(X)$.

Метрична логіка відстаней використовує просторові відношення типу «в *n* одиницях». Нечітка логіка відстаней використовує просторові відношення типу «близько» або «приблизно в *n* одиницях», при цьому в якості лінгвістичної змінної виступає змінна *X* = *відстань*, а в якості її значення T_1 = *близько*, T_2 = *приблизно в n одиницях*, $T_1,T_2\in\Theta(X)$.

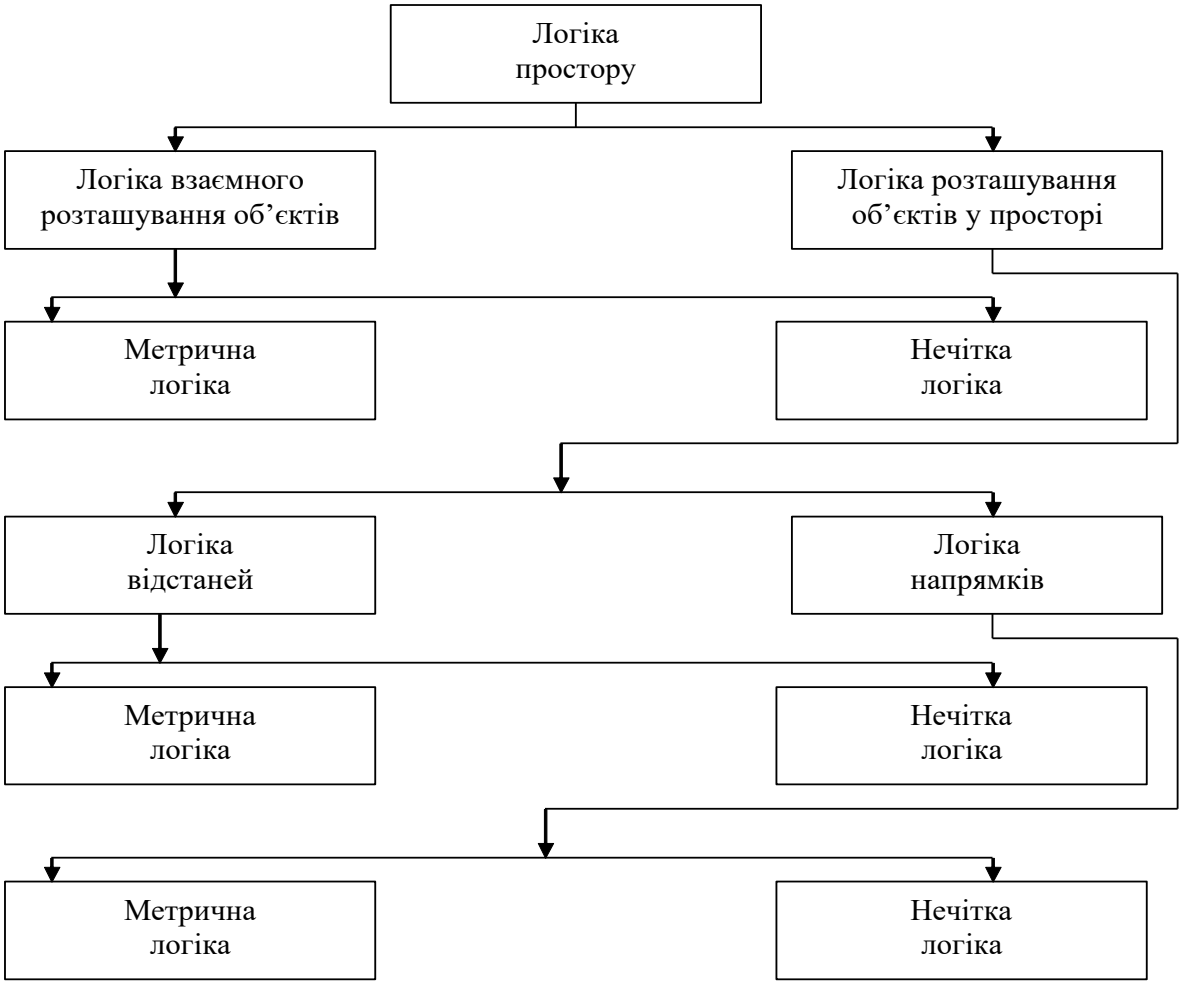


Рис. 4.2 – Загальна структура логіки простору

Метрична логіка напрямків використовує просторові відношення типу «під кутом *A*». Нечітка логіка напрямків використовує просторові відношення типу «справа» або «приблизно під кутом *A*», при цьому лінгвістичною змінною виступає змінна *X* = *напрямок*, а її значенням T_1 = *праворуч*, T_2 = *приблизно під кутом A*, $T_1, T_2 \in \Theta(X)$.

Для ПФЛ простору вводяться:

а) поняття (об'єкти) і відношення між ними;

б) правила виведення.

Для опису понять і відношень введемо такі множини:

$O = \{O_j\}$, де $O_j = (name, type, d_{\max}, d_{\min}, h, orientation)$ – об'єкти, *name* – ім'я,

$type \in$ {*крапка, інтервал, плоска фігура, об'ємна фігура*} – тип,

$d_{\max}$ – довга сторона,

$d_{\min}$ – коротка сторона,

h – висота об’єкта, $orientation \in \{передня\ \ сторона, нижня\ \ сторона\}$ – орієнтація об’єкта.

$R^3 = \{R_k^3\}$, де R_k^3 – просторові відношення для логіки відстаней.

$R^3 = \{R_k^4\}$, де R_k^4 – просторові відношення для логіки напрямків.

$R^5 = \{R_k^5\}$, де R_k^5 – просторові відношення для логіки взаємного розташування об’єктів.

Приклад

$O_i R_1^3 O_j$ – об’єкт O_i впритул з об’єктом O_j,

$O_i R_7^4 O_j$ – об’єкт O_i ззаду об’єкта O_j,

$O_i R_{11}^5 O_j$ – об’єкт O_i є частиною об’єкта O_j.

Узагальнені правила виведення представлені у вигляді:

$$\alpha \to \beta \text{ (якщо } \alpha \text{ то } \beta).$$

Приклад

$O_i R_1^3 O_j \to O_j R_{21}^5 O_i$, де R_{21}^5 – стикатися,

$O_i R_7^4 O_j \to O_i R_{37}^5 O_j$, де R_{37}^5 – перебувати позаду.

4.3. Каузальна логіка

Каузальна логіка являє собою систему, що описує знання про причинно-наслідкові відношення, характерні для кожної проблемної області, та містить механізм поповнення цих знань. На відміну від логік часу та простору, ця логіка проблемно-залежна.

Для каузальної ПФЛ вводяться:

а) поняття (моменти часу, події і дії) і відношення між ними;

б) правила виведення.

Для опису понять і відношень введемо такі множини:

$T = \{t_i\}$, де t_i – моменти часу,

$P = \{p_j\}$, де p_j – події,

$D = \{d_j\}$, де d_j – дії,

$R^6 = \{R_k^6\}$, де R_k^6 – каузальні відношення.

Приклад

R_0^6 – загальне умовне відношення, що відображає залежність будь-якого роду між подіями (діями) p та q.

R_1^6 – необхідне та достатнє причинне відношення між подіями (діями) p та q. Воно означає, що реалізацію події (дії) p завжди викликає подія (дія) q, і навпаки, поява q завжди викликається p. Найчастіше R_1^6 відображає фізичні закони зовнішнього світу (наприклад, «блиснула блискавка – гримнув грім»).

R_2^6 – достатнє причинне відношення між подіями (діями) p та q. Воно означає, що реалізація події (дії) p завжди викликає подію (дію) q, однак із появою q не завжди йде поява p (наприклад, «увійшов у кімнату – увімкнув світло»).

R_3^6 – обумовлює причинне відношення між подіями (діями) p та q. Воно означає, що реалізація події (дії) p забезпечує необхідні умови для реалізації q, яка може й не статися (наприклад, «увійшов в кімнату – увімкнув світло»).

Узагальнені правила виведення представлені у вигляді:

$$\alpha \to \beta .$$

Приклад

$$p_i R p_j \to p_j R_1^1 p_i ,\ R \in \{R_1^6, R_2^6, R_3^6\}.$$

Приклад

Дано такий текст: «Робочий виконував нарізку болта на верстаті».

Опишемо події тексту:

p_1 – робочий виконував нарізку болта на верстаті.

Операція нарізки болта передбачає такі події:

p_2 – робочий бере болт,

p_3 – робочий встановлює болт на верстат.

Структура тексту представлена у вигляді

$$ST = (p_2 R_3^6 p_1) \wedge (p_3 R_3^6 p_1) \wedge (p_2 R_3^6 p_3).$$

Поповнимо структуру ST на основі правил виведення новими фактами

$$p_2 R_3^6 p_3 \to p_2 R_1^1 p_3.$$

Приєднавши виведені факти на ST, отримаємо $ST^* = ST \cup \{p_2 R_1^1 p_3\}$.

4.4. Логіка дій

Логіка дій являє собою систему, що описує знання про дії, умови їх протікання в часі і просторі, їх взаємодії, цілі та мотиви дійових осіб і містить механізм поповнення цих знань.

Для ПФЛ дій вводяться:

а) поняття (моменти часу, локалізації, суб'єкти, об'єкти, інструменти та дії) і відношення між ними;

б) правила виведення.

Для опису понять та відношень введемо такі множини:

$T = \{t_i\}$, де t_i – моменти часу,

$L = \{l_j\}$, де l_i – локалізації,

$S = \{S_j\}$, де S_j – суб'єкти,

$O = \{O_j\}$, де O_j – об'єкти,

$I = \{i_j\}$, де i_j – інструменти (суб'єкт впливає ними на об'єкт),

$D = \{d_j\}$, де $d_j = d_j(s_m, o_n, i_p, t_r, l_s)$ – дії,

$R^7 = \{R_k^7\}$, де R_k^7 – відношення дії.

Приклад

$d_i R_2^7 d_j$ – дія d_i призводить до результату d_j,

Узагальнені правила виведення представлені у вигляді:

$$\alpha \to \beta .$$

Приклад

$d_i(S_m, O_n, t_r) \to d_i R_2^7 d_j(S_m, O_n, t_r)$,

$d_i(S_m, O_n, t_r), (O_n R_{11}^5 O_k) \to d_i(S_m, O_k, t_r)$,

$d_i(S_m, O_n, t_r) \to d_j(S_m, O_k, t_r + \Delta t)$,

$d_i(S_m, O_n, t_r + \Delta t), d_j(O_n, l_s, t_r + \Delta t) \to d_j(S_m, l_s, t_r + \Delta t)$.

Правила дають змогу аналізувати і поповнювати динамічні ситуації.

Приклад

Дано такий текст: «Він увійшов у вагон, і поїзд домчав до Києва за 3 години».

У тексті виділяються дії:

d_1 – увійшов,

d_2 – домчав,

d_3 – знаходиться в.

У тексті виділяються суб'єкти:

S_1 – він.

У тексті виділяються об'єкти:

O_1 – вагон,

O_2 – поїзд.

У тексті виділяються локалізації:

l_1 – Київ.

Структура дій представлена у вигляді

$ST = d_1(S_1, O_1, t_1) \wedge (O_1 R_{11}^5 O_2) \wedge d_2(O_2, l_1, t_1 + \Delta t)$, де Δt = 3 години.

Поповнимо структуру *ST* на основі правил виведення новими фактами

$d_1(S_1, O_1, t_1) \to d_1 R_2^7 d_3(S_1, O_1, t_1)$,

$d_1(S_1, O_1, t_1), (O_1 R_{11}^5 O_2) \to d_1(S_1, O_2, t_1)$,

$d_1(S_1, O_1, t_1) \to d_3(S_1, O_2, t_1 + \Delta t)$, де Δt =3 години,

$d_3(S_1, O_2, t_1 + \Delta t), d_2(O_2, l_1, t_1 + \Delta t) \to d_2(S_1, l_1, t_1 + \Delta t)$.

Приєднавши виведені факти до *ST*, отримаємо

$$ST^* = ST \cup \{d_1 R_2^7 d_3(S_1, O_1, t_1) \wedge d_1(S_1, O_2, t_1) \wedge d_3(S_1, O_2, t_1 + \Delta t) \wedge d_2(S_1, l_1, t_1 + \Delta t)\}.$$

5. ВИВЕДЕННЯ НА ЗНАННЯХ

5.1. Виведення на знаннях у вигляді продукційної моделі

5.1.1. Машина виведення

Найбільшого поширення набула продукційна модель представлення знань. Із використанням продукційної моделі БЗ ЕС складається з набору правил. *Програма, що керує перебором правил, називається машиною виведення*.

Машина виведення (інтерпретатор правил) ЕС виконує дві функції:

1) перегляд наявних фактів з робочої пам'яті (БД) та правил із БЗ і додавання (в міру можливостей) у робочу пам'ять нових фактів;

2) визначення порядку перегляду і застосування правил.

Цей механізм керує процесом консультацій, зберігаючи для користувача інформацію про отримані заключення, і запитує в нього інформацію, коли для спрацьовування чергового правила в робочій пам'яті виявляється недостатньо даних.

У переважній більшості систем, заснованих на знаннях, механізм виведення являє собою невелику за обсягом програму і містить два компоненти – один реалізує виведення, інший керує цим процесом.

Дія *компонента виведення* заснована на застосуванні правила, названого *modus ponens* (якщо відомо, що твердження А істинне та існує правило виду «ЯКЩО А, ТО В», тоді твердження В також істинне).

Правила спрацьовують, коли знаходяться факти, що задовольняють їх у лівій частині: якщо умова істинна, то має бути істинним і заключення.

Компонент виведення повинен функціонувати навіть у разі нестачі інформації. Отримане рішення може не бути точним, проте система не повинна зупинятися через те, що відсутня будь-яка частина вхідної інформації.

Керуючий компонент визначає порядок використання правил і виконує чотири функції:

1. Зіставлення – зразок правила зіставляється з наявними фактами.

2. Вибір – якщо в конкретній ситуації може бути застосовано відразу кілька правил, то з них вибирається одне, яке найбільше задовольняє заданий критерій (вирішення конфлікту).

3. Спрацювання – якщо зразок правила під час зіставлення збігся з будь-якими фактами з робочої пам'яті, то правило спрацьовує.

4. Дія – робоча пам'ять піддається зміні шляхом додавання в неї заключення правила, що спрацювало. Якщо у правій частині правила міститься вказівка на будь-яку дію, то воно виконується.

Інтерпретатор правил працює циклічно. У кожному циклі він переглядає всі правила, щоб виявити ті, посилки яких збігаються з відомими на цей момент фактами з робочої пам’яті. Після вибору правило спрацьовує, отриманий факт заноситься в робочу пам’ять і потім цикл повторюється.

В одному циклі може спрацювати тільки одне правило. Якщо кілька правил успішно зіставлені з фактами, то інтерпретатор робить вибір за певним критерієм єдиного правила, яке спрацьовує в цьому циклі.

Схема циклу роботи інтерпретатора правил зображена на рис. 5.1.

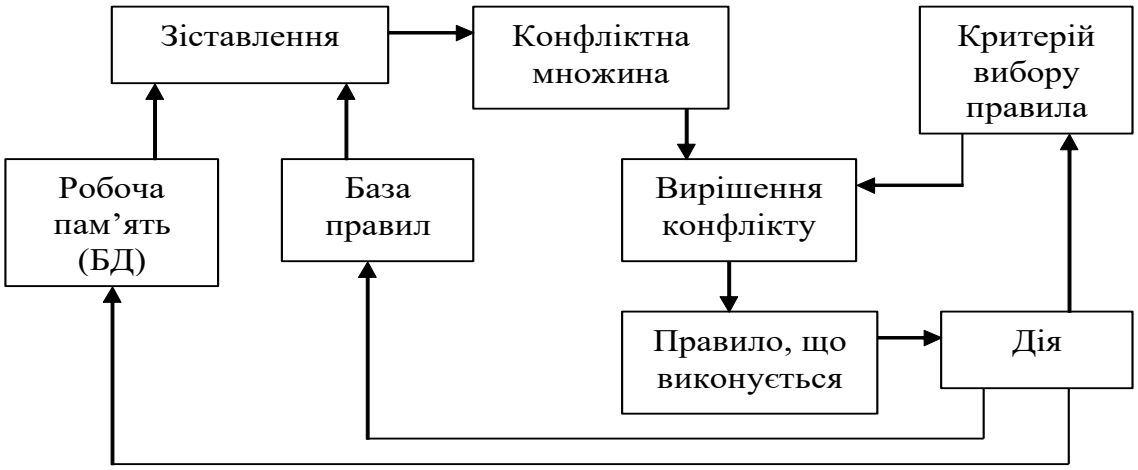


Рис. 5.1 – Цикл роботи інтерпретатора правил

Інформація з робочої пам’яті послідовно зіставляється з посилками правил для виявлення успішного зіставлення. Сукупність відібраних правил складає так звана *конфліктна множина*. Для вирішення конфлікту інтерпретатор має критерій, за допомогою якого він обирає єдине правило, після чого воно спрацьовує. Це виражається у занесенні фактів, що утворюють заключення правила, в робочу пам’ять або у зміну критерію вибору конфліктуючих правил. Якщо до заключення правила входить назва якої-небудь дії, то воно виконується.

Робота машини виведення залежить тільки від стану робочої пам’яті та від складу БЗ. На практиці зазвичай враховується історія роботи, тобто поведінка механізму виведення в попередніх циклах. Інформація про поведінку механізму виведення запам’ятовується в пам’яті станів (рис. 5.2). Зазвичай пам’ять станів містить протокол системи.

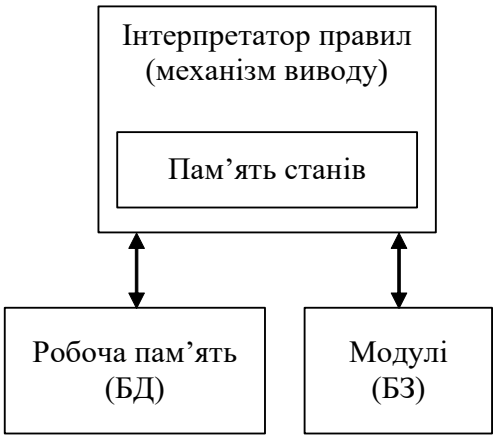


Рис. 5.2 – Схема функціонування інтерпретатора правил

5.1.2. Стратегії керування виведенням

Від обраного методу пошуку, тобто стратегії виведення, буде залежати порядок застосування та спрацювання правил. Процедура вибору зводиться до визначення напряму пошуку та способу його здійснення. Процедури, які реалізують пошук, зазвичай «зашиті» в механізм виведення, тому у більшості систем інженери знань не мають до них доступу, а отже, не можуть у них нічого змінити за своїм бажанням.

Під час розробки стратегії керування виведенням важливо визначити два питання:

1. З чого слід розпочати виведення? Від відповіді це питання залежить метод здійснення пошуку – у прямому чи зворотному напрямі.

2. Як вирішити конфлікт? Від відповіді це питання залежить спосіб керування порядком застосування правил конфліктної множини (пошук у глибину чи ширину та інших).

5.1.2.1. Пряме та зворотне виведення

В ЕС зі зворотним виведенням спочатку висувається деяка гіпотеза, а потім механізм виведення ніби повертається назад, переходячи до фактів, намагаючись знайти ті, які підтверджують гіпотезу (рис. 5.3, права частина). Якщо вона виявилася правильною, то вибирається наступна гіпотеза, що деталізує першу і є щодо неї підціллю. Далі відшукуються факти, що підтверджують істинність підпорядкованої гіпотези. Виведення такого типу називається виведенням, що керується цілями або консеквентами. Зворотний пошук застосовується в тих випадках, коли цілі відомі і їх порівняно небагато.

В ЕС із прямим *виведенням* за відомими фактами знаходиться заключення, яке з цих фактів випливає (ліва частина рис. 5.3). Якщо такий факт вдається знайти, то він заноситься в робочу пам’ять. Пряме виведення часто називають виведенням, що керується даними або антецедентами.

Існують системи, в яких виведення ґрунтується на поєднанні зворотного та прямого методу. Такий комбінований метод називається циклічним.

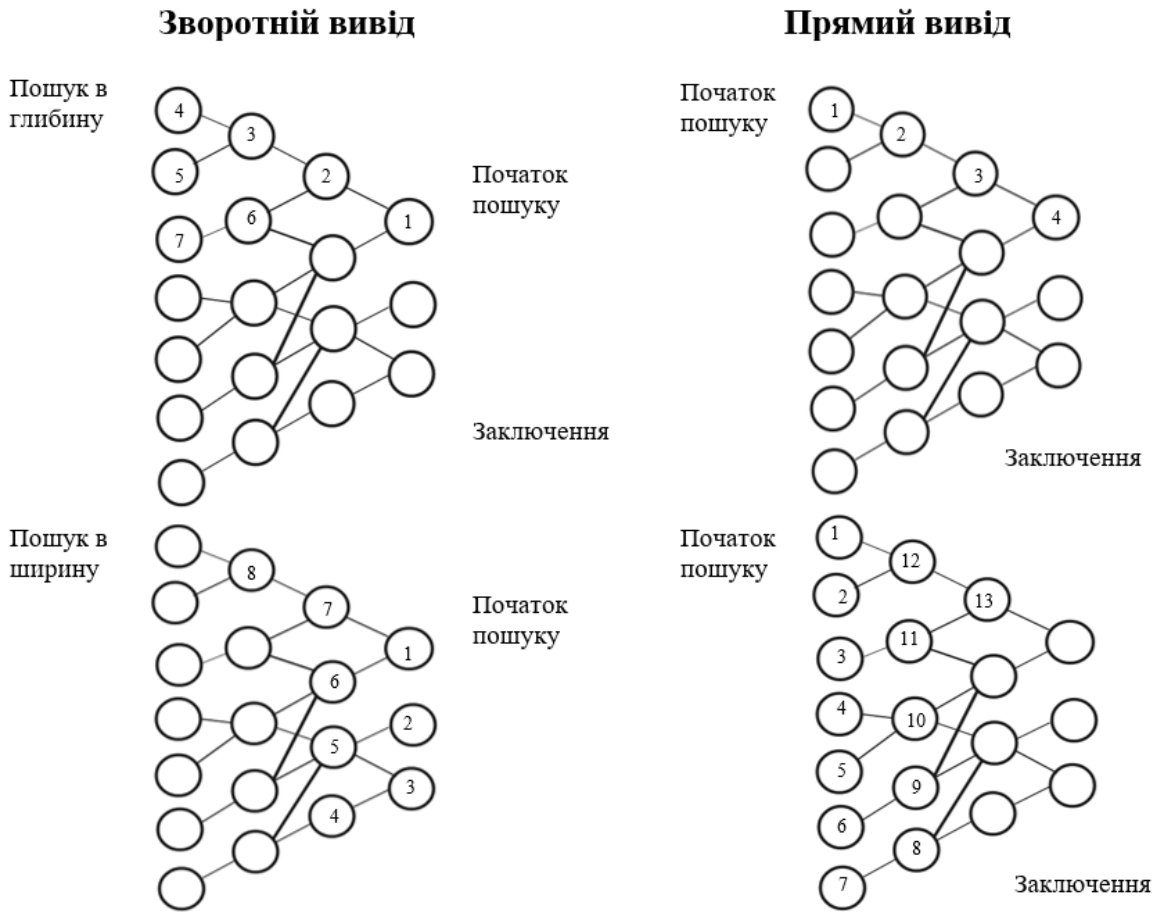


Рис. 5.3 – Стратегії виведення

Приклад

Є фрагмент БЗ із трьох правил:

П1. Якщо «відпочинок взимку» і «людина – активна», то «їхати в гори».

П2. Якщо «любить море», то «відпочинок влітку».

П3. Якщо «любить кататися на лижах», то «відпочинок взимку».

ПРЯМЕ ВИВЕДЕННЯ: спираючись на фактичні дані, отримати рекомендацію. Для вирішення конфлікту застосовуємо найбільш часто використовувану стратегію – з конфліктної множини обирається останнє з успішно зіставлених правил.

Крок 0. Нехай у систему надійшли факти «людина – активна», «любить кататися на лижах».

Крок 1. Перевіряємо всі правила й аналізуємо їх антецеденти (умови правил). Створюємо конфліктну множину. У цьому випадку вона складається з успішно зіставленого правила П3. Оскільки конфліктну множину складає тільки одне правило, то спрацьовує П3. У БД надходить факт «відпочинок взимку».

Крок 2. Перевіряємо всі правила й аналізуємо їх антецеденти. Створюємо конфліктну множину. У цьому випадку вона складається з успішно зіставленого правила П1. Оскільки конфліктну множину складає тільки одне правило, то спрацьовує правило П1. У БД надходить факт «їхати в гори», який виступає як ціль.

ЗВОРОТНЕ ВИВЕДЕННЯ – підтвердити обрану ціль за допомогою наявних правил і даних.

Крок 0. Нехай в систему надходить факт «їхати в гори», який виступає як ціль, а також факт «людина – активна», «любить кататися на лижах».

Крок 1. Проходимо за всіма правилами та аналізуємо їх консеквенти. У конфліктну множину заноситься правило П1. Правило П1 не вдається активізувати (факту «відпочинок взимку» немає), тому новою ціллю стає факт «відпочинок взимку».

Крок 2. Проходимо за всіма правилами й аналізуємо їх консеквенти. У конфліктну множину заноситься правило П3. Активізуємо правило П3. У БД надходить факт «відпочинок взимку». Поточна ціль досягнута (істинність факту «відпочинок взимку» підтверджена) та повертаємося до попередньої цілі.

Крок 3. Активізуємо правило П1. Шукана ціль досягнута (істинність факту «їхати в гори» підтверджена).

5.1.2.2. Методи пошуку в глибину та ширину

Вирішення конфлікту – важлива проблема, пов’язана з управлінням порядком застосування правил, що утворюють конфліктну множину. Порядок застосування правил конфліктної множини визначається обраною стратегією вирішення конфлікту. До таких стратегій належать: пошук у глибину, пошук в ширину, принцип «стосу книг», принцип найдовшої умови, принцип пріоритетного вибору та інші.

Під час пошуку в ширину вибирається правило конфліктної множини, яке було додано першим.

Під час пошуку в глибину вибирається правило конфліктної множини, яке було додано останнім.

Принцип «стосу книг» у тому, що список конфліктуючих правил упорядковується відповідно до частоти використання цих правил у минулому. Насамперед вибирається правило, яке використовується найчастіше.

Принцип найдовшої умови віддає пріоритет тому правилу, що має найдовшу умову.

Принцип пріоритетного вибору у тому, що з кожним правилом пов'язується пріоритет, який визначає порядок застосування.

5.2. Виведення на знаннях у вигляді формальної логічної моделі

5.2.1. Принцип резолюції

Для виведення на знаннях у вигляді формальної логічної моделі широке застосування отримав принцип резолюції. На цьому принципі базується і мова логічного програмування ПРОЛОГ.

Наведемо основні положення цього принципу.

Вихідна формула може бути представлена у вигляді кон'юнктивної нормальної форми (КНФ) $F_1 \wedge ... \wedge F_i ... \wedge F_n$, де F_i – диз'юнкт. Отже, будь-яка формула може бути представлена множиною диз'юнктів.

Основна ідея принципу резолюції полягає в перевірці, чи містить множину диз'юнктів S порожній диз'юнкт, що позначається як $\vee$. Якщо S містить порожній диз'юнкт, то вона суперечлива (нездійсненна). Якщо S не містить порожній диз'юнкт, то наступні кроки полягають у виведенні нових диз'юнктів. Якщо порожній диз'юнкт отримати неможливо, то S здійснювана.

Принцип резолюції для алгебри предикатів діє так: два диз'юнкти можуть бути резольвовані один з одним, якщо один із них містить предикат без заперечення, а інший – відповідний предикат із відмовою з одним і тим самим позначенням предиката і однаковою кількістю аргументів, і якщо аргументи обох предикатів можуть бути *уніфіковані* (узгоджені) один з одним. Внаслідок резолюції виходить новий диз'юнкт, який називається *резольвентою*.

Для методу резолюції важливу роль відіграє алгоритм уніфікації.

5.2.2. Алгоритм уніфікації

1. Приведення формули до наведеної нормальної форми.

1.1. Усі логічні операції виражаються через $\wedge$, $\vee$, $\neg$.

Для цього використовуються теореми:

$A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A)$,

$A \rightarrow B = \neg A \vee B$.

1.2. Зменшується радіус дії заперечень.

Для цього використовуються закони де Моргана, які дають змогу поширити дію заперечення тільки на поодинокі предикати:

$\neg(A \vee B) = \neg A \wedge \neg B$,

$\neg(A \wedge B) = \neg A \vee \neg B$.

2. Приведення формули до попередженої (пренексної) нормальної форми.

2.1. Перейменовуються зв'язані змінні, щоб одна і та сама змінна не мала вільних і пов'язаних входжень.

2.2. Видаляються ті квантори, область дії яких не містить входжень квантифікованої змінної.

2.3. Усі квантори переводяться в початок формули. Водночас зберігається порядок проходження кванторів:

$$(Q1\,x)A(x) \circ (Q2\,x)B(x) = (Q1\,x)(Q2\,y)(A(x) \circ B(x)),$$

де Qi відповідає кванторам $\forall$ або $\exists$, а $\circ$ означає $\wedge$ або $\vee$.

Також можуть використовуватися формули:

$\exists x A(x) \vee \exists x B(x) = \exists x(A(x) \vee B(x))$,

$\forall x A(x) \vee \forall x B(x) = \forall x(A(x) \vee B(x))$.

3. Приведення формули до сколемівської нормальної форми.

3.1. Видалення кванторів існування.

Будь-яка змінна, на яку розподіляється дія квантора існування, замінюється функцією від усіх поставлених перед нею змінних, на які поширюється дія квантора загальності. Якщо таких змінних немає, то ця змінна замінюється константою.

3.2. Видалення кванторів загальності.

Усі змінні тепер схильні до дії кванторів загальності, які на цьому етапі видаляються.

4. Приведення формули до каузальної нормальної форми.

Формула приводиться до вигляду:

$C1 \wedge ...Ci \wedge ...Cn$,

де Ci містить тільки логічні операції $\vee$, $\neg$.

Для цього використовуються теореми дистрибутивності:

$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$,

$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$.

Приклад алгоритму уніфікації

Дано формулу:

$(\forall x \forall y(P(y) \to \overline{Q(x,y)})) \wedge (\forall y(F(y) \to \overline{P(y)})) \to \forall x \forall y(F(y) \to Q(x,y))$.

1.1. $\overline{(\exists x \exists y(\overline{P(y)} \vee \overline{Q(x,y)})) \vee (\exists y(\overline{F(y)} \vee \overline{P(y)}))} \vee \forall x \forall y(\overline{F(y)} \vee Q(x,y))$,

1.2. $(\exists x \exists y(P(y) \wedge Q(x,y))) \vee (\exists y(F(y) \wedge P(y))) \vee \forall x \forall y(\overline{F(y)} \vee Q(x,y))$,

2.1. $(\exists x \exists y(P(y) \wedge Q(x,y))) \vee (\exists y(F(y) \wedge P(y))) \vee \forall z \forall u(\overline{F(u)} \vee Q(z,u))$,

2.3. $\exists x \exists y \forall z \forall u(P(y) \wedge Q(x,y) \vee F(y) \wedge P(y) \vee \overline{F(u)} \vee Q(z,u))$,

3.1. $\forall z \forall u(P(b) \wedge Q(a,b) \vee F(b) \wedge P(b) \vee \overline{F(u)} \vee Q(z,u))$,

3.2. $P(b) \wedge Q(a,b) \vee F(b) \wedge P(b) \vee \overline{F(u)} \vee Q(z,u)$,

4. $(P(b) \vee \overline{F(u)} \vee Q(z,u)) \wedge (Q(a,b) \vee F(b) \vee \overline{F(u)} \vee Q(z,u))$.

5.2.3. Приклади принципу резолюції

Приклад принципу резолюції без використання уніфікації

Нехай є два диз'юнкти:

C_1: P(a) $\vee$ $\neg$Q(b,c).

C_2: Q(b,c) $\vee$ $\neg$R(b,c).

Оскільки в першому диз'юнкті міститься предикат із запереченням $\neg$Q(b,c), а в другому – відповідний предикат без заперечення Q(b,c), й аргументи обох предикатів узгоджені (тобто b першого узгоджується з b другого, c першого узгоджується з c другого), то перший диз'юнкт може бути резольвовано з другим. Унаслідок резолюції виходить новий диз'юнкт (резольвента):

C_3: P(a) $\vee$ $\neg$R(b,c).

Для прикладу нижче отримана резольвента буде порожньою, оскільки:

C_1: P(a) $\vee$ $\neg Q(b,c)$,

C_2: $Q(b,c)$ $\vee$ $\neg$P(a),

C_3: □

Приклад принципу резолюції з використанням уніфікації

Довести, що фраза є логічно істинною (тавтологією): «Є студенти, які люблять усіх викладачів. Студенти не люблять невігласів. Отже, серед викладачів немає невігласів».

Виконаємо формалізацію цієї фрази. Введемо предикати:

$C(x)$ – x є студентом,

$P(y)$ – y є викладачем,

$H(y)$ – y є невігласом,

$L(x,y)$ – x любить y.

Тоді дві посилки (аксіоми) мовою числення предикатів першого порядку мають вигляд:

$\exists x(C(x) \wedge \forall y(P(y) \to L(x,y)))$,

$\forall x(C(x) \to \forall y(H(y) \to \neg L(x,y)))$.

Заключення буде виражене як $\forall y(P(y) \to \neg H(y))$.

Уся фраза подається у вигляді формули:

$$\exists x(C(x) \wedge \forall y(P(y) \to L(x,y))) \wedge \forall x(C(x) \to \forall y(H(y) \to \neg L(x,y))) \to \forall y(P(y) \to \neg H(y)).$$

Далі проводиться уніфікація відповідно до наведеного вище алгоритму.

Отримані диз'юнкти мають вигляд:

1) $C(a)$,

2) $\neg P(b) \vee L(a,b)$,

3) $\neg C(a) \vee \neg H(b) \vee \neg L(a,b)$,

4) $P(b)$,

5) $H(b)$.

Доведемо, що зазначена вище множина диз'юнктів нездійсненна:

6) $\neg H(b) \vee \neg L(a,b)$ з 1) та 3),

7) $L(a,b)$ з 2) та 4),

8) $\neg H(b)$ з 6) та 7),

9) з 5) та 8).

Зазначене виведення видається деревом виведення (рис. 5.4)

До недоліків методу резолюції належить формування всіляких компонент, більшість із яких виявляються непотрібними.

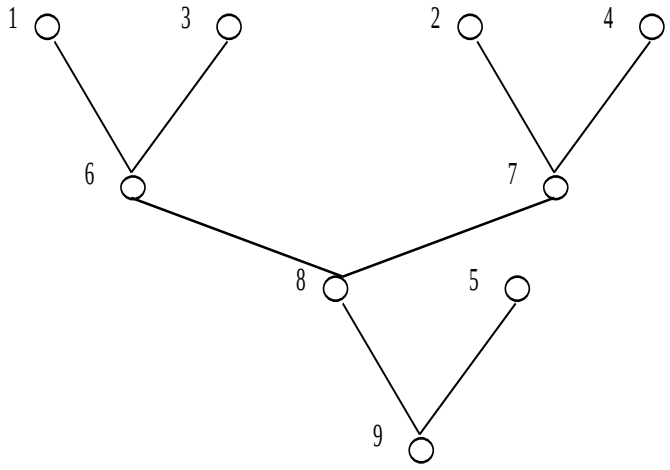


Рис. 5.4 – Дерево виведення

5.3. Виведення на знаннях у вигляді фреймової моделі

Використовуються два способи керування виведенням:

- за допомогою механізму наслідування;
- за допомогою приєднаних процедур.

Механізм успадкування є основним вбудованим засобом виведення. Він забезпечує значну економію пам'яті та автоматичне визначення значень для слотів фреймів нижніх рівнів. Наприклад, нехай дано мережу фреймів з одиночним успадкуванням (рис. 3.7) і поставлено запитання: «чи люблять учні солодке?». Оскільки фрейм «учень» успадковує всі слоти фрейму «дитина», а успадкована комірка «любить» має значення «солодке», то відповідь буде ствердною. Якщо у фрейм «учень» додати слот «любить» зі значенням «кисле», то відповідь буде негативною, оскільки значення слота «любить» у фреймі «учень» перериває значення слота «любить» у фреймі «дитина». Наприклад, у мовах об'єктно-орієнтованого програмування атрибут батьківського класу можна перевизначити у дочірньому класі.

Приєднана процедура виступає як значення слота і запускається за повідомленням, яке є функцією (містить як аргументи ім'я фрейму, ім'я слота і параметри) і, зі свого боку, може передати повідомлення іншому фрейму для запуску його приєднаної процедури. Виникає ланцюжок передач повідомлень від одного фрейму до іншого та подальше повернення відповіді в точку початкового виклику повідомлення. Наприклад, у мовах об'єктно-орієнтованого програмування викликаний метод дочірнього класу може викликати метод батьківського класу.

5.4. Виведення на знаннях у вигляді моделі семантичної мережі

Методи виведення на семантичних мережах використовують асоціативні та зіставні алгоритми, які зводяться до знаходження шляхів на графі (щоб встановити відношення між поняттями), побудови транзитивних замикань ($\alpha R\beta \wedge \beta R\gamma \rightarrow \alpha R\gamma$), витягу підграфів з певними властивостями (підграф запиту накладається на відповідний фрагмент). Наприклад, нехай дано семантичну мережу (рис. 5.5), підграфом запиту є: «який автомобіль належить Степаненко?» (рис. 5.5).

Рис. 5.5 – Семантична мережа

6. ВИВЕДЕННЯ НА НЕЧІТКИХ ЗНАННЯХ

6.1. Введення в нечіткі множини та нечітку логіку

Нечіткі (або якісні) знання застосовуються в умовах неповної інформації, коли використовуються якісні характеристики об'єктів (наприклад, багато, мало), і не можуть бути інтерпретовані як істинні або хибні (0/1). У 70-х роках XX ст. Л. Заде запропонував формальний апарат нечіткої алгебри і логіки та ввів поняття лінгвістичної змінної.

Нечітка множина – це множина, про елементи якої відомо лише те, що вони належать цій множині.

Формально нечітку множину можна визначити в такий спосіб: нехай X – універсальна множина (універсум), $\tilde{A}$ – нечітка множина, $\tilde{A} \subseteq X$. Ступінь приналежності елементів із універсуму X нечіткий множині $\tilde{A}$ визначає функцію належності $\mu_{\tilde{A}}$, яка ставить у відповідність до кожного елемента $x \in X$ число з інтервалу $[0,1]$, тобто $\mu_{\tilde{A}} : X \to [0,1]$. Тоді нечітка множина $\tilde{A}$ визначається як множина пар, кожна з яких складається з елемента $x \in X$ та його ступеня приналежності даній нечіткій множині $\mu_{\tilde{A}}(x)$, тобто $\tilde{A} = \{(x, \mu_{\tilde{A}}(x))\}$.

Для порожньої множини $\varnothing$ виконується умова:

$\forall x \in X \quad \mu_{\varnothing}(x) = 0$.

Для універсуму X виконується умова:

$\forall x \in X \quad \mu_{X}(x) = 1$.

Зазвичай використовуються такі функції належності:

1. Кусково-лінійна: Z-подібна (рис. 6.1) та S-подібна (рис. 6.2) (напівлінійна з насиченням):

$$\mu_Z(x,a,b) = \begin{cases} 1, & x \le a \\ \dfrac{b-x}{b-a}, & a < x < b \\ 0, & x \ge b \end{cases}$$

$$\mu_S(x,a,b) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a < x < b \\ 1, & x \ge b \end{cases}$$

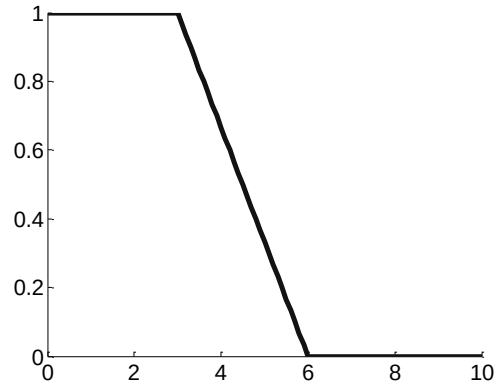


Рис. 6.1 – Кусково-лінійна Z-подібна (напівлінійна з насиченням) функція належності μ_Z

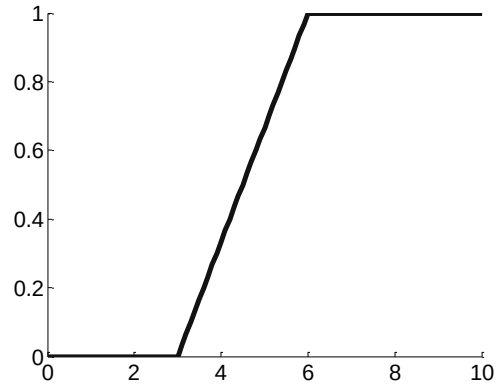


Рис. 6.2 – Кусково-лінійна S-подібна (напівлінійна з насиченням) функція належності μ_S

2. Сплайн-функція Z-подібна (рис. 6.3) і S- подібна (рис. 6.4):

$$\mu_Z(x,a,b)=\begin{cases}1, & x\le a\\ 1-2\left(\dfrac{x-a}{b-a}\right)^2, & a<x\le\dfrac{a+b}{2}\\ 2\left(\dfrac{b-x}{b-a}\right)^2, & \dfrac{a+b}{2}<x<b\\ 0, & x\ge b\end{cases};$$

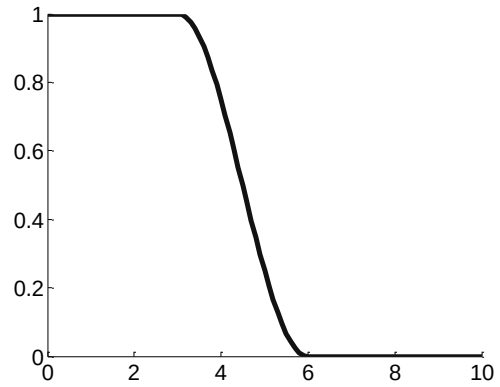


Рис. 6.3 – Сплайн Z-подібна функція належності

$$\mu_S(x,a,b)=\begin{cases}0, & x\le a\\ 2\left(\dfrac{x-a}{b-a}\right)^2, & a<x\le\dfrac{a+b}{2}\\ 1-2\left(\dfrac{b-x}{b-a}\right)^2, & \dfrac{a+b}{2}<x<b\\ 1, & x\ge b\end{cases};$$

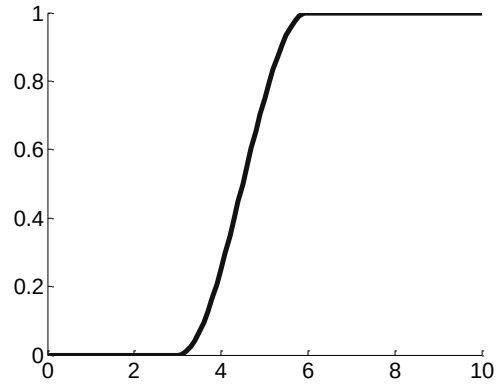


Рис. 6.4 – Сплайн S-подібна функція належності

3. Кусково-лінійна трикутна (рис. 6.5):

$$\mu_\Delta(x,a,b,c)=\begin{cases}0, & x\le a\\ \dfrac{x-a}{b-a}, & a\le x\le b\\ \dfrac{c-x}{c-b}, & b\le x\le c\\ 0, & x\ge c\end{cases}.$$

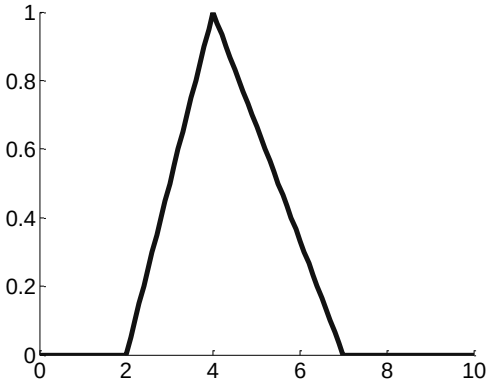


Рис. 6.5 – Кусково-лінійна трикутна функція належності

4. Кусково-лінійна П-подібна (рис. 6.6):

$$\mu_{\Pi}(x,a,b,c,d)=\begin{cases}0, & x\le a\\ \dfrac{x-a}{b-a}, & a\le x\le b\\ 1, & b\le x\le c\\ \dfrac{d-x}{d-c}, & c\le x\le d\\ 0, & x\ge d\end{cases}.$$

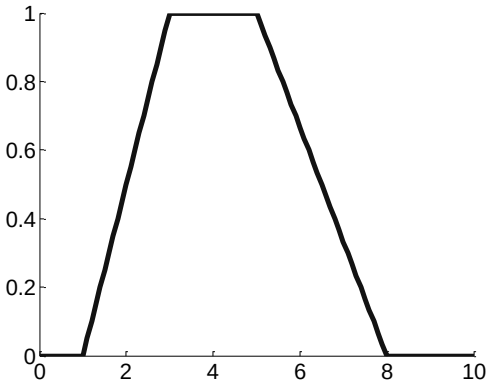


Рис. 6.6 – Кусково-лінійна П-подібна функція належності

5. Сигмоїдальна S-подібна (рис. 6.7) та Z-подібна (логістична) (рис. 6.8):

$$\mu_S(x,a,b)=\frac{1}{1+\exp(-a(x-b))}\,\text{і}\,\mu_Z(x,a,b)=\frac{1}{1+\exp(a(x-b))},a>0.$$

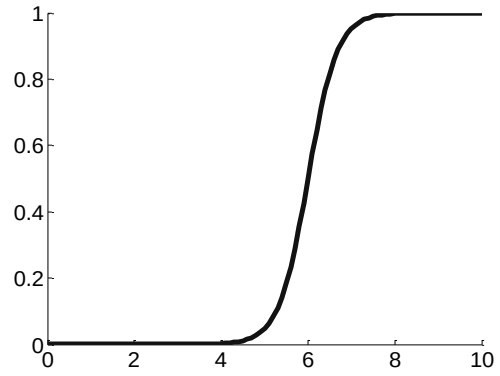


Рис. 6.7 – Сигмоїдальна S-подібна (логістична) функція належності μ_Z

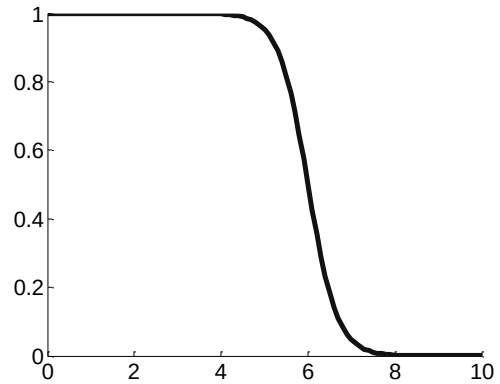


Рис. 6.8 – Сігмоїдальна Z-подібна (логістична) функція належності μ_S

6. Радіально-базова Гауса (рис. 6.9):

$$\mu_{RB}(x, m, \sigma) = \exp\left(-\frac{(x-m)^2}{2\sigma^2}\right); \text{ причому } \sqrt{2\pi}\sigma = 1.$$

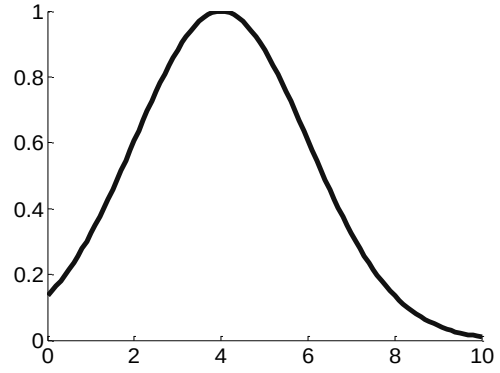


Рис. 6.9 – Радіально-базова Гаусса функція належності μ_{RB}

7. Дзвоноподібна П-подібна (рис. 6.10):

$$\mu_{\Pi}(x,a,b,c)=\frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}}.$$

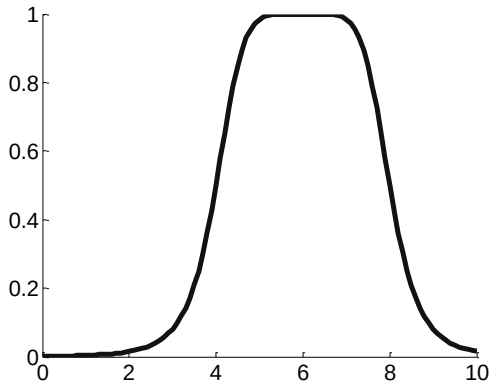


Рис. 6.10 – Дзвоноподібна П-подібна функція належності μ_{Π}

8. Двостороння функція Гауса:

$$\mu_{\Pi_3}(x,m1,\sigma1,m2,\sigma2)=\begin{cases}\mu_{\Pi_1}(x,m1,\sigma1), & x<x1_M\\ 1, & x1_M\le x\le x2_M\\ \mu_{\Pi_2}(x,m2,\sigma2), & x>x2_M\end{cases},$$

$x1_M=\min\{\arg\max_x \mu_{\Pi_1}(x,m1,\sigma1)\}$ – найбільша ліва мода;

$x2_M=\max\{\arg\max_x \mu_{\Pi_2}(x,m2,\sigma2)\}$ – найбільша права мода.

9. Добуток двох сигмоїдальних функцій:

$$\mu_{\Pi_4}(x,a1,b1,a2,b2)=\mu_{ZS}(x,a1,b1)\mu_{ZS}(x,a2,b2).$$

10. Різниця двох сигмоїдальних функцій:

$$\mu_{\Pi_5}(x,a1,b1,a2,b2)=\mu_{ZS}(x,a1,b1)-\mu_{ZS}(x,a2,b2).$$

11. Добуток Z- подібної і S-подібної сплайн-функцій:

$$\mu_{\Pi_6}(x,a1,b1,a2,b2)=\mu_{Z_1}(x,a1,b1)\mu_{S_1}(x,a2,b2),$$

$$\mu_{\Pi_7}(x,a1,b1,a2,b2)=\mu_{Z_1}(x,a1,b1)\mu_{S_2}(x,a2,b2),$$

$$\mu_{\Pi_8}(x,a1,b1,a2,b2)=\mu_{Z_2}(x,a1,b1)\mu_{S_1}(x,a2,b2),$$

$$\mu_{\Pi_9}(x,a1,b1,a2,b2)=\mu_{Z_2}(x,a1,b1)\mu_{S_2}(x,a2,b2).$$

Нечітка змінна представлена набором $(\tilde{\alpha}, X, \tilde{A})$,

де $\tilde{\alpha}$ – назва нечіткої змінної;

X – універсум (область її визначення);

$\tilde{A}$ – нечітка множина, $\tilde{A} \subseteq X$, яка визначає область значень нечіткої змінної.

Лінгвістична змінна представлена набором $(\tilde{x}, T, X, G, M)$,

де $\tilde{x}$ – назва лінгвістичної змінної;

T – базова терм-множина (множина значень) лінгвістичної змінної, що є назвами нечітких змінних, областю визначення яких є універсум X;

X – універсум;

G – синтаксична процедура, яка породжує (генерує) з множини T нові значення (терми) для лінгвістичної змінної. Множина $T \bigcup G(T)$, де $G(T)$ – множина згенерованих термів, називається розширеною терм-множиною;

M – семантична процедура, яка ставить у відповідність до кожного нового значення (терму) даної лінгвістичної змінної, отриманої за допомогою процедури G, її зміст, тобто формує нечітку множину для нового терму.

У процедуру G можуть входити правила виду:

Терм→Базовий терм | не Терм | Терм або Терм | Терм і Терм

| дуже Терм | більш-менш Терм,

які містять логічні зв'язки «НЕ», «АБО», «І» та модифікатори «дуже», «більш-менш».

Тоді для базової терм-множини $T = \{\tilde{\alpha}_1, \tilde{\alpha}_2, ...\}$, відповідно до правил процедури G, можна отримати множину згенерованих термів у вигляді:

$$G(T) = \{\textit{не } \tilde{\alpha}_1, \tilde{\alpha}_1 \textit{ або } \tilde{\alpha}_2, \tilde{\alpha}_1 \textit{ і } \tilde{\alpha}_2, \textit{дуже } \tilde{\alpha}_1, \textit{більш-менш } \tilde{\alpha}_1, \ldots\}.$$

Зазвичай терму «*не* $\tilde{\alpha}_1$» відповідає $\tilde{A} = \overline{\tilde{A}_1} = \{(x, 1 - \mu_{\tilde{A}_1}(x))\}$.

Зазвичай терму «$\tilde{\alpha}_1$ або $\tilde{\alpha}_2$» відповідає

$$\tilde{A} = \tilde{A}_1 \cup \tilde{A}_2 = \{(x, \max(\mu_{\tilde{A}_1}(x), \mu_{\tilde{A}_2}(x))\}$$

або

$$\tilde{A} = \tilde{A}_1 + \tilde{A}_2 = \{(x, \mu_{\tilde{A}_1}(x) + \mu_{\tilde{A}_2}(x) - \mu_{\tilde{A}_1}(x)\mu_{\tilde{A}_2}(x))\}.$$

Зазвичай терму «$\tilde{\alpha}_1$ *і* $\tilde{\alpha}_2$» відповідає

$$\tilde{A} = \tilde{A}_1 \cap \tilde{A}_2 = \{(x, \min(\mu_{\tilde{A}_1}(x), \mu_{\tilde{A}_2}(x))\}$$

або

$$\tilde{A} = \tilde{A}_1 \cdot \tilde{A}_2 = \{(x, \mu_{\tilde{A}_1}(x)\mu_{\tilde{A}_2}(x))\}.$$

Зазвичай терму «*дуже* $\tilde{\alpha}_1$» відповідає $\tilde{A} = \tilde{A}_1^2 = \{(x, (\mu_{\tilde{A}_1}(x))^2)\}$.

Зазвичай терму «*більш-менш* $\tilde{\alpha}_1$» відповідає

$$\tilde{A} = \tilde{A}_1^{0.5} = \{(x, (\mu_{\tilde{A}_1}(x))^{0.5})\}.$$

Приклад

Перша нечітка змінна представлена набором

$(\tilde{\alpha}_1, X, \tilde{A}_1)$,

де $\tilde{\alpha}_1$ = холодно,

$X = [-40, 40]$,

$$\tilde{A}_1 = \{(x, \mu_{\tilde{A}_1}(x))\}, \mu_{\tilde{A}_1}(x) = \frac{1}{1 + \exp(a_1(x - b_1)}, \; x \in X.$$

Друга нечітка змінна представлена набором

$(\tilde{\alpha}_2, X, \tilde{A}_2)$,

де $\tilde{\alpha}_2$ = жарко,

$X = [-40, 40]$,

$$\tilde{A}_2 = \{(x, \mu_{\tilde{A}_2}(x))\}, \mu_{\tilde{A}_2}(x) = \frac{1}{1 + \exp(-a_2(x - b_2))}, \; x \in X.$$

Лінгвістична змінна представлена набором

$(\tilde{x}, T, X, G, M)$,

де $\tilde{x}$ = температура,

$T = \{\tilde{\alpha}_1, \tilde{\alpha}_2\}$,

$X = [-40, 40]$,

$G(T) = \{\tilde{\alpha}_3\} = \{$*не* $\tilde{\alpha}_1$ *і не* $\tilde{\alpha}_2$ $\} = \{$не холодно і не жарко$\}$,

$$M(\tilde{\alpha}_3) = \{(x, \mu_{\tilde{A}_3}(x))\} = \{(x, (1 - \mu_{\tilde{A}_1}(x))(1 - \mu_{\tilde{A}_2}(x)))\}$$

або

$$M(\tilde{\alpha}_3) = \{(x, \mu_{\tilde{A}_3}(x))\} = \{(x, \min((1 - \mu_{\tilde{A}_1}(x), (1 - \mu_{\tilde{A}_2}(x)))\}.$$

6.2. Структура нечіткого виведення

На рис. 6.11 представлено структуру нечіткого виведення.

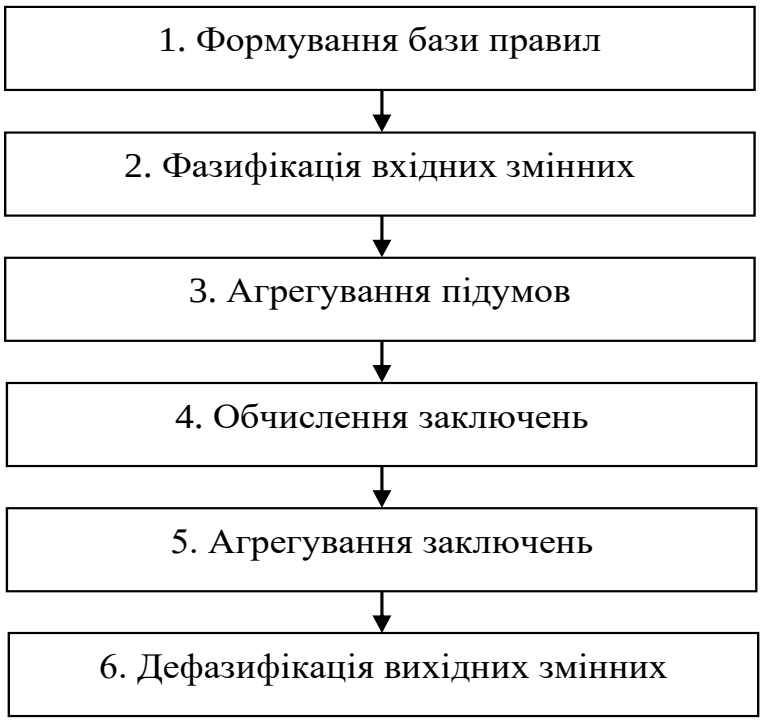


Рис. 6.11 – Структура нечіткого виведення

6.2.1. Формування бази правил

База правил систем нечіткого виведення призначена для формального представлення знань експертів у проблемній галузі у вигляді нечітких продукцій.

Загалом під нечіткою продукцією розуміється вираз такого виду:

$$(i): Q; P; \tilde{A} \Rightarrow \tilde{B}; S, F, N,$$

де (i) – ім’я продукції,

Q – передумова продукції (описує предметну область знання, яку представляє окрема продукція, у своїй декомпозиції предметна область на окремі незалежні області може значно підвищити ефективність міркувань у продукційній системі),

P – передумова застосування ядра нечіткої продукції (зазвичай у вигляді предикату),

$\tilde{A} \Rightarrow \tilde{B}$ – ядро продукції,

$\tilde{A}$ – умова ядра продукції (антецедент) (зразок, яким здійснюється пошук у БЗ),

$\tilde{B}$ – заключення ядра продукції (консеквент) (може бути проміжним і виступати далі як умова або термінальним (цільовим) і завершувати роботу системи),

S – метод визначення кількісного значення ступеня істинності заключення,

F – коефіцієнт визначеності, коефіцієнт впевненості або ваговий коефіцієнт нечіткої продукції, $F \in [0,1]$,

N – постумова продукції (описує дії та процедури, які необхідно виконати у разі реалізації ядра продукції).

Слід зазначити, що ядро продукції виду

ЯКЩО $\tilde{A}$ ТО $\tilde{B}$ ІНАКШЕ $\tilde{C}$

еквівалентне двом ядрам продукції виду

ЯКЩО $\tilde{A}$ ТО $\tilde{B}$, ЯКЩО НЕ $\tilde{A}$ ТО $\tilde{C}$.

На практиці зазвичай використовують такий окремий випадок нечітких продукцій:

ПРАВИЛО k: ЯКЩО умова, k ТО заключення k (F^k),

де умова k – це сукупність умов (нечітких висловлювань) виду

$\tilde{x}_1 \in \tilde{\alpha}_1^k$ операція … операція $\tilde{x}_n \in \tilde{\alpha}_n^k$;

заключення k – це сукупність підзаключень (нечітких висловлювань) виду

$\tilde{y}_1 \in \tilde{\beta}_1^k$ операція … операція $\tilde{y}_m \in \tilde{\beta}_m^k$.

Зазвичай вважається, що $\tilde{y}_1, \ldots, \tilde{y}_m$ взаємно незалежні, тому заключення k без втрати спільності подають у вигляді $\tilde{y} \in \tilde{\beta}^k$,

операція – це бінарна операція нечіткої кон'юнкції «І» або нечіткої диз'юнкції «АБО».

$\tilde{x}_i$ – ім'я вхідної лінгвістичної змінної, $i \in \overline{1,n}$;

$\tilde{y}_j$ – ім'я вихідної лінгвістичної змінної, $j \in \overline{1,m}$;

$\tilde{\alpha}_i^k$ – значення (терм) змінної $\tilde{x}_i, k \in \overline{1,r}$, $i \in \overline{1,n}$;

$\tilde{\beta}_j^k$ – значення (терм) змінної $\tilde{y}_j, k \in \overline{1,r}$, $j \in \overline{1,m}$;

Якщо вагові коефіцієнти відсутні, то $F^k = 1$.

Зауваження. Також заключення k може бути чітким. У цьому разі його можна уявити у вигляді $y = \varepsilon_0^k + \varepsilon_1^k x_1 + \ldots + \varepsilon_n^k x_n$,

де ε_i^k – задані вагові коефіцієнти.

Приклад

Нечітка продукція k «Якщо температура ($\tilde{x}_1$) висока ($\tilde{\alpha}_1^k$) і кількість лейкоцитів ($\tilde{x}_2$) набагато перевищує норму ($\tilde{\alpha}_2^k$), то кількість днів хвороби ($\tilde{y}$) велика ($\tilde{\beta}^k$)» можна подати у вигляді такого правила:

ПРАВИЛО k: ЯКЩО $\tilde{x}_1 \in \tilde{\alpha}_1^k$ І $\tilde{x}_2 \in \tilde{\alpha}_2^k$ ТО $\tilde{y} \in \tilde{\beta}^k$.

6.2.2. Фазифікація

Фазифікація є процедурою визначення ступеня істинності підумов нечітких продукцій. Фазифікацію також називають запровадженням нечіткості.

Метою фазифікації є встановлення відповідності між значенням вхідної чіткої змінної та значенням функції належності відповідного їй терма вхідній лінгвістичній змінній у вигляді:

$$a1_i^k = \mu_{\tilde{A}_i^k}(x_i^0), k \in \overline{1,r}, i \in \overline{1,n},$$

де $a1_i^k$ – ступінь істинності i-ї підумови $\tilde{x}_i \in \tilde{\alpha}_i^k$ k-ї нечіткої продукції,

$\mu_{\tilde{A}_i^k}$ – функції належності нечіткої множини $\tilde{A}_i^k$, яка визначає область значень нечіткої змінної $\tilde{\alpha}_i^k$,

x_i – i-а чітка вхідна змінна, $x_i \in X_i$,

$\tilde{x}_i$ – i-а лінгвістична змінна,

x_i^0 – значення i-ї чіткої вихідної змінної x_i.

6.2.3. Агрегування умов

Агрегування умов є процедурою визначення ступеня істинності умов нечітких продукцій.

Якщо умова складається з однієї умови, то

$$a2^k = a1_1^k, k \in \overline{1,r}.$$

Якщо умова складається з підумов, з'єднаних нечіткою кон'юнкцією, зазвичай використовується один із таких методів:

– метод мінімального значення

$$a2^k = \min_{i\in\overline{1,n}}\{a1_i^k\}, k \in \overline{1,r};$$

– метод алгебраїчного добутку

$$a2^k = \prod_{i=1}^{n} a1_i^k, k \in \overline{1,r}.$$

Якщо умова складається з підумов, з'єднаних нечіткою диз'юнкцією, зазвичай використовується один із таких методів:

– метод максимального значення

$$a2^k = \max_{i\in\overline{1,n}}\{a1_i^k\}, k \in \overline{1,r};$$

– метод алгебраїчної суми

$$a2^k = 1 - \prod_{i=1}^{n}(1 - a1_i^k).$$

Слід зазначити, що під час використання формул для визначення нечіткої кон'юнкції та нечіткої диз'юнкції доцільно застосовувати попарно узгоджені формули (наприклад, добуток алгебри та алгебраїчну суму або мінімальне значення і максимальне значення).

6.2.4. Активізація заключень

Спочатку ступінь істинності нечітких продукцій множиться на їх вагу:

$$a3^k = a2^k F^k, k \in \overline{1,r}.$$

Потім проводиться обчислення заключень, яке є процедурою визначення ступенів істинності заключень нечіткого виведення або значень вихідних чітких змінних.

Для першого випадку використовується такай підхід: ступені істинності заключень обчислюються за допомогою прямого нечіткого висновку у вигляді композиції:

$$\mu_{\tilde{B}'^k}(y) = \mu_{\tilde{A}'^k}(x^0) \circ \mu_{\tilde{A}^k \to \tilde{B}^k}(x^0, y),\ y \in Y,\ k \in \overline{1,r},$$

де $\mu_{\tilde{B}'^k}$ – функція належності нечіткої множини $\tilde{B}'^k$, яка представляє висновок нечіткого висновку і загалом не збігається з нечіткою множиною $\tilde{B}^k$, яке є висновком нечіткої імплікації,

$\mu_{\tilde{A}'^k}$ – функція належності нечіткого відношення $\tilde{A}'^k = \tilde{A}_1'^k \times ... \times \tilde{A}_n'^k$, яке є посилкою нечіткого висновку та загалом не збігається з нечітким відношенням $\tilde{A}^k = \tilde{A}_1^k \times ... \times \tilde{A}_n^k$, яке є посилкою (умовою) нечіткої імплікації,

$\mu_{\tilde{A}^k \to \tilde{B}^k}$ – функція належності нечіткого відношення $\tilde{A}^k \to \tilde{B}^k$, яке є нечіткою імплікацією (продукцією),

y – чітка вихідна змінна.

Для спрощення обчислень приймається, що функція приналежності кожної нечіткої множини $\tilde{A}_i'^k$ визначена у вигляді:

$$\mu_{\tilde{A}_i'^k}(u_i) = \begin{cases} 1, & u_i = x_i^0 \\ 0, & u_i \neq x_i^0 \end{cases},$$

тобто $\tilde{A}_i'^k$ є синглетоном (одноелементною множиною), а для композиції $\mu_{\tilde{B}'^k}(y) = \mu_{\tilde{A}'^k}(x^0) \circ \mu_{\tilde{A}^k \to \tilde{B}^k}(x^0, y)$ використовується метод мінімального значення або алгебраїчного добутку. Тоді $\mu_{\tilde{B}'^k}(y) = \mu_{\tilde{A}^k \to \tilde{B}^k}(x^0, y)$, $y \in Y$, $k \in \overline{1,r}$.

Для обчислення ступеня істинності висновків нечіткого висновку використовується один із таких методів:

– метод мінімального значення (правило Мамдані)

$$\mu_{\tilde{B}'^k}(y) = \mu_{\tilde{A}'^k \to \tilde{B}^k}(x^0, y) = min\ (a3^k, \mu_{\tilde{B}^k}(y)), y \in Y, k \in \overline{1,r};$$

– метод алгебраїчного добутку (правило Ларсена)

$$\mu_{\tilde{B}'^k}(y) = \mu_{\tilde{A}'^k \to \tilde{B}^k}(x^0, y) = a3^k \mu_{\tilde{B}^k}(y), y \in Y, k \in \overline{1,r},$$

де $\mu_{\tilde{B}^k}$ – функція належності нечіткої множини $\tilde{B}^k$, яка визначає область значень нечіткої змінної $\tilde{\beta}^k$,

$\mu_{\tilde{C}^k}$ – функції належності нечіткої множини $\tilde{C}^k$, $\tilde{C}^k = \{(y, \mu_{\tilde{C}^k}(y))\}$,

y – чітка вихідна змінна, $y \in Y$.

Для другого випадку використовується один із таких методів:

– метод зворотної функції (якщо $a3^k = \mu_{\tilde{B}^k}(z^k)$ і $\mu_{\tilde{B}^k}$ монотонна функція)

$$z^k = \mu_{\tilde{B}^k}^{-1}(a3^k), k \in \overline{1,r};$$

– метод зваженої суми (якщо заключення подано у вигляді $y = \varepsilon_0^k + \varepsilon_1^k x_1 + ... + \varepsilon_n^k x_n$)

$$z^k = \varepsilon_0^k + \sum_{i=1}^{n} \varepsilon_i^k x_i, k \in \overline{1,r}.$$

6.2.5. Агрегування заключень

Агрегування заключень є процедурою визначення ступенів істинності підсумкових висновків (агрегуються висновки з однаковими вихідними лінгвістичними змінними). Зазвичай використовується один із таких методів:

– метод максимального значення

$$\mu_{\tilde{B}'}(y) = \max_{k \in \overline{1,r}} \{\mu_{\tilde{B}'^k}(y)\},\ y \in Y;$$

– метод алгебраїчної суми

$$\mu_{\tilde{B}'}(y) = 1 - \prod_{k=1}^{r} (1 - \mu_{\tilde{B}'^k}(y)),\ y \in Y.$$

Якщо на етапі обчислення заключень було обчислено чіткі значення вихідних лінгвістичних змінних заключення, то агрегування нечітких заключень відсутнє.

6.2.6. Дефазифікація

Дефазифікація є процедурою визначення значення вихідної чіткої змінної. Дефазифікацію також називають приведенням до чіткості. На етапі обчислення заключень значення чітких змінних не обчислені або обчислені.

Для першого випадку вважається, що z^k – центр нечіткої множини $\tilde{C}^k$ (функція належності $\mu_{\tilde{C}^k}$ приймає в z^k максимальне значення), і зазвичай використовується один із таких методів:

– метод центру ваги

$$y^* = \frac{\sum_{y \in Y} \mu_{\tilde{B}'}(y) y}{\sum_{y \in Y} \mu_{\tilde{B}'}(y)};$$

– метод центру площі

$$y^* = \arg\min_{y \in Y} \left| \sum_{u \in \{\min Y, \ldots, y\}} \mu_{\tilde{B}'}(u) - \sum_{u \in \{y, \ldots, \max Y\}} \mu_{\tilde{B}'}(u) \right|;$$

– метод лівого максимуму

$$y^* = \min V;$$

– метод правого максимуму

$$y^* = \max V;$$

– метод середнього максимуму

$$y^* = \frac{1}{|V|} \sum_{i=1}^{|V|} v_i,$$

де $V = \left\{ v \mid v = z^{k^*} = \arg\max_k \mu_{\tilde{C}}(z^k) \right\}$ – множина мод нечіткої множини $\tilde{B}'$.

Для другого випадку зазвичай використовується один із таких методів:

– метод зваженої суми

$$y^* = \sum_{k=1}^{r} a3^k z^k;$$

– метод зваженого середнього

$$y^* = \frac{\sum_{k=1}^{r} a3^k z^k}{\sum_{k=1}^{r} a3^k}.$$

6.3. Основні алгоритми нечіткого виведення

Зазвичай розглядають чотири алгоритми, причому $r = 2, n = 2, m = 1.$

Алгоритм Мамдані

1. Формування основи правил. Заключення представлено у вигляді $\tilde{y} \in \tilde{\beta}^k$.
2. Фазифікація. Збігається з описаним вище етапом.

3. Агрегування умов. Використовується метод мінімального чи максимального значення.

4. Обчислення заключень. Використовується метод мінімального значення.

5. Агрегування заключень. Використовується спосіб максимального значення.

6. Дефазифікація. Використовується метод центру ваги, центру площі, лівого модального значення, правого модального значення або середнього максимуму.

Алгоритм Ларсена

1. Формування основи правил. Заключення представлено у вигляді $\tilde{y} \in \tilde{\beta}^k$.

2. Фазифікація. Збігається з описаним вище етапом.

3. Агрегування умов. Використовується метод мінімального чи максимального значення.

4. Обчислення заключень. Використовується метод алгебраїчного добутку.

5. Агрегування заключень. Використовується спосіб максимального значення.

6. Дефазифікація. Використовується метод центру ваги, центру площі, лівого модального значення, правого модального значення або середнього максимуму.

Алгоритм Цукамото

1. Формування основи правил. Заключення представлено у вигляді $\tilde{y} \in \tilde{\beta}^k$.

2. Фазифікація. Збігається з описаним вище етапом.

3. Агрегування умов. Використовується метод мінімального чи максимального значення.

4. Обчислення заключень. Знаходяться чіткі значення кожної з лінгвістичних змінних шляхом зворотної функції.

5. Агрегування заключень. Відсутнє.

6. Дефазифікація. Використовується метод виваженої середньої чи виваженої суми.

Алгоритм Сугено

1. Формування основи правил. Заключення представлено у вигляді $y = \varepsilon_0^k + \varepsilon_1^k x_1 + ... + \varepsilon_n^k x_n$.

2. Фазифікація. Збігається з описаним вище етапом.

3. Агрегування умов. Використовується метод мінімального чи максимального значення.

4. Обчислення заключень. Знаходяться чіткі значення кожної з лінгвістичних змінних шляхом зваженої суми.

5. Агрегування заключень. Відсутнє.

6. Дефазифікація. Використовується метод виваженої середньої чи виваженої суми.

Приклад

Розглянемо приклад ухвалення рішення про кількість днів хвороби на основі двох ознак: температура та кількість лейкоцитів.

Визначимо такі змінні:

x_1 – перша вхідна змінна (температура),

x_2 – друга вхідна змінна (кількість лейкоцитів),

y – вихідна змінна (кількість днів хвороби).

$\tilde{x}_1$ – перша вхідна лінгвістична змінна (температура) із терм-множиною $\{\tilde{\alpha}_1^1, \tilde{\alpha}_1^2, \tilde{\alpha}_1^3\}$,

$\tilde{x}_2$ – друга вхідна лінгвістична змінна (кількість лейкоцитів) із терм-множиною $\{\tilde{\alpha}_2^1, \tilde{\alpha}_2^2, \tilde{\alpha}_2^3\}$,

$\tilde{y}$ – друга вхідна лінгвістична змінна (кількість лейкоцитів) із терм-множиною $\{\tilde{\beta}^1, \tilde{\beta}^2, \tilde{\beta}^3\}$.

Використовуємо алгоритм Мамдані

1. Формування бази правил

Нечітка продукція 1 «Якщо температура ($\tilde{x}_1$) висока ($\tilde{\alpha}_1^k$) і кількість лейкоцитів ($\tilde{x}_2$) значно перевищує норму ($\tilde{\alpha}_2^k$), то кількість днів хвороби ($\tilde{y}$) велика ($\tilde{\beta}^k$)» представлена в такому вигляді:

ПРАВИЛО 1: ЯКЩО $\tilde{x}_1$ є $\tilde{\alpha}_1^1$ І $\tilde{x}_2$ є $\tilde{\alpha}_2^1$ ТО $\tilde{y}$ є $\tilde{\beta}^1$ з вагою $F^1 = 1$.

Нечітка продукція 2 «Якщо температура ($\tilde{x}_1$) не дуже висока ($\tilde{\alpha}_1^2$) і кількість лейкоцитів ($\tilde{x}_2$) не дуже значно перевищує норму ($\tilde{\alpha}_2^2$), то кількість днів хвороби ($\tilde{y}$) не дуже велика ($\tilde{\beta}^2$)» представлена в такому вигляді:

ПРАВИЛО 2: ЯКЩО $\tilde{x}_1$ є $\tilde{\alpha}_1^2$ І $\tilde{x}_2$ є $\tilde{\alpha}_2^2$ ТО $\tilde{y}$ є $\tilde{\beta}^2$ з вагою $F^2 = 1$.

Нечітка продукція 3 «Якщо температура ($\tilde{x}_1$) невисока ($\tilde{\alpha}_1^3$) і кількість лейкоцитів ($\tilde{x}_2$) слабо перевищує норму ($\tilde{\alpha}_2^3$), то перевищує норму ($\tilde{y}$) мале ($\tilde{\beta}^3$)» представлена в такому вигляді.

ПРАВИЛО 3: ЯКЩО $\tilde{x}_1$ є $\tilde{\alpha}_1^3$ І $\tilde{x}_2$ є $\tilde{\alpha}_2^3$ ТО $\tilde{y}$ є $\tilde{\beta}^3$ з вагою $F^3 = 1$.

2. Фазифікація значень x_1^0 та x_2^0 чітких змінних x_1 та x_2:

$$a1_1^1 = \mu_{\tilde{A}_1^1}(x_1^0) = \frac{1}{1+\exp(-c1_1(x_1^0 - b1_1)}},$$

$$a1_2^1 = \mu_{\tilde{A}_2^1}(x_2^0) = \frac{1}{1+\exp(-c1_2(x_2^0 - b1_2))},$$

$$a1_1^2 = \mu_{\tilde{A}_1^2}(x_1^0) = \exp\left(-\frac{(x_1^0 - m2_1)^2}{2(\sigma 2_1)^2}\right),$$

$$a1_2^2 = \mu_{\tilde{A}_2^2}(x_2^0) = \exp\left(-\frac{(x_2^0 - m2_2)^2}{2(\sigma 2_2)^2}\right),$$

$$a1_1^3 = \mu_{\tilde{A}_1^3}(x_1^0) = \frac{1}{1+\exp(c3_1(x_1^0 - b3_1))},$$

$$a1_2^3 = \mu_{\tilde{A}_2^3}(x_2^0) = \frac{1}{1+\exp(c3_2(x_2^0 - b3_2))}.$$

3. Агрегування підумов (метод мінімального значення)

$$a2^1 = \mu_{\tilde{A}^1}(x^0) = \min\{a1_1^1, a1_2^1\},$$

$$a2^2 = \mu_{\tilde{A}^2}(x^0) = \min\{a1_1^2, a1_2^2\},$$

$$a2^3 = \mu_{\tilde{A}^3}(x^0) = \min\{a1_1^3, a1_2^3\},$$

де

$$\tilde{A}^k = \tilde{A}_1^k \times \tilde{A}_2^k,\ x^0 = (x_1^0, x_2^0).$$

4. Обчислення заключень

На початку ступені істинності умов нечітких продукцій множаться з їх ваги:

$$a3^1 = a2^1 F^1,$$
$$a3^2 = a2^2 F^2,$$
$$a3^3 = a2^3 F^3.$$

Функції належності $\mu_{\tilde{B}^k}$ представлені у вигляді:

$$\mu_{\tilde{B}^1}(y) = \frac{1}{1+\exp(-c1(y-b1))},$$
$$\mu_{\tilde{B}^2}(y) = \exp\left(-\frac{(y-m2)^2}{2(\sigma 2)^2}\right),$$
$$\mu_{\tilde{B}^3}(y) = \frac{1}{1+\exp(c3(y-b3))}.$$

Використовується метод мінімального значення (правило Мамдані):

$$\mu_{\tilde{B}'^1}(y) = \mu_{\tilde{A}^1 \to \tilde{B}^1}(x^0, y) = \min(a3^1, \mu_{\tilde{B}^1}(y)),\ y \in Y,$$
$$\mu_{\tilde{B}'^2}(y) = \mu_{\tilde{A}^2 \to \tilde{B}^2}(x^0, y) = \min(a3^2, \mu_{\tilde{B}^2}(y)),\ y \in Y,$$
$$\mu_{\tilde{B}'^3}(y) = \mu_{\tilde{A}^3 \to \tilde{B}^3}(x^0, y) = \min(a3^3, \mu_{\tilde{B}^3}(y)),\ y \in Y.$$

5. Агрегування висновків (метод максимального значення):

$$\mu_{\tilde{B}'}(y) = \max\{\mu_{\tilde{B}'^1}(y), \mu_{\tilde{B}'^2}(y), \mu_{\tilde{B}'^3}(y)\},\ y \in Y.$$

6. Дефазифікація (метод центру тяжкості):

$$y^* = \frac{\sum_{y \in Y} \mu_{\tilde{B}'}(y) y}{\sum_{y \in Y} \mu_{\tilde{B}'}(y)}.$$

7. ВИВЕДЕННЯ НА НЕТОЧНИХ ЗНАННЯХ

7.1. Введення у виведення на неточних знаннях

Дані та знання, з якими доводиться мати справу в ЕС, рідко бувають абсолютно точними і достовірними. Будемо називати висловлювання *неточним*, якщо його істинність або хибність не може бути встановлена з певністю. Основоположним поняттям для виведення на неточних знаннях є поняття ймовірності.

Модель оперування з неточними даними та знаннями містить два складники: мова представлення неточності та механізм виведення на неточних знаннях. Для побудови мови необхідно вибрати форму подання неточності (скаляр, інтервал, розподіл, лінгвістичне вираження, множина) та передбачити можливість приписування міри неточності всім висловлюванням.

Механізми оперування з неточними висловлюваннями можна розділити на два типи. До *першого типу* відносять механізми, що носять «приєднаний» характер, тобто виведення ведеться на точних висловлюваннях, а перерахунок мір неточності супроводжує процес виведення. Для *другого типу* характерною є наявність схем виведення, спеціально орієнтованих на використовувану мову представлення неточності. На відміну від першого типу, механізми цього типу застосовні не тільки до знань, представлених у формі правил.

Для розробки моделі першого типу (приєднаної) необхідно налаштувати установки перерахунку, що обчислюють:

а) міру неточності x у лівій частині правила (антецедента) щодо мір неточності x_i складників його висловлювань, $x = f(x_1,...,x_n)$;

б) міру неточності y у правій частині правила (консеквента) щодо мір неточності правила r та посилки правила x, $y = h(r,x)$;

в) об’єднану міру неточності висловлювання A за мірами, отриманим з правил, консеквентом яких є A.

Початок розвитку проблематики виведення на неточних знаннях пов’язаний з першими ЕС – MYCIN та PROSPECTOR. Для них формою подання неточності був скаляр – найпростіша форма, а механізм виведення належав до першого типу. Слід зазначити, що жорсткість скалярної форми призводить або до використання механізму другого типу, або до її зміни (наприклад, в ЕС INFERNO).

7.2. Байєсівський підхід

До байєсівського підходу належить механізм виведення в ЕС PROSPECTOR і схему Перла.

Імовірнісні методи прийняття рішень засновані на байєсівському підході. За теоремою Байєса:

$$P(H_i \mid E) = \frac{P(E \mid H_i)P(H_i)}{P(E)} = \frac{P(E \mid H_i)P(H_i)}{\sum_{k=1}^{n} P(E \mid H_k)P(H_k)}, i \in \overline{1,n}.$$

У разі двох несумісних гіпотез H та $\overline{H}$

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)} = \frac{P(E \mid H)P(H)}{P(E \mid H)P(H) + P(E \mid \overline{H})(1 - P(H))}.$$

Приклад

Нехай H позначає деяке захворювання, E – симптом.

Дано: M_H – кількість жителів певного регіону, що мають захворювання H,

M_E – кількість жителів, у яких спостерігається симптом E,

$M_{E|H}$ – кількість хворих жителів із симптомом E,

M – кількість жителів усього регіону.

Тоді

$$P(H) = \frac{M_H}{M}, P(E) = \frac{M_E}{M}, P(E \mid H) = \frac{M_{E|H}}{M_H}, P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}.$$

7.2.1. Механізм виведення в ЕС PROSPECTOR

В ЕС PROSPECTOR, що застосовується в геології для пошуку мінералів як вагові коефіцієнти, що призначаються висловлюванням (фактам і гіпотезам), виступають ймовірності відповідних подій. Множини подій організовані в мережі виведення, за якими проводиться перерахунок апріорних ймовірностей висловлювань в апостеріорні. Правила в системі мають вигляд «якщо E, то H» (або $P(H \mid E)$) і «якщо $\overline{E}$, то H» (або $P(H \mid \overline{E})$), причому в мережі виведення для кожної пари висловлювань $\{E, H\}$ ці правила присутні або відсутні одночасно.

В ЕС PROSPECTOR від теореми Байєса переходять до апостеріорних шансів. Для цього вводяться два відношення, які називаються *відношеннями правдоподібності*, – фактор достатності та фактор необхідності.

Фактор достатності – відношення ймовірності отримання факту E за умови, що гіпотеза H правдива, до ймовірності отримання факту E за умови, що гіпотеза H хибна:

$$D_E = \frac{P(E \mid H)}{P(E \mid \overline{H})} \text{ або } D_E = \frac{P(H \mid E)}{1 - P(H \mid E)} \frac{1 - P(H)}{P(H)},\ D_E \in [1, \infty).$$

Фактор необхідності – відношення ймовірності неодержання факту E за умови, що гіпотеза H правдива, до ймовірності неодержання факту E за умови, що гіпотеза H хибна:

$$N_E = \frac{P(\overline{E} \mid H)}{P(\overline{E} \mid \overline{H})},\ N_E \in [0,1].$$

Ці фактори пов’язані між собою співвідношенням:

$$N_E = \frac{1 - D_E P(E \mid H)}{1 - P(E \mid \overline{H})}.$$

Фактор D_E використовується, коли факт E правдивий, фактор N_E використовується, коли факт E хибний.

Апостеріорні шанси на користь гіпотези H представлені у вигляді:

$$O(H \mid E) = D_E O(H),\ O(H) = \frac{P(H)}{1 - P(H)},$$

де $O(H)$ – апріорні шанси на користь гіпотези H.

Апостеріорні шанси на користь гіпотези H, за умови, що факт хибний E, представлені у вигляді:

$$O(H \mid \overline{E}) = N_E O(H).$$

На практиці гіпотеза H може підтверджуватися не одним фактом, а кількома.

Наприклад, якщо реалізувати факт E_1 і факт E_2, то

$$D_{E_1 \cap E_2} = D_{E_1} D_{E_2}$$

або

$$D_{E_1 \cap E_2} = \min(D_{E_1}, D_{E_2}),$$

а у випадку *n* фактів

$$D_{E_1 \cap \ldots \cap E_n} = \prod_{i=1}^{n} D_{E_i}$$

або

$$D_{E_1 \cap \ldots \cap E_n} = \min(D_{E_1}, \ldots, D_{E_n}).$$

Наприклад, якщо реалізувати факт E_1 або факт E_2, то

$$D_{E_1 \cup E_2} = 1 - (1 - D_{E_1})(1 - D_{E_2}),$$
$$D_{E_1 \cup E_2} = \max(D_{E_1}, D_{E_2}),$$

а у випадку *n* фактів

$$D_{E_1 \cup \ldots \cup E_n} = 1 - \prod_{i=1}^{n} (1 - D_{E_i})$$

або

$$D_{E_1 \cup \ldots \cup E_n} = \max(D_{E_1}, \ldots, D_{E_n}).$$

Потім правила активуються.

Наприклад, якщо реалізувати факт E_1 і факт E_2 на користь гіпотези *H*, то

$$O(H \mid E_1 \cap E_2) = D_{E_1 \cap E_2} O(H),$$

а у випадку *n* фактів

$$O(H \mid E_1 \cap \ldots \cap E_n) = D_{E_1 \cap \ldots \cap E_n} O(H).$$

Наприклад, якщо реалізувати факт E_1 або факт E_2 на користь гіпотези *H*, то

$$O(H \mid E_1 \cup E_2) = D_{E_1 \cup E_2} O(H),$$

а у випадку *n* фактів

$$O(H \mid E_1 \cup ... \cup E_n) = D_{E_1 \cup ... \cup E_n} O(H).$$

На практиці одне заключення (консеквент) може бути підтверджене не одним правилом, а кількома, тому необхідно провести агрегування правил з однаковими заключеннями.

Наприклад, якщо одне заключення підтримується двома правилами, то

$$O = 1 - (1 - O_1)(1 - O_2)$$

або

$$O = \max(O_1, O_2),$$

а у випадку *n* фактів

$$O_1 \cup ... \cup O_n = 1 - \prod_{i=1}^{n} (1 - O_i)$$

або

$$O_1 \cup ... \cup O_n = \max(O_1, ..., O_n).$$

Якщо факти *E* не підтверджуються зі ймовірністю 1, що часто буває на практиці (наприклад, неточні відповіді користувача), то необхідно враховувати ненадійність фактів. Відповідно до цього вводиться $P(E \mid E')$ – ймовірність підтвердження факту *E* фактом E'. Тоді апостеріорна ймовірність гіпотези *H* обчислюється у вигляді:

$$P(H \mid E') = P(H \mid \overline{E}) + \frac{P(H) - P(H \mid \overline{E})}{P(E)} P(E \mid E'), \text{ якщо } 0 \le p(E \mid E') \le p(E),$$

$$P(H \mid E') = P(H) + \frac{P(H \mid E) - P(H)}{1 - P(E)} (P(E \mid E') - P(E)), \text{ якщо } P(E) \le P(E \mid E') \le 1,$$

де $P(E) = P(E \mid H)P(H) + P(E \mid \overline{H})(1 - P(H))$.

В ЕС PROSPECTOR користувач міг замість ймовірності $P(E \mid E')$ ввести коефіцієнти (фактори) впевненості, що лежать у діапазоні від –5 до 5. Надалі вони перетворюються у ймовірності $P(E \mid E')$.

Апостеріорні шанси на користь гіпотези *H*, за умови, що факт правдивий E', представлені у вигляді:

$$O(H \mid E') = D_{E'} O(H),\ D_{E'} = \frac{P(H \mid E')}{1 - P(H \mid E')} \frac{1 - P(H)}{P(H)}.$$

Приклад

У табл. 7.1 містяться дані за 100 померлими людьми, з яких 44 прожили 75 і більше років, і 56 осіб не дожили до 75 років. Причому зазначено, хто з них був курцем, а хто – ні.

Таблиця 7.1

Дані до прикладу

Ставлення до паління	**Тривалість життя**		**всього**
	> 75 років	**75 років або менше**	
Палять (осіб)	20	33	53
Не палять (осіб)	24	23	47
УСЬОГО	44	56	100

Апріорні шанси в цій вибірці зі 100 випадків на користь того, що людина проживе більше 75 років, визначені у вигляді:

O (Довгожитель) = 44/56 = 0.786.

Фактор достатності визначено у вигляді:

$D_{\text{не палить}}$ = P (Палить | Довгожитель) / P (Палить | Не довгожитель) = (24/44) / (23/56) = 1.328.

Фактор необхідності визначено у вигляді:

$N_{\text{не палить}}$ = P (Палить | Довгожитель) / P (Палить | Не довгожитель) = (20/44) / (33/56) = 0.882.

Апостеріорні шанси на користь того, що той, хто не палить, є довгожителем, визначений у вигляді:

O (Довгожитель | Палить) = $D_{\text{не палить}}$ O (Довгожитель) = 1.328 * 0.786 = 1.044.

Апостеріорні шанси на користь того, що той, хто палить, є довгожителем, визначений у вигляді:

O (Довгожитель | Палить) = $N_{\text{не палить}}$ O (Довгожитель) = 0.882 * 0.786 = 0.693.

Перевагами цього підходу є:

1) хороший теоретичний фундамент.

2) після багаторазового застосування теореми Байєса вплив усіх вихідних припущень на результат стає мінімальним. Тому, хоча апріорні ймовірності визначаються наближено, їх співвідношення обчислюється досить надійно.

Недоліками цього підходу є:

1) для всіх гіпотез H_i мають бути відомі ймовірності $P(H_i)$ (або шанси $O(H_i)$) та умовні ймовірності $P(\overline{E} \mid H_i), P(\overline{E} \mid \overline{H}_i), P(E \mid H_i), P(E \mid \overline{H}_i)$;

2) з умов теореми Байєса слідує, що все гіпотези H_i повинні бути несумісні, а умовні ймовірності $P(E \mid H_i)$ та ймовірності $P(E)$ – незалежні. Ця вимога часто не виконується.

3) під час введення нової гіпотези необхідно заново перераховувати таблиці ймовірностей, що також обмежує можливість застосування цього підходу.

7.2.2. Схема Перла (байєсівські мережі довіри)

Байєсівські методи виведення на неточних знаннях отримали більш глибокий розвиток у роботах Перла. У нього знання людей проблемної області подаються у вигляді спільного розподілу ймовірностей $P(x_1,...,x_n)$, водночас значеннями змінних $x_1,...,x_n$ є висловлювання. Ліва (факт) та права (гіпотеза) частини правила (E та H) складені з елементарних висловлювань (значень x_i).

Нехай кожній змінній відповідає вершина графа V_j^i. Тоді $P(x_1,...,x_n) = P(x_n \mid x_{n-1},...,x_1)...P(x_2 \mid x_1)P(x_1)$. Нехай S_i – мінімальна підмножина вершин множини $\{x_1,...,x_{i-1}\}$, $S_i \subseteq \{x_1,...,x_{i-1}\}$, що задовольняє умову $P(x_i \mid S_i) = P(x_i \mid x_{i-1},...,x_1)$. Підмножина S_i є єдиною. Для побудови графа від вершин S_i проводяться спрямовані дуги до кожної вершини x_i. Побудований граф називається *байєсівською мережею довіри*. Внаслідок виведення кожній вершині приписується апостеріорна ймовірність $Bl(V_j^i) = P(V_j^i \mid D)$, де D – факти, які надійшли. Механізм перерахунку ймовірностей має таку структуру: з кожною вершиною в мережі асоційований процесор, який отримує повідомлення від пов'язаних із ним дуг сусідніх процесорів, здійснює перерахунок $Bl(V_j^i)$ для всіх можливих значень V_j^i (або x_i) і посилає сусіднім вершинам відповідні повідомлення. Діяльність процесора ініціюється порушенням умов узгодженості зі станами сусідніх процесорів і триває до відновлення цих умов.

7.3. Метод коефіцієнтів впевненості

Метод коефіцієнтів (факторів) впевненості був вперше застосований в ЕС MYCIN, яка використовується в медицині. На відміну від байєсівського підходу, який використовує теорію ймовірностей для підтвердження гіпотез, цей метод базується на евристичних міркуваннях, які були запозичені з практичного досвіду роботи експертів. Коли експерт оцінює ступінь достовірності певного факту, він використовує поняття «точно», «можливо» та ін. В ЕС MYCIN ці нечіткі поняття відображаються на шкалу [–1,1]. Отже, висловлюванням приписуються вагові коефіцієнти, названі *коефіцієнтами впевненості*. Коефіцієнт впевненості факту визначається у вигляді $CF(E)$:

$$CF(E) = MB(E) - MD(E),\ 0 \le MB(E),\ MD(E) \le 1,\ -1 \le CF(E) \le 1,$$

де $MB(E)$ – ступінь істинності (міра зростання впевненості) факту *E*,

$MD(E)$ – ступінь помилковості (міра зростання невпевненості) факту *E*.

Для агрегування умов над фактами, складовими передумови правила (антецедент) виконуються логічні операції. В цьому разі коефіцієнти впевненості обчислюються так:

1. За умови логічного зв'язку І між фактами E_1 та E_2

$$CF(E_1 \wedge E_2) = \min(CF(E_1), CF(E_2)).$$

2. За умови логічного зв'язку АБО між фактами E_1 та E_2

$$CF(E_1 \vee E_2) = \max(CF(E_1), CF(E_2)).$$

Для активізації правил в ЕС MYCIN коефіцієнти впевненості приписуються не тільки фактам, а й правилам. У такий спосіб забезпечується облік ненадійності правил, які часто формуються на основі евристичних міркувань. Коефіцієнт впевненості заключення (консеквента) $CF(B)$ визначається добутком коефіцієнтів впевненості антецедента $CF(A)$ та коефіцієнта впевненості правила $CF(R)$:

$$CF(B) = CF(A)CF(R).$$

Слід зазначити, що $CF(B)$ можна уявити через ймовірності у вигляді

$$CF(B)=\begin{cases}\dfrac{P(H\mid E)-P(H)}{1-P(H)}, & P(H\mid E)\ge P(H)\\ \dfrac{P(H\mid E)-P(H)}{P(H)}, & P(H\mid E)\le P(H)\end{cases}.$$

На практиці одне заключення (консеквент) може підтверджуватися не одним правилом, а кількома, тому необхідно провести агрегування правил з однаковим заключенням.

Наприклад, якщо одне заключення підтверджується двома правилами, то в ЕС EMYCIN

$$CF=\begin{cases}CF_1+CF_2-CF_1CF_2, & CF_1>0, CF_2>0\\ CF_1+CF_2+CF_1CF_2, & CF_1>0, CF_2>0\\ \dfrac{CF_1+CF_2}{1-\min(|CF_1|,|CF_2|)}, & CF_1CF_2\le 0, CF_1\ne\pm1, CF_2\ne\pm1\\ 1, & CF_1=\pm1, CF_2=\pm1\end{cases}.$$

Коефіцієнти впевненості заключень, які підтверджуються трьома і більше правилами, можна вивести послідовно, застосовуючи цю формулу.

Приклад

Нехай E_1, E_2, E_3 – симптоми захворювання (наприклад, E_1 – висока температура, E_2 – почервоніння горла, E_3 – нежить) та вже обчислені їх коефіцієнти впевненості, тобто $CF(E_1) = 0.8$, $CF(E_2) = 0.6$, $CF(E_3) = 0.7$.

Нехай антецедент представлений у вигляді $A=(E_1\wedge E_2)\vee E_3$.

Тоді коефіцієнт впевненості антецедента обчислюється у вигляді

$$CF(A)=CF((E_1\wedge E_2)\vee E_3)=\max(\min(CF(E_1),CF(E_2)),CF(E_3)) = 0.7.$$

Нехай обчислений коефіцієнт впевненості правила, тобто $CF(R) = 0.9$.

Тоді коефіцієнт впевненості консеквента обчислюється у вигляді

$$CF(B)=CF(A)\cdot CF(R)=0.7*0.9=0.62.$$

Перевагами методу є:

а) немає потреби у завжди апріорній жорсткій фіксації мереж виведення: правила разом із приписаними їм коефіцієнтами впевненості можна змінювати, видаляти або додавати в базу знань, не піклуючись про забезпечення необхідної для перерахунку інформації;

б) немає потреби у завданнях апріорних ймовірностей, оскільки *CF, MB, MD* акумулюють і апріорну, і апостеріорну інформацію.

Недоліки методу пов'язані з відсутністю теоретичного фундаменту, а також складністю підбору коефіцієнтів впевненості. До того ж відмова від явного використання апріорних ймовірностей у процесі перерахунку може призводити до небажаних наслідків, як-от прийняття гіпотези з меншою апостеріорною ймовірністю.

7.4. Теорія Демпстера–Шефера (ТДШ)

Теорії Демпстера–Шефера (ТДШ) була розроблена з метою узагальнення імовірнісного підходу до опису неточності.

На відміну від байєсівського підходу і методу коефіцієнтів впевненості, в ТДШ використовується не точкова оцінка впевненості (ймовірність або коефіцієнт впевненості), а інтервальна оцінка. Така оцінка характеризується нижньою і верхньою межею, що більш надійно.

Розглянемо кінцеву множину можливих фактів $\Omega = \{E_i, ..., E_n\}$, що взаємно виключають один одного. На множині всіх підмножин множині Ω задаємо функцію $m: 2^{\Omega} \to [0,1]$, що має такі властивості:

$$m(\varnothing) = 0,\ m(\Omega) = 1,\ \sum_{C \subset \Omega} m(C) = 1.$$

Величина $m(C)$ називається *ступенем (мірою) довіри (впевненості)* або *базовою ймовірністю множини C*.

Фіксованими базовими ймовірностями є $m(\{E_1\}), ..., m(\{E_n\})$, оскільки вони суттєво пов'язані тільки з одним фактом. *Вільними базовими ймовірностями* є $m(\{e_1, e_2\}), ..., m(\{e_1, ..., e_{n-1}\}), e_i \in \Omega$, які стосуються відразу двох і більше фактів. Сума всіх фіксованих і вільних базових ймовірностей повинна дорівнювати 1.

Задаємо функцію $Bl: 2^{\Omega} \to [0,1]$, що має такі властивості:

$$Bl(C) = \sum_{\hat{C} \subseteq C} m(\hat{C}),\ Bl(C) \in [0,1].$$

Величина $Bl(C)$ називається *ступенем (мірою) загальної довіри (впевненості)* або *нижньою ймовірністю множини C*.

Задамо функцію $Pl: 2^{\Omega} \to [0,1]$, що має такі властивості:

$$Pl(C) = 1 - Bl(\overline{C}) = \sum_{\hat{C} \subseteq \overline{C}} m(\hat{C}),\ Pl(C) \in [0,1].$$

Величина $Pl(C)$ називається *ступенем (мірою) правдоподібності (впевненості)* або *верхньою ймовірністю множини C.*

Міри $Bl(C)$ та $Pl(C)$ називають відповідно нижніми та верхніми можливостями, оскільки $Bl(C) \le Pl(C)$. Отже, неточність знань про множину C виражається інтервалом $[Bl(C), Pl(C)]$.

Для агрегування умов над фактами складниками передумови правила (антецедент) виконуються логічні операції. У цьому разі нижня ймовірність обчислюється так:

1. За умови логічного зв'язку І між фактами E_1 та E_2

$$Bl(E_1 \wedge E_2) = \min(Bl(E_1), Bl(E_2)).$$

2. За умови логічного зв'язку АБО між фактами E_1 та E_2

$$Bl(E_1 \vee E_2) = \max(Bl(E_1), Bl(E_2)).$$

Для активізації правил ТДШ базові ймовірності приписуються не тільки фактам, а й правилам. Саме так забезпечується облік ненадійності правил, які часто формуються на основі евристичних міркувань. Базова ймовірність заключення (консеквента) $m(B)$ визначається добутком нижньої ймовірності антецедента $Bl(A)$ та базової ймовірності правила $m(R)$

$$m(B) = Bl(A)m(R).$$

На практиці одне заключення (консеквент) може підтверджуватися не одним правилом, а кількома, тому необхідно провести агрегування правил з однаковим заключенням.

Наприклад, якщо одне заключення підтверджується двома правилами, то

$$m = m_1 m_2.$$

Недоліками ТДШ є те, що обчислювальна складність наведеного правила дорівнює 2^{Ω}.

Приклад

Нехай $\Omega = \{E_1, E_2, E_3\}$ – множина симптомів захворювання (наприклад, E_1 – висока температура, E_2 – почервоніння горла, E_3 – нежить).

Нехай базові ймовірності задані у вигляді $m(\{E_1\}) = 0.3, m(\{E_2\}) = 0.1, m(\{E_3\}) = 0.2$,
$m(\{E_1, E_2\}) = 0.2, m(\{E_2, E_3\}) = 0.2, m(\{E_1, E_3\}) = 0$.

Тоді відповідно до формули $Bl(C) = \sum_{\widehat{C} \subseteq C} m(\widehat{C})$ отримаємо нижні ймовірності:

$$Bl(\{E_1\}) = m(\{E_1\}) = 0.3,\ Bl(\{E_2\}) = m(\{E_2\}) = 0.1,\ Bl(\{E_3\}) = m(\{E_3\}) = 0.2,$$
$$Bl(\{E_1, E_2\}) = m(\{E_1\}) + m(\{E_2\}) + m(\{E_1, E_2\}) = 0.3 + 0.1 + 0.2 = 0.6,$$
$$Bl(\{E_2, E_3\}) = m(\{E_2\}) + m(\{E_3\}) + m(\{E_2, E_3\}) = 0.1 + 0.2 + 0.2 = 0.5,$$
$$Bl(\{E_1, E_3\}) = m(\{E_1\}) + m(\{E_3\}) + m(\{E_1, E_3\}) = 0.3 + 0.2 + 0 = 0.5.$$

Тоді відповідно до формули $Pl(C) = 1 - Bl(\overline{C})$ отримаємо верхні ймовірності:

$$Pl(\{E_1\}) = 1 - Bl(\{E_2, E_3\}) = 1 - 0.5 = 0.5,$$
$$Pl(\{E_2\}) = 1 - Bl(\{E_1, E_3\}) = 1 - 0.5 = 0.5,$$
$$Pl(\{E_3\}) = 1 - Bl(\{E_1, E_2\}) = 1 - 0.6 = 0.4,$$
$$Pl(\{E_1, E_2\}) = 1 - Bl(\{E_3\}) = 1 - 0.2 = 0.8,$$
$$Pl(\{E_2, E_3\}) = 1 - Bl(\{E_1\}) = 1 - 0.3 = 0.7,$$
$$Pl(\{E_1, E_3\}) = 1 - Bl(\{E_2\}) = 1 - 0.1 = 0.9.$$

Отже, базова ймовірність для симптому E_1 змінюється від 0.3 до 0.5, базова ймовірність для симптому E_2 змінюється від 0.1 до 0.5, базова ймовірність для симптому E_3 змінюється від 0.2 до 0.4.

Нехай антецедент представлений у вигляді $A = (E_1 \wedge E_2) \vee E_3$.

Тоді нижня ймовірність антецедента обчислюється у вигляді

$$Bl(A) = Bl((E_1 \wedge E_2) \vee E_3) = \max(\min(Bl(E_1), Bl(E_2)), Bl(E_3)) = 0.2.$$

Нехай задана базова ймовірність правила у вигляді $m(R) = 0.9$.

Тоді базова ймовірність консеквента обчислюються у вигляді

$$m(B) = Bl(A) \cdot m(R) = 0.2 * 0.9 = 0.18.$$

7.5. Механізм виведення INFERNO

Механізм неточного виведення INFERNO розроблявся з огляду на такі критерії:

а) система виведення не повинна залежати від будь-яких припущень про імовірнісний розподіл на множині висловлювань;

б) наявна про цей розподіл інформація (наприклад, дані про незалежність подій) повинна легко вводитися в систему;

в) висловлювання не повинні бути чітко розділені на факти і гіпотези – повинна бути можливість гнучкої зміни напрямку введення;

г) за наявності протиріч у вихідних даних система повинна їх виявляти і пропонувати шляхи усунення.

В INFERNO кожному висловлюванню A ставиться у відповідність імовірнісний інтервал $[t(A),1-f(A)]$, де $t(A)$ – оцінка знизу ймовірності $P(A)$, $f(A)$ – оцінка ймовірності $P(\overline{A})$. Отже, $t(A)\le P(A)\le 1-f(A)$. У процесі перерахунку оцінка $t(A)$ акумулює інформацію, що підтверджує A, а оцінка $f(A)$ акумулює інформацію, що спростовує A. Релевантні вислову A дані вважаються суперечливими, якщо $t(A)+f(A)>1$. Спочатку кожному висловлюванню A приписані значення $t(A)=f(A)=0$. Процесом перерахунку керують обмеження на $t(A)$ та $f(A)$ типу нерівності «≥», отримані хоч зі слабкої, але справедливої в усіх випадках нерівності $\max P(S_i)\le P(S_1\vee\ldots\vee S_n)\le\sum_{i=1}^{n}P(S_i)$. Наприклад, із твердженням «A тягне B з імовірністю S» ($P(B\mid A)=S$) пов'язані два обмеження:

$$t(B)/S\ge t(A),$$
$$f(A)\ge 1-(1-f(B))/S.$$

Перерахунок триває доти, поки всі обмеження не будуть задоволені. Якщо після закінчення перерахунку для будь-яких висловлювань A_i наявні нерівності $t(A_i)+f(A_i)>1$, тобто вихідні умови суперечливі, то INFERNO за допомогою спеціальних обмежень здійснює зворотний перерахунок і формує множини можливих виправлень, внесення яких у вихідні дані робить їх спільними. Вибір конкретної з цих множин здійснює користувач, звертаючись за необхідності до інтегрального показника «суттєвості» виправлень, що вносяться.

7.6. Обчислення інцидентів

Кожному висловлюванню A ставиться у відповідність підмножина $i(A)$ множини W. $i(A)$ називають інцидентом A та інтерпретують як відповідну висловлюванню A подію з імовірнісного простору елементарних подій W. Інцидент $i(A)$ приймається як міра неточності, що приписана A. У разі такого введення міри неточності логічні зв'язки виявляються функціонально-істинними щодо інцидентів, наприклад, $i(A \wedge B) = i(A) \cap i(B)$. Якщо для всіх висловлювань A відомі їх інциденти, то відомі не тільки ймовірності всіх A (елементарні висловлювання з W вважаються рівноймовірними), а й залежності між ними. Водночас інциденти та ймовірності складних висловлювань обчислюються за інцидентам висловлювань, їх складниками.

Процесом виведення в обчисленні інцидентів керує механізм виявлення допустимих присвоєнь (МВДП). На вхід МВДП надходить визначене користувачем початкове присвоєння F, тобто множина пар оцінок зверху та знизу $\{\sup_F(A),\ \inf_F(A)\}$ інцидентів висловлювань A. Користуючись спеціальними правилами виведення, МВДП будує множину $\{G_k\}$, що складається з усіх допустимих суджень G_k присвоєння F, $\sup_{G_k}(A_j) \subseteq \sup_F(A_j)$, $\inf_{G_k}(A_j) \subseteq \inf_F(A_j)$. Підмножини G_k визначені на всій множині цікавих для користувача висловлювань. Якщо допустимих суджень G_k присвоєння F не існує, тобто початкове присвоєння містить протиріччя, то результатом МВДП є встановлення факту наявності протиріччя. Основні труднощі застосування обчислення інцидентів зводяться до визначення початкового присвоєння.

8. ВИЗНАЧЕННЯ, ЦІЛІ, ЗАВДАННЯ, ПРИКЛАДИ ТА КЛАСИФІКАЦІЯ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

8.1. Визначення та цілі рекомендаційних систем

Одним із найбільш поширених видів інтелектуальних систем є рекомендаційні системи.

Рекомендаційна система (РС) – це інтелектуальна система, здатна на основі інформації про користувачів і предмети рекомендувати предмети, що підходять користувачам.

Основна мета рекомендаційної системи – збільшення доходів продавця через збільшення обсягу продажів рекомендованих предметів.

Для досягнення цієї мети рекомендаційні системи мають забезпечувати таке:

1. Актуальність. Рекомендовані предмети повинні бути актуальними (становити явний інтерес) для поточного користувача. Повинна бути присутня у всіх РС.

2. Новизна. Рекомендовані предмети повинні бути новинкою для користувача.

3. Виявлення прихованого інтересу користувача. Рекомендовані предмети повинні також дивувати користувача, а не просто бути актуальними або новинкою. Для виявлення прихованого інтересу використовується випадковий вибір предметів. Хоча деякі випадково вибрані предмети можуть бути нерелевантними, але в багатьох випадках довгострокові і стратегічні переваги випадкового вибору переважають ці короткострокові недоліки.

4. Збільшення різноманітності списку ТОП-k рекомендованих предметів.

До того ж рекомендаційна система може:

- допомогти підвищити загальну задоволеність користувача вебсайтом (це може підвищити лояльність користувачів і збільшити продажі на сайті);
- дати продавцю уявлення про потреби користувача і допомогти продавцю у подальшій взаємодії з користувачем;
- дати користувачу пояснення того, чому рекомендується той чи інший предмет.

8.2. Завдання рекомендаційних систем

Два основні завдання рекомендаційних систем:

1. *Задача прогнозування.* Полягає в обчисленні оцінки рейтингу для комбінації «користувач–предмет». Передбачається, що доступні навчальні дані, що вказують, які предмети надають перевагу користувачам. Для m користувачів і n предметів це відповідає неповній рейтинговій матриці розмірності $m \times n$, де зазначені значення (або спостережувані), використовуються для навчання, а відсутні значення (або ті, що не спостерігаються) обчислюються за допомогою навченої моделі. Ця задача також називається *задачею заповнення матриці*.

2. *Задача ранжування.* На практиці немає необхідності обчислювати оцінки рейтингів користувачів із конкретних предметів. Замість цього можна визначити ТОП-k предметів для конкретного користувача або визначити ТОП-k користувачів для конкретного предмета. Визначення ТОП-k предметів більш поширене, ніж визначення ТОП-k користувачів, хоча методи в цих двох випадках повністю аналогічні. Ця задача також *називається задачею ТОП-k рекомендацій*.

Для задачі ранжування абсолютні значення обчислюваних оцінок рейтингів неважливі. Задача прогнозування є більш загальною, адже вирішення завдання ранжування може бути отримано шляхом вирішення завдання прогнозування для різних комбінацій «користувач–предмет» і подальшого ранжування обчислених оцінок рейтингів. Однак у багатьох випадках простіше і природніше розробити методи для безпосереднього вирішення завдання ранжування.

8.3. Приклади рекомендаційних систем

Рекомендаційна система GroupLens

PC GroupLens була створена для рекомендації новин Usenet. PC GroupLens збирала оцінки читачів Usenet і використовувала їх, щоб визначити, чи сподобається стаття іншим читачам, перш ніж вони її прочитають. PC GroupLens використовує методи колаборативної (спільної) фільтрації. Загальні ідеї PC GroupLens були використані в рекомендаційних системах BookLens і MovieLens для рекомендації книг і фільмів.

Рекомендаційна система Amazon.com

Amazon.com спочатку була створена як книжковий інтернет-магазин, але згодом поширилася на практично всі види товарів (книги, компакт-диски, програмне забезпечення, електроніку тощо). PC Amazon.com надає рекомен-

дації на основі наявних рейтингів, які задані самими користувачами, а також buying-поведінки (дії, пов'язані з купівлею товару) і browsing-поведінки (дії, пов'язані з пошуком інформації в інтернеті) користувачів, які визначаються рекомендаційною системою і неявним рейтингом. Рейтинги на Amazon.com вказані за п'ятибальною шкалою, водночас найнижчий рейтинг – 1 зірка, а найвищий – 5 зірок. Дані про buying-поведінку і browsing-поведінку конкретних користувачів можуть бути зібрані, коли вони входять у систему за допомогою механізму автентифікації облікового запису, підтримуваного РС Amazon.com. Рекомендації надаються користувачам на головній вебсторінці сайта щоразу, коли вони входять до системи під своїми обліковими записами. У багатьох випадках даються пояснення до рекомендацій (наприклад, показується зв'язок рекомендованого товару з раніше купленими товарами).

Рекомендаційна система Netflix

Netflix спочатку була створена як компанія, що доставляє поштою DVD-диски з фільмами і телешоу, але потім була розширена до streaming-доставки (доставки потокового відео). РС Netflix надає користувачам можливість оцінювати фільми і телешоу за 5-бальною шкалою. До того ж дії користувача, пов'язані з переглядом різних предметів, також зберігаються в РС Netflix. Ці рейтинги і дії потім використовуються РС Netflix для складання рекомендацій. РС Netflix дає якісні пояснення до рекомендацій (наприклад, показується зв'язок рекомендованого товару з раніше купленими товарами, а також даються додаткові пояснення), що підвищує лояльність і утримання клієнтів. РС Netflix використовує методи колаборативної (спільної) фільтрації.

Рекомендаційна система Google News

РС Google News може рекомендувати новини користувачам на основі їх історії вибору новинних статей. Вибрані новинні статті пов'язуються з конкретними користувачами на основі механізмів ідентифікації, вміщених в облікові записи Gmail. У цьому разі новинні статті розглядаються як предмети. Те, що користувач вибирає новинну статтю, можна розглядати як позитивну оцінку цієї статті. Такі рейтинги можна розглядати як унарні рейтинги, в яких існує механізм, що дає змогу користувачам висловити свою симпатію до предмета, але не існує механізму, що дає змогу їм показати свою неприязнь. Ба більше, рейтинги є неявними, оскільки вони виводяться на основі дій користувача, а не вказуються ним чітко. Однак варіанти цього підходу можуть застосовуватися і у випадках, коли рейтинги чітко вказані. РС Google News використовує методи колаборативної (спільної) фільтрації.

Рекомендаційна система Facebook

PC Facebook рекомендує потенційних друзів користувачам, щоб збільшити кількість соціальних зв'язків на вебсайті «Facebook». Рекомендації друзів PC Facebook відрізняються від рекомендацій по продукту – рекомендації друзів PC Facebook засновані на структурних взаємозв'язках, а не на рейтингових даних продукту. До того ж рекомендація продукту безпосередньо збільшує прибуток продавця через збільшення продажів продукту, а збільшення кількості соціальних зв'язків користувачів сприяє зростанню соціальної мережі та збільшенню доходів від реклами.

Інші рекомендаційні системи

PC Jester для жартів, PC last.fm для музики, PC Pandora для музики, PC YouTube для онлайн-відео, PC IMDb для фільмів, PC Tripadvisor для продукції для мандрівників, PC Google Search для пошукової машини.

8.4. Класифікація рекомендаційних систем

Рекомендаційні системи класифікуються за такими параметрами:

1. Спеціалізація рекомендаційної системи, тобто тип рекомендованих предметів.
2. Завдання з погляду користувача (зазвичай знаходження відповідного предмета) і продавця (зазвичай збільшення обсягу продажів).
3. Обстановка, в якій користувач отримує рекомендацію (наприклад, швидкість прийняття рішення про вибір предмета, настрій користувача, час року і доби, поточне заняття та ін.).
4. Ступінь персоналізації рекомендації:
   - неперсональна рекомендація (користувачу подобаються такі предмети, які подобаються більшості);
   - напівперсональна рекомендація (користувачу подобаються такі предмети, які подобаються групі, якій притаманні ті самі ознаки, що і користувачу);
   - персональна рекомендація (базується на попередніх взаємодіях користувача з рекомендаційною системою).
5. Думка яких експертів враховується (якщо рекомендаційна система використовує експертну систему).
6. Конфіденційність даних користувача і надійність рекомендації.
7. Інтерфейс рекомендаційної системи:

- форма вводу інформації (вводить сам користувач або визначає систему);

- форма висновку інформації (рекомендація природно або неприродно вписана у вебсторінку, рекомендація з поясненням або без пояснення).

8. Методи формування рекомендацій:

- методи колаборативної фільтрації (використовують явні і неявні рейтинги, пов'язані з різними користувачами);

- методи контентної фільтрації (використовують явні і неявні рейтинги, пов'язані з конкретним користувачем, і опис предметів, що містить атрибути предметів);

- методи на основі знань (використовують базу знань та інтерактивні вимоги конкретного користувача, зазначені у вигляді обмежень на атрибути предметів або у вигляді предметів, що нагадують бажаний предмет);

- контекстно-залежні методи (зазвичай враховують залежність від часу, локації, соціального оточення);

- ансамблеві та гібридні методи.

8.5. Класифікація та розподіл рейтингів

Рейтинги можуть бути:

- безперервними (значення рейтингу знаходяться на безперервній шкалі, наприклад, [–10, 10]);

- інтервальними (значення рейтингів у вигляді дискретних значень; особливо поширено використання 5-бальних, 7-бальних та 10-бальних рейтингів);

- порядковими (значення рейтингів у вигляді категоріальних значень; наприклад, «абсолютно не згоден», «не згоден», «нейтрально», «згоден», «повністю згоден»);

- бінарними (частинний випадок інтервальних або порядкових рейтингів);

- унарними (в цьому разі користувач вказує тільки вподобаний предмет без явного виставлення рейтингу (це називається неявним зворотним зв'язком)).

Розподіл рейтингів предметів часто є довгохвостим розподілом (рис. 8.1), причому у хвості виявляються найменш популярні товари (їм рідко присвоюють рейтинг). Часто продавцю (наприклад, Amazon.com) вигідно продавати менш популярні товари для отримання більшого прибутку, оскільки вони менш конкурентноспроможні. Рекомендаційні системи часто рекомендують тільки популярні товари, що знижує різноманітність і робить менш цікавим такі системи. Рейтинги непопулярних товарів не є репрезентативними.

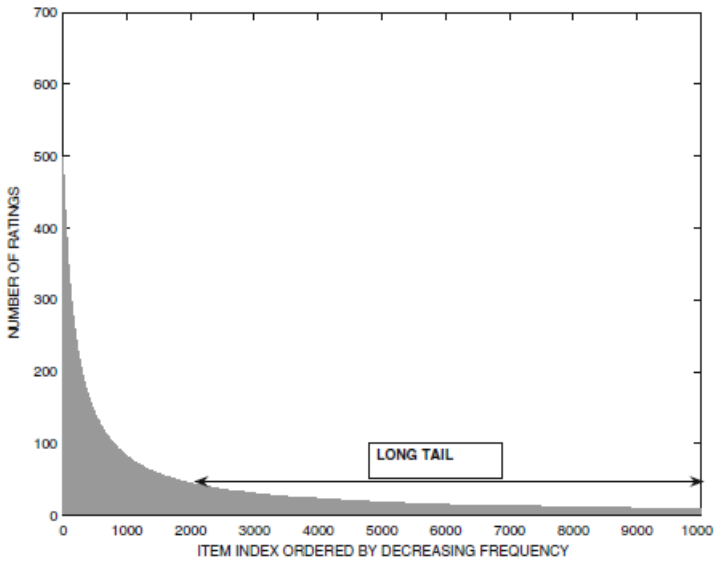


Рис. 8.1 – Приклад розподілу рейтингів предметів

9. ОСНОВНІ МЕТОДИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

9.1. Методи колаборативної фільтрації

У методах колаборативної (спільної) фільтрації для формування рекомендацій використовуються явні і неявні рейтинги. Основна проблема під час розробки методів колаборативної фільтрації полягає в тому, що рейтингові матриці розріджені. Зазначені рейтинги також називаються спостережуваними рейтингами. Незазначені рейтинги також називаються неспостережуваними, або відсутніми.

Існує два типи методів, які зазвичай використовуються в колаборативній фільтрації:

1. Методи на основі пам'яті або сусідства. У цих методах відсутні значення рейтингової матриці обчислюються на основі околиць. Ці околиці можна визначити одним із двох способів:

- колаборативна фільтрація на основі користувачів. Основна ідея полягає в тому, щоб визначити околицю (групу користувачів, які подібні до цільового користувача) і обчислити кожне відсутнє значення рейтингової матриці цільового користувача як зважене середнє рейтингів цієї околиці. Функції подібності обчислюються між рядками матриці рейтингів для виявлення схожих користувачів;
- колаборативна фільтрація на основі предметів. Основна ідея полягає в тому, щоб визначити околицю (групу предметів, які найбільш подібні до цільового предмета) і обчислити кожне відсутнє значення рейтингової матриці цільового предмета як зважене середнє рейтингів цієї околиці. Функції подібності обчислюються між стовпчиками матриці рейтингів для виявлення схожих предметів.

Колаборативна фільтрація на основі предметів більш точна, а також більш стабільна в разі зміни рейтингів. Колаборативна фільтрація на основі користувачів забезпечує більшу різноманітність.

2. Методи на основі моделей. У цих методах використовуються:

- моделі класифікаторів та апроксиматорів, які обчислюють відсутні значення рейтингової матриці;
- моделі латентних факторів, що представляють рейтингову матрицю у вигляді добутку двох редукованих щільних рейтингових матриць на основі методів матричного розкладання (матричної факторизації).

9.1.1. Методи на основі пам’яті або сусідства

Переваги методів на основі пам’яті або сусідства полягають у тому, що їх легко реалізувати, а отримані рекомендації часто легко пояснити, і вони відносно стабільні в разі додавання нових предметів і користувачів.

Недоліки методів на основі пам’яті або сусідства полягають у тому, що вони мають високу обчислювальну складність і не дуже добре працюють із розрідженими рейтинговими матрицями. Однак розрідженість не є проблемою, коли потрібні тільки ТОП-k предметів.

Наприклад (рис. 9.1), у Netflix рекомендації розділу More Like This формуються на основі сусідства.

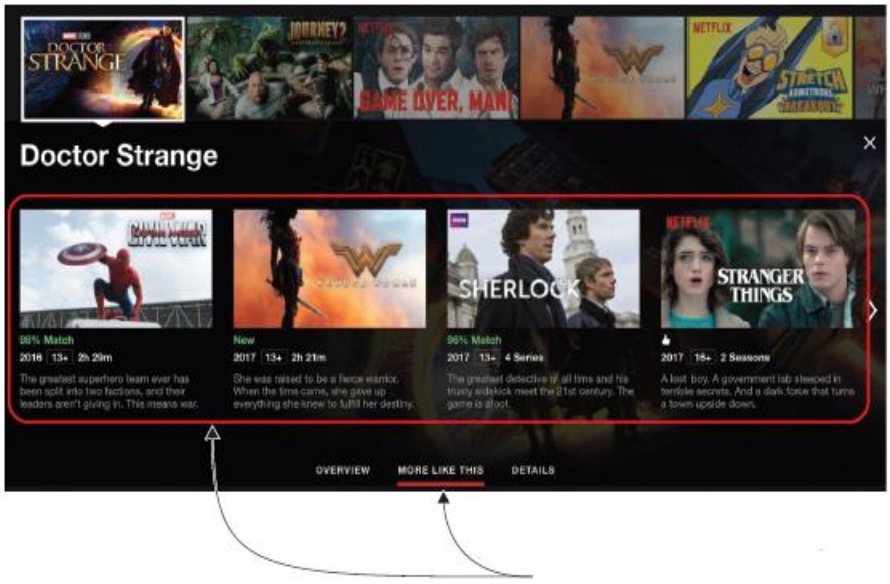


Рис. 9.1 – Приклад результату роботи рекомендаційної підсистеми Netflix

У разі колаборативної фільтрації на основі користувачів найчастіше оцінки рейтингів обчислюються на основі коефіцієнта кореляції Пірсона:

$$\hat{r}_{ai} = \frac{\sum\limits_{b \in P_a(i)} \text{Pearson}(a,b)(r_{bi} + \mu_a - \mu_b)}{\sum\limits_{b \in P_a(i)} \text{Pearson}(a,b)},$$

$$\text{Pearson}(a,b) = \frac{\sum\limits_{i \in I_a \cap I_b} (r_{ai} - \mu_a)(r_{bi} - \mu_b)}{\sqrt{\sum\limits_{i \in I_a \cap I_b} (r_{ai} - \mu_a)^2} \sqrt{\sum\limits_{i \in I_a \cap I_b} (r_{bi} - \mu_b)^2}},$$

$$\mu_a = \frac{\sum_{i \in I_a} r_{ai}}{|I_a|} \quad ,$$

де $\hat{r}_{ai}$ – оцінка рейтингу за моделлю (для користувача a за предметом i);

r_{ai}, r_{bi} – спостережуваний рейтинг;

$P_a(i)$ – множини k-найближчих користувачів для користувача a, який виставив рейтинг предмету i;

$\text{Pearson}(a,b)$ – кореляція користувачів a і b із множини спільно оцінених предметів $I_a \bigcap I_b$;

I_a – множина індексів предметів, яким виставив рейтинг користувач a;

μ_a – середній рейтинг користувача a за множиною оцінених предметів I_a.

У разі колаборативної фільтрації на основі предметів найчастіше оцінки рейтингів обчислюються на основі скоригованої косинусної подібності:

$$\hat{r}_{ud} = \frac{\sum_{c \in Q_d(u)} \text{AjustedCosine}(c,d) r_{uc}}{\sum_{c \in Q_d(u)} \text{AjustedCosine}(c,d)} \quad ,$$

$$\text{AjustedCosine}(c,d) = \frac{\sum_{u \in U_c \bigcap U_d} (r_{uc} - \mu_u)(r_{ud} - \mu_u)}{\sqrt{\sum_{u \in U_c \bigcap U_d} (r_{uc} - \mu_u)^2} \sqrt{\sum_{u \in U_c \bigcap U_d} (r_{ud} - \mu_u)^2}} \quad ,$$

$$\mu_u = \frac{\sum_{i \in I_u} r_{ui}}{|I_u|} \quad ,$$

де $Q_d(u)$ – множина k-найближчих предметів для предмета d, яким виставив рейтинг користувач u;

$\text{AjustedCosine}(c,d)$ – кореляція предметів c і d за множиною спільно оцінюючих їх користувачів $U_c \bigcap U_d$;

U_c – множина індексів користувачів, які виставили рейтинг предмету c.

Зауваження. Для прискорення обчислень рекомендується попередньо провести кластеризацію (наприклад, методом *k*-середніх), щоб розбити користувачів або предмети на кластери і обчислювати схожість тільки в тому кластері, до якого належить користувач або предмет.

У разі колаборативної фільтрації на основі користувачів оцінки рейтингів можуть також обчислюватися на основі регресії найближчих сусідів:

$$\hat{r}_{ai} = \sum_{b \in P_a(i)} w_{ab}(r_{bi} + \mu_a - \mu_b),\ \mu_a = \frac{\sum_{i \in I_a} r_{ai}}{|I_a|},$$

де w_{ab} – ваговий коефіцієнт (відповідає коефіцієнту регресії);

$\hat{r}_{ai}$ – оцінка рейтингу за моделлю (для користувача a за предметом i);

r_{bi} – спостережуваний рейтинг;

$P_a(i)$ – множина k-найближчих користувачів для користувача a, який виставив рейтинг предмету i;

I_a – множина індексів предметів, яким виставив рейтинг користувач a;

μ_a – середній рейтинг користувача a за множиною предметів I_a, які він оцінив.

У разі колаборативної фільтрації на основі предметів оцінки рейтингів можуть також обчислюватися на основі регресії найближчих сусідів:

$$\hat{r}_{ud} = \sum_{c \in Q_d(u)} w_{cd} r_{uc},$$

де $Q_d(u)$ – множина k-найближчих предметів для предмета d, яким виставив рейтинг користувач u.

Функція втрат у разі апроксимації і без урахування представлена у вигляді:

$$F(\mathbf{w}) = R(\mathbf{w}) + RSS(\mathbf{w}) \text{ з обмеженнями } w_{ab}, w_{cd} \geq 0,\ w_{aa}, w_{cc} = 0,$$

де $RSS(\mathbf{w})$ – втрати на навчальних даних (сума квадратів залишків);

$R(\mathbf{w})$ – втрати регуляризації.

Втрати на навчальних даних у разі:

а) колаборативної фільтрації на основі користувачів

$$RSS(\mathbf{w}) = \sum_{a=1}^{m} \sum_{i \in I_a} (r_{ai} - \hat{r}_{ai})^2 ,$$

де $\hat{r}_{ai}$ – рейтинг за моделлю;

r_{ai} – навчальний рейтинг;

б) колаборативної фільтрації на основі предметів

$$RSS(\mathbf{w}) = \sum_{d=1}^{n} \sum_{u \in U_d} (r_{ud} - \hat{r}_{ud})^2 .$$

Втрати регуляризації у разі:

- відсутності регуляризації (звичайна регресія)

$$R(\mathbf{w}) = 0 ,$$

- регуляризації l_1 (регресія LASSO (Least Absolute Shrinkage and Selection Operator))

$$R(\mathbf{w}) = \lambda \|\mathbf{w}\|_{l_1} ;$$

- регуляризації l_2 (регресія Ridge)

$$R(\mathbf{w}) = \lambda \frac{1}{2} \|\mathbf{w}\|_{l_2}^2 ;$$

- комбінації регуляризації l_1 і l_2 (еластична мережа)

$$R(\mathbf{w}) = \lambda \left(\frac{1-\rho}{2} \|\mathbf{w}\|_{l_2}^2 + \rho \|\mathbf{w}\|_{l_1} \right),$$

де λ – параметр регуляризації, $\lambda > 0$;

ρ – параметр змішування, $0 < \rho < 1$.

Завдання ідентифікації вектора параметрів представлено у вигляді:

$$F(\mathbf{w}) \to \min_{\mathbf{w}} .$$

Зауваження. Вектор параметрів $\mathbf{w}$ може бути обчислений на основі методу найменших квадратів (у разі відсутності регуляризації або регуляризації l_2); методу регресії найменших кутів і стиснення (у разі регуляризації l_1); методу покоординатного спуску (у разі відсутності регуляризації або всіх видів регуляризації).

У разі колаборативної фільтрації на основі користувачів і предметів оцінки рейтингів можуть також обчислюватися на основі регресії найближчих сусідів:

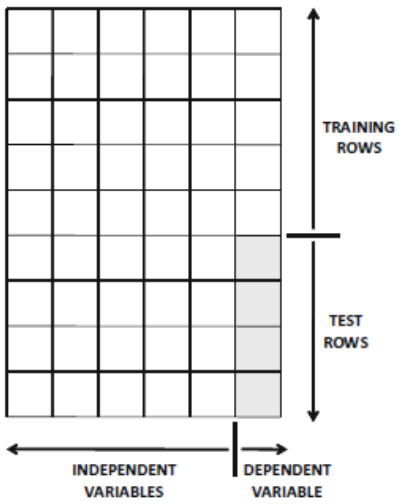
$$\hat{r}_{ui} = \sum_{b \in P_u(i)} w_{ub}^{user} (r_{bi} + \mu_u - \mu_b) + \sum_{c \in Q_i(u)} w_{ci}^{item} r_{uc} .$$

9.1.2. Методи на основі моделей класифікаторів або апроксиматорів

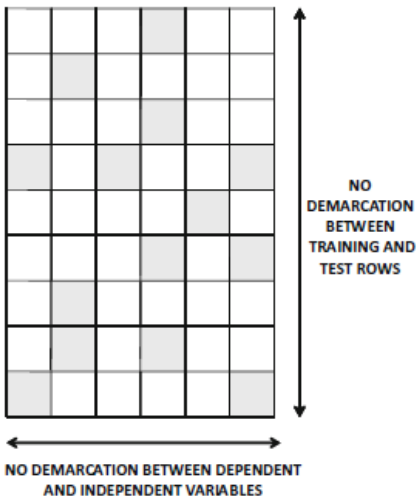
У методах на основі моделей етап навчання (або етап побудови моделі) відокремлений від етапу обчислення оцінки рейтингів. Прикладами таких методів у машинному навчанні є дерева рішень, методи знаходження взаємозв'язків, байєсівські класифікатори, регресійні моделі, штучні нейронні мережі. Завдання класифікації та регресії є приватними випадками завдання заповнення рейтингової матриці.

У завданні класифікації даних є матриця розміру $m \times n$, у якій перші $(n-1)$ стовпчики є ознаками (незалежними змінними), а останній (n) стовпчик – це номер класу (залежна змінна). Усі дані в перших $(n-1)$ стовпчиках вказані повністю, водночас в останньому (n) стовпчику вказано лише частину даних. Отже, підмножину рядків у матриці повністю визначено, і ці рядки називаються навчальними даними. Інші рядки називаються тестовими даними. Це показано на рис. 9.2(а), де заштриховані значення представляють відсутні елементи в матриці.

У методах на основі моделей будь-які дані в рейтинговій матриці можуть бути відсутні, як показано заштрихованими значеннями на рис. 9.2(б). Отже, завдання заповнення рейтингової матриці є узагальненням завдання класифікації або регресії.



Класифікація (а)



Заповнення рейтингової матриці (б)

Рис. 9.2 – Приклад задач класифікації та заповнення рейтингової матриці

Отже, принципові відмінності між цими двома задачами полягають у такому:

1. У задачі класифікації даних існує чіткий поділ між ознаками (незалежними змінними) і номерами класів (залежними змінними). У задачі заповнення рейтингової матриці такого чіткого поділу не існує, і кожен стовпчик може бути і залежною, і незалежною змінною.

2. У задачі класифікації даних існує чіткий поділ між навчальними та тестовими даними. У задачі заповнення рейтингової матриці цієї чіткої межі між рядками матриці не існує. У кращому разі можна вважати зазначені (спостережувані) дані навчальними даними, а невказані (відсутні) дані – тестовими даними.

3. У задачі класифікації даних стовпчики представляють ознаки, а рядки представляють зразки. Однак для вирішення задачі заповнення рейтингової матриці можна застосувати один і той самий підхід і до рейтингової матриці, і до транспонованої рейтингової матриці.

Не завжди легко використовувати моделі класифікації та апроксимації даних для вирішення задачі заповнення рейтингової матриці, особливо коли переважна більшість елементів відсутня.

Методи на основі моделей мають такі переваги, порівняно з методами на основі пам’яті або сусідства:

1. Розмір моделі набагато менший, ніж вихідна рейтингова матриця.

2. Швидкість навчання моделі і швидкість обчислення оцінок рейтингів за моделлю вища, ніж етапи попередньої обробки й обчислення оцінок рейтингів методів на основі пам’яті або сусідства.

3. Можна уникнути перенавчання шляхом регуляризації.

Незважаючи на те, що методи на основі пам'яті сусідства були одними з перших методів колаборативної фільтрації, а також одними з найпопулярніших через свою простоту, вони не обов'язково є найточнішими. Наразі деякі з найбільш точних методів є методами на основі моделей.

Методи дерев рішень для класифікації. Для обчислення оцінки інтервального k-бального рейтингу кожного предмета j будується своє j-е дерево рішень.

На самому початку з рейтингової матриці $R=[r_{ui}]$ розмірності $m\times n$ виключається j-й стовпчик, а відсутні значення заповнюються (наприклад, нулем). Потім до отриманої матриці застосовується матричне розкладання, внаслідок якого вона перетворюється на добуток двох редукованих щільних матриць $\mathbf{P}$ і $\mathbf{V}$ розмірності $m\times d$ і $n\times d$ відповідно, де $d<<n-1$. Обчислюється d-мірний вектор оцінок рейтингів для кожного користувача u у вигляді:

$$\mathbf{a}_u=(a_{u1},...,a_{ud}),\ a_{ui}=\frac{\sum_{k\in I_u} r_{uk}v_{ki}}{|I_u|},$$

де I_u – множина індексів предметів, яким виставив рейтинг користувач u.

Ці вектори використовуються для побудови дерева рішень для предмета j, причому в цьому дереві термінальні вершини будуть містити бали.

Методи знаходження взаємозв'язків. Ці методи зазвичай використовуються для унарних рейтингів. Найчастіше використовується метод Apriori.

Підтримка (support) використовується для вимірювання популярності предметів. Підтримка $Support(c)$ розраховується як відношення кількості користувачів, які обрали предмет c, до загальної кількості користувачів, тобто:

$$Support(c)=\frac{\sum_{a\in U_c}[r_{ac}=like]}{m},$$

де U_c – множина індексів користувачів, які виставили рейтинг предмету c.

Підтримка $Support(c\cap d)$ розраховується як відношення кількості користувачів, які обрали і предмет c, і предмет d, до загальної кількості користувачів, тобто:

$$Support(c \bigcap d) = \frac{\sum_{a \in U_c \bigcap U_d} [r_{ac} = like][r_{ad} = like]}{m}.$$

Ймовірність (probability) або достовірність (confidence) правила $c \Rightarrow d$ розраховується у вигляді:

$$Probability(d \mid c) = Support(c \bigcap d) / Support(c)$$

або

$$Probability(d \mid c) = \frac{\sum_{a \in U_c \bigcap U_d} [r_{ac} = like][r_{ad} = like]}{\sum_{a \in U_c} [r_{ac} = like]}.$$

Спрощений (наївний) метод Байєса. Для обчислення оцінки інтервального k-бального рейтингу кожного предмета j будується свій j-й байєсівський класифікатор.

На самому початку з рейтингової матриці $R = [r_{ui}]$ розмірності $m \times n$, виключається j-й стовпчик. На входи байєсівського класифікатора подаються рейтинги $n-1$ предмета. Байєсівський класифікатор буде обчислювати оцінку рейтингу предмета, що залишився. Частина рядків рейтингової матриці використовується як навчальні, а інша частина рядків використовується як тестові. У разі інтервальних k-бальних рейтингів у байєсівського класифікатора буде $n-1$ входів і k виходів.

Апостеріорна ймовірність обчислюється за теоремою Байєса у вигляді:

$$P(y = k \mid \mathbf{x} = (l_1,...,l_{n-1})) = \frac{P(\mathbf{x} = (l_1,...,l_{n-1}) \mid y = k) P(y = k)}{P(\mathbf{x} = (l_1,...,l_{n-1}))},$$

$$P(\mathbf{x} = (l_1,...,l_{n-1}) \mid y = k) = \prod_{z=1}^{n-1} P(x_z = l_z \mid y = k),$$

$$P(y = k) = \frac{\sum_{a \in U_j} [r_{aj} = k]}{|U_j|}, \; [r_{aj} = k] = \begin{cases} 1, & r_{aj} = k \\ 0, & r_{aj} \neq k \end{cases},$$

$$P(x_z = l_z \mid y = k) = \frac{\sum\limits_{a \in U_z \cap U_j} [r_{az} = l_z][r_{aj} = k]}{\sum\limits_{a \in U_j} [r_{aj} = k]},$$

де $P(\mathbf{x} = (l_1,...,l_{n-1}) \mid y = k)$, $P(y = k)$ – апріорні ймовірності

U_j – множина індексів користувачів, які виставили рейтинг предмету j;

$\mathbf{x}$ – вектор значень ознак (вектор рейтингів);

y – значення відгуку (рейтинг), $y \in \{1,...,K\}$.

Для спрощення ймовірність $P(\mathbf{x})$ зазвичай опускають як константу.

На основі вирішального правила максимуму апостеріорної ймовірності оцінка рейтингу визначається у вигляді:

$$k^* = \arg\max_k P(y = k \mid \mathbf{x} = (l_1,...,l_{n-1})),\ k \in \overline{1,K}.$$

Наприклад, нехай дано 5-бальну рейтингову матрицю у вигляді таблиці і треба знайти оцінку рейтингу у п’ятого предмета, причому перші чотири рядки є навчальними, а п’ятий – тестовим, тобто $\mathbf{x} = (1,3,3,2)$.

Таблиця 9.1

Рейтингова матриця

	Item1	**Item2**	**Item3**	**Item4**	**Item5**
User1	2	4	2	2	4
User2	1	3	3	5	1
User3	4	5	2	3	3
User4	1	1	5	2	1
User5	1	3	3	2	?

Тоді

$P(y = 1) = 2/4$, $P(y = 2) = 0$, $P(y = 3) = 1/4$, $P(y = 4) = 1/4$,

$P(y = 4) = 1/4$, $P(y = 5) = 0$.

$$P(\mathbf{x}=(1,3,3,2)\,|\,y=1)=P(x_1=1\,|\,y=1)P(x_2=3\,|\,y=1)P(x_3=3\,|\,y=1)\cdot$$
$$\cdot P(x_4=2\,|\,y=1)=2/2*1/2*1/2*1/2=0.125,$$
$$P(\mathbf{x}=(1,3,3,2)\,|\,y=2)=0,$$

$$P(\mathbf{x}=(1,3,3,2)\,|\,y=3)=0,\ P(\mathbf{x}=(1,3,3,2)\,|\,y=4)=0.$$

$$P(y=1\,|\,\mathbf{x}=(1,3,3,2))=P(y=1)P(\mathbf{x}=(1,3,3,2)\,|\,y=1)=0.0625,$$
$$P(y=2\,|\,\mathbf{x}=(1,3,3,2))=P(y=2)P(\mathbf{x}=(1,3,3,2)\,|\,y=2)=0,$$
$$P(y=3\,|\,\mathbf{x}=(1,3,3,2))=P(y=3)P(\mathbf{x}=(1,3,3,2)\,|\,y=3)=0,$$
$$P(y=4\,|\,\mathbf{x}=(1,3,3,2))=P(y=4)P(\mathbf{x}=(1,3,3,2)\,|\,y=4)=0,$$
$$P(y=5\,|\,\mathbf{x}=(1,3,3,2))=P(y=5)P(\mathbf{x}=(1,3,3,2)\,|\,y=5)=0.$$

$$k^*=\arg\max_{k\in 1,5}P(y=k\,|\,\mathbf{x}=(1,3,3,2))=1.$$

Отже, оцінка рейтингу дорівнює 1.

Метод мультиноміальної (поліноміальної, softmax) логістичної регресії (або класифікатора максимальної ентропії). Для обчислення оцінки інтервального k-бального рейтингу кожного предмета j будується свій j-й байєсівський класифікатор.

На самому початку з рейтингової матриці $R=[r_{ui}]$ розмірності $m\times n$ виключається j-й стовпчик, а відсутні значення заповнюються (наприклад, нулем). На входи байєсівського класифікатора подаються рейтинги $n-1$ предмета. Байєсівський класифікатор буде обчислювати оцінку рейтингу предмета, що залишився. Частина рядків рейтингової матриці використовується як навчальні, а інша частина рядків використовується як тестові. У разі інтервальних k-бальних рейтингів у байєсівського класифікатора буде $n-1$ входів і k виходів.

Апостеріорна ймовірність обчислюється softmax-функцією у вигляді:

$$P(y=k\,|\,\mathbf{x}=(l_1,\ldots,l_{n-1}))=\mathrm{softmax}(\mathbf{x})=\frac{\exp\left(-\left(\sum_{j=1}^{n-1}w_{kj}x_j+w_{k0}\right)\right)}{\sum_{s=1}^{K}\exp\left(-\left(\sum_{j=1}^{n-1}w_{sj}x_j+w_{s0}\right)\right)},$$

де w_j – вага для j-ої ознаки (предмета);

w_0 – зміщення.

Функція втрат у разі мультикласової класифікації:

$$F(\mathbf{w}) = R(\mathbf{w}) + L(\mathbf{w}),$$

$$L(\mathbf{w}) = -\ln \prod_{i=1}^{I} \prod_{k=1}^{K} P(y = k \mid \mathbf{x}_i = (l_{i1},..., l_{i,n-1}))^{[d_i = k]} =$$

$$= -\sum_{i=1}^{I} \sum_{k=1}^{K} [d_i = k] \ln P(y = k \mid \mathbf{x}_i = (l_{i1},..., l_{i,n-1})),$$

де $L(\mathbf{w})$ – втрати на навчальних даних (негативний логарифм правдоподібності або категоріальна крос-ентропійна функція);

$R(\mathbf{w})$ – втрати регуляризації;

$\mathbf{x}_i$ – i-й вектор значень ознак (вектор рейтингів);

d_i – i-е значення навчального відгуку (рейтинг), $d_i \in \{1,..., K\}$.

Втрати регуляризації у разі:

- відсутності регуляризації (звичайна мультиноміальна логістична регресія)

$$R(\mathbf{w}) = 0;$$

- регуляризації l_1 (регресія LASSO)

$$R(\mathbf{w}) = \lambda \|\mathbf{w}\|_{l_1};$$

- регуляризації l_2 (регресія Ridge)

$$R(\mathbf{w}) = \lambda \frac{1}{2} \|\mathbf{w}\|_{l_2}^2;$$

- комбінації регуляризації l_1 і l_2 (еластична мережа)

$$R(\mathbf{w}) = \lambda \left(\frac{1-\rho}{2} \|\mathbf{w}\|_{l_2}^2 + \rho \|\mathbf{w}\|_{l_1} \right),$$

де λ – параметр регуляризації, $\lambda > 0$;

ρ – параметр змішування, $0 < \rho < 1$.

Завдання ідентифікації вектора параметрів представлено у вигляді:

$$F(\mathbf{w}) \to \min_{\mathbf{w}}.$$

Метод градієнтного спуску з дельта-правилом у пакетному режимі без регуляризації (звичайна мультиноміальна логістична регресія)

$$w_{kj} = w_{kj} + \eta\left(\sum_{i=1}^{I}([d_i = k] - P(y_i = k \mid \mathbf{x}_i = (l_{i1},...,l_{i,n-1})))x_{ij}\right),\ k \in \overline{1,K},$$

$j \in \overline{1,n-1},$

$$w_{k0} = w_{k0} + \eta\left(\sum_{i=1}^{I}([d_i = k] - P(y_i = k \mid \mathbf{x}_i = (l_{i1},...,l_{i,n-1})))\right),$$

$k \in \overline{1,K}$,

де η – параметр, що визначає швидкість навчання (при великому η навчання відбувається швидше, але збільшується небезпека отримати неправильне рішення), $0 < \eta < 1$.

Метод градієнтного спуску з дельта-правилом у пакетному режимі з регуляризацією l_1 (регресією LASSO)

$$w_{kj} = w_{kj} + \eta\left(-\lambda\,\mathrm{sgn}(w_{kj}) + \sum_{i=1}^{I}([d_i = k] - P(y_i = k \mid \mathbf{x}_i = (l_{i1},...,l_{i,n-1})))x_{ij}\right),$$

$k \in \overline{1,K}$, $j \in \overline{1,n-1},$

$$w_{k0} = w_{k0} + \eta\left(-\lambda\,\mathrm{sgn}(w_{k0}) + \sum_{i=1}^{I}([d_i = k] - P(y_i = k \mid \mathbf{x}_i == (l_{i1},...,l_{i,n-1})))\right),$$

$k \in \overline{1,K}$.

Метод градієнтного спуску з дельта-правилом у пакетному режимі з регуляризацією l_2 (регресією Ridge)

$$w_{kj} = w_{kj} + \eta\left(-\lambda w_{kj} + \sum_{i=1}^{I}([d_i = k] - P(y_i = k \mid \mathbf{x}_i = (l_{i1},...,l_{i,n-1})))x_{ij}\right),$$

$k \in \overline{1,K}$, $j \in \overline{1,n-1},$

$$w_{k0} = w_{k0} + \eta\left(-\lambda w_{k0} + \sum_{i=1}^{I}([d_i = k] - P(y_i = k \mid \mathbf{x}_i == (l_{i1},...,l_{i,n-1}))) \right),$$

$k \in \overline{1, K}$.

Метод градієнтного спуску з дельта-правилом у пакетному режимі з еластичною мережею

$$w_{kj} = w_{kj} + \eta\Big(-\lambda\big((1-\rho)w_j + \rho\,\mathrm{sgn}(w_{kj})\big) +$$

$$+ \sum_{i=1}^{I}([d_i = k] - P(y_i = k \mid \mathbf{x}_i = (l_{i1},...,l_{i,n-1})))x_{ij} \Big),\ k \in \overline{1, K},\ j \in \overline{1, n-1},$$

$$w_{k0} = w_{k0} + \eta\Big(-\lambda\big((1-\rho)w_{k0} + \rho\,\mathrm{sgn}(w_{k0})\big) +$$

$$+ \sum_{i=1}^{I}([d_i = k] - P(y_i = k \mid \mathbf{x}_i = (l_{i1},...,l_{i,n-1}))) \Big),\ k \in \overline{1, K}.$$

На основі вирішального правила максимуму апостеріорної ймовірності оцінка рейтингу визначається у вигляді:

$$k^* = \max_k P(y = k \mid \mathbf{x} = (l_1,...,l_{n-1})),\ k \in \overline{1, K}.$$

Лінійна регресія. Для обчислення оцінки k-бального рейтингу кожного предмета j будується своя j-а регресійна модель.

На самому початку з рейтингової матриці $R = [r_{ui}]$ розмірності $m \times n$ виключається j-й стовпчик, а відсутні значення заповнюються (наприклад, нулем). На входи регресійної моделі подаються рейтинги $n-1$ предмета. Регресійна модель буде обчислювати оцінку рейтингу решти предмета. Частина рядків рейтингової матриці використовується як навчальні, а інша частина рядків використовується як тестові. У регресійної моделі буде $n-1$ входів і один вихід.

Лінійний регресійний поліном для апроксимації представлений у вигляді:

$$y = f(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{n-1} w_j x_j + w_0,$$

де w_j – вага для j-ї ознаки (предмета);

w_0 – зміщення.

Функція втрат у разі апроксимації представлена у вигляді:

$$F(\mathbf{w}) = R(\mathbf{w}) + RSS(\mathbf{w}),$$

$$RSS(\mathbf{w}) = \sum_{i=1}^{I} (d_i - f(\mathbf{x}_i, \mathbf{w}))^2,$$

де $RSS(\mathbf{w})$ – втрати на навчальних даних (сума квадратів залишків);

$R(\mathbf{w})$ - втрати регуляризації;

$\mathbf{x}_i$ – i-й вектор значень ознак (вектор рейтингів);

d_i – i -е значення навчального відгуку (рейтинг).

Втрати регуляризації у разі:

- відсутності регуляризації (звичайна лінійна регресія)

$$R(\mathbf{w}) = 0,$$

- регуляризації l_1 (регресія LASSO)

$$R(\mathbf{w}) = \lambda \|\mathbf{w}\|_{l_1},$$

- регуляризації l_2 (регресія Ridge)

$$R(\mathbf{w}) = \lambda \frac{1}{2} \|\mathbf{w}\|_{l_2}^2,$$

- комбінації регуляризації l_1 і l_2 (еластична мережа)

$$R(\mathbf{w}) = \lambda \left(\frac{1-\rho}{2} \|\mathbf{w}\|_{l_2}^2 + \rho \|\mathbf{w}\|_{l_1} \right),$$

де λ – параметр регуляризації, $\lambda > 0$;

ρ – параметр змішування, $0 < \rho < 1$.

Задачу ідентифікації вектора параметрів представлено у вигляді:

$$F(\mathbf{w}) \to \min_{\mathbf{w}}.$$

Зауваження. Вектор параметрів $\mathbf{w}$ може бути обчислений на основі методу найменших квадратів (у разі відсутності регуляризації або регуляризації l_2); методу регресії найменших кутів і стиснення (у разі регуляризації l_1);

методу покоординатного спуску (у разі відсутності регуляризації або всіх видів регуляризації).

Штучна нейронна мережа (ШНМ). Для обчислення оцінки k-бального рейтингу кожного предмета j будується своя j-а ШНМ.

На самому початку з рейтингової матриці $R=[r_{ui}]$ розмірності $m\times n$ виключається j-й стовпчик, а відсутні значення заповнюються (наприклад, нулем). На входи ШНМ подаються рейтинги $n-1$ предмета. ШНМ буде обчислювати оцінку рейтингу предмета, що залишився. Частина рядків рейтингової матриці використовується як навчальні, а інша частина рядків використовується як тестові. У разі інтервальних k-бальних рейтингів у ШНМ буде $n-1$ входів, один вихід для апроксимації і k виходів для мультикласової класифікації (кожному виходу відповідає значення рейтингу, що розглядається як мітка класу).

На рис. 9.3–9.4 наведено багатошаровий персептрон (MLP) для апроксимації та мультикласової класифікації, який є нерекурентною статичною багатошаровою ШНМ, що містить один або більше прихованих шарів і вихідний шар. Класи поділяються гіперплощинами.

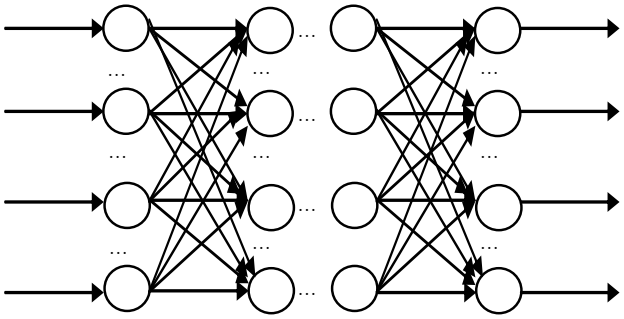


Рис. 9.3 – Багатошаровий персептрон (MLP) для мультикласової класифікації

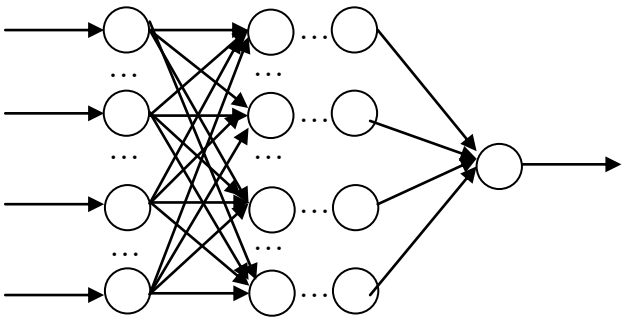


Рис. 9.4 – Багатошаровий персептрон (MLP) для апроксимації

Для MLP використовується навчання на основі корекції помилок (навчання з учителем), водночас найчастіше застосовується метод градієнтного спуску, на якому заснований метод зворотного поширення (ВР) для ШНМ прямого поширення.

MLP для мультикласової класифікації або апроксимації визначено у вигляді

$$\mathbf{y}^{(0)} = \mathbf{x},\ y_k^{(l)} = f^{(l)}\left(w_{0k}^{(l)} + \sum_{j=1}^{S^{(l-1)}} w_{kj}^{(l)} y_j^{(l-1)} \right),\ k \in \overline{1, S^{(l)}}, l \in \overline{1, L}$$

або у вигляді

$$\mathbf{y} = f(\mathbf{x}, \mathbf{w}^{(1)}, ..., \mathbf{w}^{(L)}) = f^{(1)} \circ ... \circ f^{(L)}(\mathbf{x}, \mathbf{w}^{(1)}, ..., \mathbf{w}^{(L)}),$$

де $w_{kj}^{(l)}$ – вага зв'язку від j-го нейрона $l-1$-го шару до k-го нейрона l-го шару;

$w_{0k}^{(l)}$ – зміщення k-го нейрона на l-му шарі;

$S^{(l)}$ – кількість нейронів у l-му шарі;

$y_k^{(l)}$ – вихід k-го нейрона на l-му шарі;

$f^{(l)}$ – функція активація нейронів l-го шару;

L – кількість шарів.

Середньоквадратична помилка, отримана за i-м зразком у разі мультикласової класифікації, визначена у вигляді:

$$e_i = \frac{1}{2} \left\| \mathbf{d}_i - f(\mathbf{x}_i, \mathbf{w}^{(1)}, ..., \mathbf{w}^{(L)}) \right\|_{l_2}^2,$$

де $\mathbf{x}_i$ – i-й вектор значень ознак (вектор рейтингів),

d_{ik} – i-е значення навчального відгуку (рейтинг у вигляді бінарного вектора, яка містить єдину одиницю в позиції, що відповідає значенню рейтингу), $d_{ik} \in \{0,1\}$.

Середньоквадратична помилка, отримана за i-м зразком у разі апроксимації, визначена у вигляді:

$$e_i = \frac{1}{2}(d_i - f(\mathbf{x}_i, \mathbf{w}^{(1)},..., \mathbf{w}^{(L)}))^2,$$

де $\mathbf{x}_i$ –i-й вектор значень ознак (вектор рейтингів);

d_i – i-е значення навчального відгуку (рейтинг).

Функція втрат у разі мультикласової класифікації або апроксимації представлена у вигляді:

$$F(\mathbf{w}) = E(\mathbf{w}),$$

$$E(\mathbf{w}) = \sum_{i=1}^{I} e_i,$$

де $E(\mathbf{w})$ – втрати на навчальних даних (сума середньоквадратичних помилок по всій множини зразків).

Задачу ідентифікації матриці параметрів представлено у вигляді:

$$F(\mathbf{w}) \to \min_{\mathbf{w}}.$$

Навчання ШНМ (метод зворотного поширення) в послідовному режимі

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [–0.5, 0.5] зміщень (порогів) $w_{0k}^{(l)}(n)$ і вагів $w_{kj}^{(l)}(n)$, $j \in \overline{1, S^{(l-1)}}, k \in \overline{1, S^{(l)}}, l \in \overline{1, L}$, де $S^{(l)}$ – кількість нейронів у l-му шарі, L – кількість шарів.

2. Задається навчальна множина

$$\{(\mathbf{x}_i, \mathbf{d}_i)\},\ i \in \overline{1, I},$$

де $\mathbf{x}_i$ – i-й навчальний вхідний вектор (вектор рейтингів) довжини $S^{(0)}$;

$\mathbf{d}_i$ – i-й навчальний вихідний вектор довжини $S^{(L)}$;

$S^{(0)}$ – кількість нейронів вхідного шару;

$S^{(L)}$ – кількість нейронів вихідного шару;

I – потужність навчальної множини.

Номер поточної пари з навчальної множини $i = 1$.

3. Обчислення вихідного сигналу для кожного шару (прямий хід):

$$\mathbf{y}^{(0)}(n) = \mathbf{x}_i,$$

$$y_k^{(l)}(n) = f^{(l)}(s_k^{(l)}(n)), \qquad s_k^{(l)}(n) = \sum_{j=0}^{S^{(l-1)}} w_{kj}^{(l)}(n) y_j^{(l-1)}(n),$$

$$k \in \overline{1, S^{(l)}}, l \in \overline{1, L},$$

де $S^{(l)}$ – число нейронів у l-му шарі;

l – номер шару;

L – число шарів;

$w_{kj}^{(l)}(n)$ – вага зв'язку від j-го нейрона $l-1$-го шару до k-го нейрона l-го шару в момент часу n;

$y_k^{(l)}(n)$ – вихід k-го нейрона на l-му шарі;

$f^{(l)}$ – функція активації нейронів l-го шару.

Вважається, що $y_0^{(l-1)}(n) = 1$.

4. Обчислення енергії помилки ШНМ:

$$E(n) = \frac{1}{2} \sum_{k=1}^{S^{(L)}} e_k^2(n),\ e_k(n) = y_k^{(L)}(n) - d_{ik}.$$

5. Налаштування синаптичних вагів на основі узагальненого дельта-правила (зворотний хід):

$$w_{kj}^{(l)}(n+1) = w_{kj}^{(l)}(n) - \eta \frac{\partial E(n)}{\partial w_{kj}^{(l)}(n)},$$

де η – параметр, що визначає швидкість навчання (при великому η навчання відбувається швидше, але збільшується небезпека отримати неправильне рішення), $0 < \eta < 1$.

$$\frac{\partial E(n)}{\partial w_{kj}^{(l)}(n)} = y_j^{(l-1)}(n) g_k^{(l)}(n),\ j \in \overline{0, S^{(l-1)}},\ k \in \overline{1, S^{(l)}},\ l \in \overline{1, L},$$

$$g_k^{(l)}(n) = \begin{cases} f'^{(L)}(s_k^{(L)}(n))(y_k^{(L)}(n) - d_{ik}), & l = L \\ f'^{(l)}(s_k^{(l)}(n)) \sum_{z=1}^{S^{(l+1)}} w_{zk}^{(l+1)}(n) g_z^{(l+1)}(n), & l < L \end{cases}.$$

6. Перевірка умов завершення:

Якщо $i \bmod I > 0$, то $i = i + 1,\ n = n + 1$, перехід до 3.

Якщо $i \bmod I = 0$ і $\frac{1}{I}\sum_{s=1}^{I} E(n - I + s) > \varepsilon$, то $n = n + 1$, перехід до 2.

Якщо $i \bmod I = 0$ і $\frac{1}{I}\sum_{s=1}^{I} E(n - I + s) < \varepsilon$, то завершитися.

Навчання ШНМ (метод зворотного розповсюдження) в пакетному режимі

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [–0.5, 0.5] зміщень (порогів) $w_{0k}^{(l)}(n)$ і вагів $w_{kj}^{(l)}(n),\ j \in \overline{1, S^{(l-1)}}, k \in \overline{1, S^{(l)}}, l \in \overline{1, L}$, де $S^{(l)}$ – кількість нейронів у k-м шару, L – кількість шарів.

2. Задається навчальна множина

$$\{(\mathbf{x}_i, \mathbf{d}_i)\},\ i \in \overline{1, I},$$

де $\mathbf{x}_i$ – i-й навчальний вхідний вектор (вектор рейтингів) довжини $S^{(0)}$;

$\mathbf{d}_i$ – i-й навчальний вихідний вектор довжини $S^{(L)}$;

$S^{(0)}$ – кількість нейронів вхідного шару;

$S^{(L)}$ – кількість нейронів вихідного шару;

I – потужність навчальної множини.

3. Обчислення вихідного сигналу для кожного шару (прямий хід).

$$\mathbf{y}_i^{(0)}(n) = \mathbf{x}_i,\ i \in \overline{1, I},$$

$$y_{ik}^{(l)}(n)=f^{(l)}(s_{ik}^{(l)}(n)),\ s_{ik}^{(l)}(n)=\sum_{j=0}^{S^{(l-1)}} w_{kj}^{(l)}(n)y_{ij}^{(l-1)}(n),\ i\in\overline{1,I},k\in\overline{1,S^{(l)}},l\in\overline{1,L},$$

де $S^{(l)}$ – кількість нейронів у l-му шарі;

l – номер шару, L – кількість шарів;

$w_{kj}^{(l)}(n)$ – вага зв'язку від j-го нейрона $l-1$-го шару до k-го нейрона l-го шару в момент часу n;

$y_{ik}^{(l)}(n)$ – вихід k-го нейрона на l-му шарі для i-го навчального вхідного вектора;

$f^{(l)}$ – функція активації нейронів l-го шару.

Вважається, що $y_{i0}^{(l-1)}(n)=1$.

4. Обчислення енергії помилки ШНМ:

$$E(n)=\frac{1}{2I}\sum_{i=1}^{I}\sum_{k=1}^{S^{(L)}} e_{ik}^2(n),\ e_{ik}(n)=y_{ik}^{(L)}(n)-d_{ik}.$$

5. Налаштування синаптичних вагів на основі узагальненого дельта-правила (зворотний хід):

$$w_{kj}^{(l)}(n+1)=w_{kj}^{(l)}(n)-\eta\frac{\partial E(n)}{\partial w_{kj}^{(l)}(n)},$$

де η – параметр, що визначає швидкість навчання (при великому η навчання відбувається швидше, але збільшується небезпека отримати неправильне рішення), $0<\eta<1$.

$$\frac{\partial E(n)}{\partial w_{kj}^{(l)}(n)}=\frac{1}{I}\sum_{i=1}^{I} y_{ij}^{(l-1)}(n)g_{ik}^{(l)}(n),\ j\in\overline{0,S^{(l-1)}},\ k\in\overline{1,S^{(l)}},\ l\in\overline{1,L},$$

$$g_{ik}^{(l)}(n)=\begin{cases} f'^{(L)}(s_{ik}^{(L)}(n))(y_{ik}^{(L)}(n)-d_{ik}), & l=L \\ f'^{(l)}(s_{ik}^{(l)}(n))\sum\limits_{z=1}^{S^{(l+1)}} w_{zk}^{(l+1)}(n)g_{iz}^{(k+1)}(n), & l<L \end{cases}.$$

6. Перевірка умов завершення

Якщо $E(n) < \varepsilon$, то завершення, інакше $n = n + 1$, перехід до 3.

Функціонування ШНМ

У разі мультикласової класифікації:

$$\mathbf{y}^{(0)} = \mathbf{x},$$

$$y_k^{(l)} = f^{(l)}\left(b_k^{(l)} + \sum_{j=1}^{S^{(l-1)}} w_{kj}^{(l)} y_j^{(l-1)} \right),\ k \in \overline{1, S^{(l)}}, l \in \overline{1, L-1}$$

$$y_k^{(L)} = f^{(L)}(s_k^{(L)}) = \text{softmax}(s_k^{(L)}), \qquad s_k^{(L)} = b_k^{(L)} + \sum_{j=1}^{S^{(L-1)}} w_{jk}^{(L)} y_j^{(L-1)},$$

$k \in \overline{1, S^{(L)}}$,

$$k^* = \arg\max_k y_k^{(L)},\ k \in \overline{1, S^{(L)}}.$$

У разі апроксимації:

$$\mathbf{y}^{(0)} = \mathbf{x},$$

$$y_k^{(l)} = f^{(l)}\left(b_k^{(l)} + \sum_{j=1}^{S^{(l-1)}} w_{kj}^{(l)} y_j^{(l-1)} \right),\ k \in \overline{1, S^{(l)}}, l \in \overline{1, L-1},$$

$$y^{(L)} = f^{(L)}(s^{(L)}) = s^{(L)},\ s^{(L)} = b^{(L)} + \sum_{j=1}^{S^{(L-1)}} w_j^{(L)} y_j^{(L-1)},\ j \in \overline{1, S^{(L)}}.$$

На рис. 9.5 наведена машина опорних векторів (SVM) для апроксимації, яка є нерекурентною статичною двошаровою ІНС.

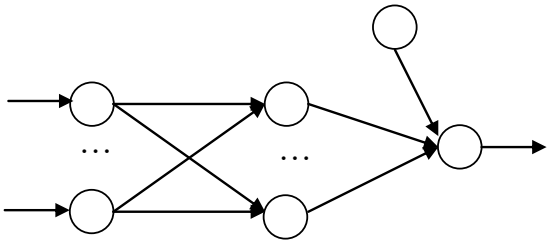


Рис. 9.5 – Машина опорних векторів (SVM)

Основна ідея створення SVM полягає у виборі підмножини навчальних даних як опорних векторів. Ця підмножина представляє стійкі властивості всієї навчальної вибірки. Метод навчання SVM (навчання з учителем) забезпечує мінімізацію емпіричного ризику.

C-апроксимація опорних векторів

Емпіричний ризик (середню величину втрат) у разі C-апроксимації визначено у вигляді:

$$Risk = \frac{1}{I}\sum_{i=1}^{I} l_i \,,\; l_i = \max\left\{0, \left| d_i - \sum_{j=1}^{n-1} w_j \varphi(x_{ij}) - w_0 \right| - \varepsilon \right\},$$

де $\mathbf{x}_i$ – i-й вектор значень ознак (вектор рейтингів);

d_i – i-е значення, яке маємо отримати внаслідок апроксимації (рейтинг);

φ – функція перетворення ознакового простору;

w_j – вага для j-ї ознаки;

w_0 – зміщення;

ε – допустиме відхилення.

Функція втрат у разі апроксимації представлена у вигляді:

$$F(\mathbf{w}) = R(\mathbf{w}) + L(\mathbf{w}),$$

$$L(\mathbf{w}) = \frac{1}{I}\sum_{i=1}^{I} l_i \,,\; R(\mathbf{w}) = \lambda \frac{1}{2}\|\mathbf{w}\|_{l_2}^2 \,,$$

де $L(\mathbf{w})$ – втрати на навчальних даних;

$R(\mathbf{w})$ – втрати регуляризації (регуляризація l_2);

λ – параметр регуляризації, $\lambda > 0$.

Завдання ідентифікації вектора параметрів представлене у вигляді:

$$F(\mathbf{w}) \to \min_{\mathbf{w}} .$$

Це еквівалентно завданню квадратичного програмування у випуклій області, яка в разі C-апроксимації визначена у вигляді:

$$\lambda \frac{1}{2}\|\mathbf{w}\|_{l^2}^2 + \frac{1}{I}\sum_{i=1}^{I}\xi_i^+ + \frac{1}{I}\sum_{i=1}^{I}\xi_i^- \to \min_{\mathbf{w}}$$

за обмежень

$$-\left(d_i - \sum_{j=1}^{n-1} w_j \varphi(x_{ij}) - w_0\right) \le \varepsilon + \xi_i^+,\ \xi_i^+ \ge 0,$$

$$d_i - \sum_{j=1}^{n-1} w_j \varphi(x_{ij}) - w_0 \le \varepsilon + \xi_i^-,\ \xi_i^- \ge 0,$$

де ξ_i^+, ξ_i^- – i-і слабкі змінні.

Двоїста задача квадратичного програмування у випуклій області (максимізація функції Лагранжа) у разі C-апроксимації визначена у вигляді:

$$L(\lambda^+, \lambda^-) = -\varepsilon\sum_{i=1}^{I}(\lambda_i^+ + \lambda_i^-) + \sum_{i=1}^{I}(\lambda_i^- - \lambda_i^+)d_i -$$

$$-\frac{1}{2}\sum_{i=1}^{I}\sum_{z=1}^{I}(\lambda_i^- - \lambda_i^+)(\lambda_z^- - \lambda_z^+)K(\mathbf{x}_i, \mathbf{x}_z) \to \max_{\boldsymbol{\lambda}^+, \boldsymbol{\lambda}^-},$$

за обмежень

$$0 \le \lambda_i^+ \le \frac{1}{\lambda I},\ 0 \le \lambda_i^- \le \frac{1}{\lambda I},$$

$$\sum_{i \in A}(\lambda_i^- - \lambda_i^+) = 0,\ A = \left\{i : 0 \le \lambda_i^+ \le \frac{1}{\lambda I}, 0 \le \lambda_i^- \le \frac{1}{\lambda I}\right\},$$

де λ_i^+ і λ_i^- – i-і множники Лагранжа;

ν – апроксимація опорних векторів.

Задачу квадратичного програмування у випуклій області у разі ν-апроксимації визначено у вигляді:

$$\lambda \frac{1}{2}\|\mathbf{w}\|_{l^2}^2 - \nu\varepsilon + \frac{1}{I}\sum_{i=1}^{I}\xi_i^+ + \frac{1}{I}\sum_{i=1}^{I}\xi_i^- \to \min_{\mathbf{w}}$$

за обмежень

$$-\left(d_i - \sum_{j=1}^{n-1} w_j \varphi(x_{ij}) - w_0\right) \le \varepsilon + \xi_i^+,\ \xi_i^+ \ge 0,$$

$$d_i - \sum_{j=1}^{n-1} w_j \varphi(x_{ij}) - w_0 \le \varepsilon + \xi_i^-,\ \xi_i^- \ge 0,$$

де $\mathbf{x}_i$ – i-й вектор значень ознак (вектор рейтингів);

d_i – i-е значення, яке маємо отримати внаслідок апроксимації (рейтинг);

φ – функція перетворення ознакового простору;

w_j – вага для j-ї ознаки;

w_0 – зміщення;

ξ_i^+, ξ_i^- – i-і слабкі змінні;

λ – параметр регуляризації, $\lambda > 0$;

ν – параметр керування, $\nu \in (0,1]$;

ε – допустиме відхилення.

Двоїста задача квадратичного програмування у випуклій області (максимізація функції Лагранжа) у разі ν-апроксимації визначена у вигляді:

$$L(\lambda^+, \lambda^-) = \sum_{i=1}^{I} (\lambda_i^- - \lambda_i^+) d_i -$$

$$-\frac{1}{2} \sum_{i=1}^{I} \sum_{z=1}^{I} (\lambda_i^- - \lambda_i^+)(\lambda_z^- - \lambda_z^+) K(\mathbf{x}_i, \mathbf{x}_z) \to \max_{\boldsymbol{\lambda}^+, \boldsymbol{\lambda}^-},$$

за обмежень

$$0 \le \lambda_i^+, \lambda_i^- \le \frac{1}{\lambda I},$$

$$\sum_{i \in A} (\lambda_i^- - \lambda_i^+) = 0, \sum_{i \in A} (\lambda_i^- + \lambda_i^+) \le \frac{\nu}{\lambda},\ A = \left\{ i : 0 \le \lambda_i^+ \le \frac{1}{\lambda I}, 0 \le \lambda_i^- \le \frac{1}{\lambda I} \right\},$$

де λ_i^+ і λ_i^- – i-і множники Лагранжа.

Навчання ШНМ

1. Задається навчальна множина $\{(\mathbf{x}_i, d_i)\}\ i \in \overline{1, I}$,

де $\mathbf{x}_i$ – i-й навчальний вхідний вектор (рейтинги предметів) довжиною $S^{(0)}$;

d_i – i-е навчальне вихідне значення (рейтинг предмета);

$S^{(0)}$ – кількість нейронів вхідного шару;

I – потужність навчальної множини;

λ – параметр регуляризації, $\lambda > 0$;

ν – керуючий параметр, $\nu \in (0,1]$;

ε – допустиме відхилення, $\varepsilon > 0$.

2. Обчислення вихідного сигналу для першого шару у вигляді лінійного ядра:

$$K(\mathbf{x}_i, \mathbf{x}_z) = \mathbf{x}_i^T \mathbf{x}_z,\ i, z \in \overline{1, I}.$$

3. Визначаються множники Лагранжа λ_i^+ і λ_i^- шляхом вирішення двоїстої задачі квадратичного програмування у випуклій області. Якщо виконується умова $0 < \lambda_i^+, \lambda_i^- < \frac{1}{\lambda I}$, то $\mathbf{x}_i$ є опорним вектором, тобто знаходиться на межі роздільної смуги.

Перенумеруємо навчальну множину та множники Лагранжа, щоб спочатку йшли опорні вектори і пов'язані з ними множники Лагранжа.

4. Визначає оптимальне зміщення (поріг) w_0, використовуючи будь-яку пару ($\mathbf{x}_m$, d_m) з наступної нерівності:

$$\left| d_m - \left(w_0 + \sum_{z=1}^{S^{(1)}} (\lambda_z^- - \lambda_z^+) K(\mathbf{x}_z, \mathbf{x}_m) \right) \right| \le \varepsilon,$$

де $S^{(1)}$ – кількість опорних векторів (кількість нейронів першого шару).

Функціонування ШНМ

$$y = w_0 + \sum_{z=1}^{S^{(1)}} (\lambda_z^- - \lambda_z^+) K(\mathbf{x}_z, \mathbf{x}).$$

9.1.3. Методи на основі моделей латентних факторів

Модель латентних факторів являє собою рейтингову матрицю у вигляді добутку двох редукованих щільних рейтингових матриць на основі методів матричного розкладання. Під латентними факторами розуміються змінні, що відповідають редукованим рядкам (користувачам) або стовпцям (предметам) рейтингової матриці.

Методи на основі моделей латентних факторів підвищують точність обчислення подібності користувачів або предметів і прискорюють формування груп користувачів або предметів. Наприклад, якщо у двох користувачів дуже мало спільно оцінених предметів, то можна вирахувати відстань між їх векторами, взятими з редукованої по стовпчиках (предметах) щільної рейтингової матриці, що підвищує точність обчислення подібності цих користувачів. Наприклад, порівняння користувачів із редукованою кількістю предметів прискорює формування груп подібних користувачів.

Розглянемо зменшення розмірності матриці рейтингів на основі методу Iterative SVD. У матриці рейтингів $\mathbf{R}$ розмірності $m \times n$ попередньо відсутні значення (наприклад, нулем) і результатом є матриця $\widehat{\mathbf{X}}$. Потім матриця $\widehat{\mathbf{X}}$ розкладається за допомогою методу SVD у вигляді:

$$\widehat{\mathbf{X}} = \widehat{\mathbf{U}}\widehat{\Sigma}\widehat{\mathbf{V}}^T,$$

де $\widehat{\mathbf{U}}$ – вектор лівих сингулярних векторів розмірності $m \times m$;

$\widehat{\mathbf{V}}$ – вектор правих сингулярних векторів розмірності $n \times n$;

$\widehat{\mathbf{\Sigma}} = diag(\sigma_1 ... \sigma_q)$ – матриця сингулярних значень розмірності $m \times n$,

$q = \min\{m, n\}$;

Потім вибираються d найбільших сингулярних значень і обчислюється матриця:

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T,$$

де $\mathbf{U}$ – вектор лівих сингулярних векторів розмірності $m \times d$;

$\mathbf{V}$ – вектор правих сингулярних векторів розмірності $n \times d$;

$\mathbf{\Sigma} = diag(\sigma_1 ... \sigma_d)$ – матриця сингулярних значень розмірності $d \times d$.

Відсутні значення вихідної матриці $\mathbf{R}$ замінюються на відповідні значення матриці $\mathbf{X}$. Матриця $\mathbf{P} = \mathbf{U}\Sigma$ є редукованою рейтинговою матрицею для колаборативної фільтрації на основі користувачів, а $\mathbf{V}$ є редукованою рейтинговою матрицею для колаборативної фільтрації на основі предметів.

Розглянемо зменшення розмірності матриці рейтингів на основі методу Funk SVD.

Рейтингова матриця може бути визначена у вигляді матричного розкладання:

$$\hat{r}_{ui} = \sum_{k=1}^{d} p_{uk} v_{ik}, \; u \in \overline{1,m}, \; i \in \overline{1,n}.$$

Функція втрат у разі моделі латентних факторів:

$$F(\mathbf{P},\mathbf{V}) = R(\mathbf{P},\mathbf{V}) + E(\mathbf{P},\mathbf{V}),$$

$$E(\mathbf{P},\mathbf{V}) = \frac{1}{2} \sum_{(u,i)\in S} (r_{ui} - \hat{r}_{ui})^2,$$

де $E(\mathbf{P},\mathbf{V})$ – втрати на навчальних даних (сума середньоквадратичних помилок по всій множині спостережуваних рейтингів);

$R(\mathbf{P},\mathbf{V})$ – втрати регуляризації;

S – множина рейтингів.

Втрати регуляризації у разі:

- відсутності регуляризації

$$R(\mathbf{P},\mathbf{V}) = 0,$$

- регуляризації l_1 (регресія LASSO)

$$R(\mathbf{P},\mathbf{V}) = \lambda\left(\|\mathbf{P}\|_{l_1} + \|\mathbf{V}\|_{l_1}\right),$$

- регуляризації l_2 (регресія Ridge)

$$R(\mathbf{P},\mathbf{V}) = \lambda \frac{1}{2}\left(\|\mathbf{P}\|_{l_2}^2 + \|\mathbf{V}\|_{l_2}^2\right),$$

- комбінації регуляризації l_1 і l_2 (еластична мережа)

$$R(\mathbf{P},\mathbf{V}) = \lambda\left(\frac{1-\rho}{2}\left(\|\mathbf{P}\|_{l_2}^2 + \|\mathbf{V}\|_{l_2}^2\right) + \rho\left(\|\mathbf{P}\|_{l_1} + \|\mathbf{V}\|_{l_1}\right)\right),$$

де λ – параметр регуляризації, $\lambda > 0$;

ρ – параметр змішування, $0 < \rho < 1$.

Завдання ідентифікації вектора параметрів представлено у вигляді:

$$F(\mathbf{P},\mathbf{V}) \to \min_{\mathbf{P},\mathbf{V}}.$$

Метод градієнтного спуску без регуляризації

Ініціалізується за допомогою рівномірного розподілу на інтервалі (0,1) матриця **P** розмірності $m \times d$ і матриця **V** розмірності $n \times d$:

$$\breve{p}_{uk} = p_{uk} + \eta\left(r_{ui} - \hat{r}_{ui}\right)v_{ik},\ u \in \overline{1,m},\ k \in \overline{1,d},$$
$$\breve{v}_{ik} = v_{ik} + \eta\left(r_{ui} - \hat{r}_{ui}\right)p_{uk},\ i \in \overline{1,n},\ k \in \overline{1,d},$$
$$p_{uk} = \breve{p}_{uk},\ u \in \overline{1,m},\ k \in \overline{1,d},$$
$$v_{ik} = \breve{v}_{ik},\ i \in \overline{1,n},\ k \in \overline{1,d}.$$

Метод градієнтного спуску з регуляризацією l_1 (регресією LASSO)

Ініціалізується за допомогою рівномірного розподілу на інтервалі (0,1) матриця **P** розмірності $m \times d$ і матриця **V** розмірності $n \times d$,

$$\breve{p}_{uk} = p_{uk} + \eta\left(-\lambda\left(\operatorname{sgn}(p_{uk}) + \operatorname{sgn}(v_{ik})\right) + r_{ui} - \hat{r}_{ui}\right)v_{ik},\ u \in \overline{1,m},\ k \in \overline{1,d},$$
$$\breve{v}_{ki} = v_{ki} + \eta\left(-\lambda\left(\operatorname{sgn}(p_{uk}) + \operatorname{sgn}(v_{ik})\right) + r_{ui} - \hat{r}_{ui}\right)p_{uk},\ i \in \overline{1,n},\ k \in \overline{1,d},$$
$$p_{uk} = \breve{p}_{uk},\ u \in \overline{1,m},\ k \in \overline{1,d},$$
$$v_{ik} = \breve{v}_{ik},\ i \in \overline{1,n},\ k \in \overline{1,d}.$$

Метод градієнтного спуску з регуляризацією l_2 (регресією Ridge)

Ініціалізується за допомогою рівномірного розподілу на інтервалі (0,1) матриця **P** розмірності $m \times d$ і матриця **V** розмірності $n \times d$:

$$\breve{p}_{uk} = p_{uk} + \eta\left(-\lambda\left(p_{uk} + v_{ik}\right) + r_{ui} - \hat{r}_{ui}\right)v_{ik},\ u \in \overline{1,m},\ k \in \overline{1,d},$$
$$\breve{v}_{ik} = v_{ik} + \eta\left(-\lambda\left(p_{uk} + v_{ik}\right) + r_{ui} - \hat{r}_{ui}\right)p_{uk},\ i \in \overline{1,n},\ k \in \overline{1,d},$$
$$p_{uk} = \breve{p}_{uk},\ u \in \overline{1,m},\ k \in \overline{1,d},$$
$$v_{ik} = \breve{v}_{ik},\ i \in \overline{1,n},\ k \in \overline{1,d}.$$

Метод градієнтного спуску з еластичною мережею

Ініціалізується за допомогою рівномірного розподілу на інтервалі (0,1) матриця **P** розмірності $m \times d$ і матриця **V** розмірності $n \times d$:

$$\breve{p}_{uk} = p_{uk} + \eta\left(-\lambda\left((1-\rho)\left(p_{uk} + v_{ik}\right) + \rho\left(\operatorname{sgn}(p_{uk}) + \operatorname{sgn}(v_{ik})\right)\right) + r_{ui} - \hat{r}_{ui}\right)v_{ik},$$
$u \in \overline{1,m},\ k \in \overline{1,d}$,

$$\breve{v}_{ki} = v_{ki} + \eta\left(-\lambda\left((1-\rho)\left(p_{uk} + v_{ik}\right) + \rho\left(\operatorname{sgn}(p_{uk}) + \operatorname{sgn}(v_{ik})\right)\right) + r_{ui} - \hat{r}_{ui}\right)p_{uk},$$
$i \in \overline{1,n},\ k \in \overline{1,d}$,

$p_{uk} = \breve{p}_{uk},\ u \in \overline{1,m},\ k \in \overline{1,d}$,

$v_{ik} = \breve{v}_{ik},\ i \in \overline{1,n},\ k \in \overline{1,d}$.

Розглянемо зменшення розмірності матриці рейтингів на основі методу невід'ємної матричної факторизації (NMF).

Рейтингова матриця може бути визначена у вигляді матричного розкладання:

$$\hat{r}_{ui} = \sum_{k=1}^{d} p_{uk} v_{ik},\ u \in \overline{1,m},\ i \in \overline{1,n}.$$

Функція втрат у разі моделі латентних факторів:

$$F(\mathbf{P},\mathbf{V}) = R(\mathbf{P},\mathbf{V}) + E(\mathbf{P},\mathbf{V}) \text{ з обмеженнями } \mathbf{P} \geq 0, \mathbf{V} \geq 0,$$

$$E(\mathbf{P},\mathbf{V}) = \frac{1}{2}\sum_{(u,i)\in S}(r_{ui} - \hat{r}_{ui})^2,$$

де $E(\mathbf{P},\mathbf{V})$ – втрати на навчальних даних (сума середньоквадратичних помилок по всій множині спостережуваних рейтингів);

$R(\mathbf{P},\mathbf{V})$ – втрати регуляризації;

S – множина рейтингів.

Втрати регуляризації у разі:

- відсутності регуляризації

$$R(\mathbf{P},\mathbf{V}) = 0,$$

- регуляризації l_1 (регресія LASSO)

$$R(\mathbf{P},\mathbf{V}) = \lambda\left(\|\mathbf{P}\|_{l_1} + \|\mathbf{V}\|_{l_1}\right),$$

- регуляризації l_2 (регресія Ridge)

$$R(\mathbf{P},\mathbf{V}) = \lambda \frac{1}{2}\left(\|\mathbf{P}\|_{l_2}^2 + \|\mathbf{V}\|_{l_2}^2\right),$$

- комбінації регуляризації l_1 і l_2 (еластична мережа)

$$R(\mathbf{P},\mathbf{V}) = \lambda\left(\frac{1-\rho}{2}\left(\|\mathbf{P}\|_{l_2}^2 + \|\mathbf{V}\|_{l_2}^2\right) + \rho\left(\|\mathbf{P}\|_{l_1} + \|\mathbf{V}\|_{l_1}\right)\right),$$

де λ – параметр регуляризації, $\lambda > 0$;

ρ – параметр змішування, $0 < \rho < 1$.

Завдання ідентифікації вектора параметрів представлено у вигляді:

$$F(\mathbf{P},\mathbf{V}) \to \min_{\mathbf{P},\mathbf{V}}.$$

Метод NMF без регуляризації

Ініціалізується за допомогою рівномірного розподілу на інтервалі (0,1) матриця $\mathbf{P}$ розмірності $m \times d$ і матриця $\mathbf{V}$ розмірності $n \times d$.

Оновлення матриці $\mathbf{P}$ за формулами

$$\mathbf{A} = \mathbf{R}\mathbf{V},\ \mathbf{B} = \mathbf{P}\mathbf{V}^T\mathbf{V},$$

$$p_{uk} = p_{uk}\frac{a_{uk}}{b_{uk}},\ u \in \overline{1,m},\ k \in \overline{1,d}.$$

Оновлюється матриця $\mathbf{V}$ за формулами

$$\mathbf{A} = \mathbf{R}^T\mathbf{P},\ \mathbf{B} = \mathbf{V}\mathbf{P}^T\widehat{\mathbf{P}},$$

$$v_{ik} = v_{ik}\frac{a_{ik}}{b_{ik}},\ i \in \overline{1,n},\ k \in \overline{1,d}.$$

Метод NMF з регуляризацією l_1 (регресією LASSO).

Ініціалізується за допомогою рівномірного розподілу на інтервалі (0,1) матриця $\mathbf{P}$ розмірності $m \times d$ і матриця $\mathbf{V}$ розмірності $n \times d$.

Оновлення матриці $\mathbf{P}$ за формулами:

$$\mathbf{A} = \mathbf{R}\mathbf{V},\ \mathbf{B} = \mathbf{P}\mathbf{V}^T\mathbf{V},$$

$$p_{uk} = \max\left\{ p_{uk} \frac{a_{uk} - \lambda \operatorname{sgn}(p_{uk})}{b_{uk}}, 0 \right\},\ u \in \overline{1,m},\ k \in \overline{1,d}\ .$$

Оновлюється матриця $\mathbf{V}$ за формулами:

$$\mathbf{A} = \mathbf{R}^T\mathbf{P},\ \mathbf{B} = \mathbf{V}\mathbf{P}^T\widehat{\mathbf{P}},$$

$$v_{ik} = \max\left\{ v_{ik} \frac{a_{ik} - \lambda \operatorname{sgn}(v_{ik})}{b_{ik}}, 0 \right\},\ i \in \overline{1,n},\ k \in \overline{1,d}\ .$$

Метод NMF з регулюванням l_2 (регресією Ridge)

Ініціалізується за допомогою рівномірного розподілу на інтервалі (0,1) матриця $\mathbf{P}$ розмірності $m \times d$ і матриця $\mathbf{V}$ розмірності $n \times d$.

Оновлення матриці $\mathbf{P}$ за формулами

$$\mathbf{A} = \mathbf{R}\mathbf{V},\ \mathbf{B} = \mathbf{P}\mathbf{V}^T\mathbf{V},$$

$$p_{uk} = \max\left\{ p_{uk} \frac{a_{uk} - \lambda p_{uk}}{b_{uk}}, 0 \right\},\ u \in \overline{1,m},\ k \in \overline{1,d}\ .$$

Оновлюється матриця $\mathbf{V}$ за формулами:

$$\mathbf{A} = \mathbf{R}^T\mathbf{P},\ \mathbf{B} = \mathbf{V}\mathbf{P}^T\widehat{\mathbf{P}},$$

$$v_{ik} = \max\left\{ v_{ik} \frac{a_{ik} - \lambda v_{ik}}{b_{ik}}, 0 \right\},\ i \in \overline{1,n},\ k \in \overline{1,d}\ .$$

Метод NMF еластичною мережею

Ініціалізується за допомогою рівномірного розподілу на інтервалі (0,1) матриця $\mathbf{P}$ розмірності $m \times d$ і матриця $\mathbf{V}$ розмірності $n \times d$.

Оновлення матриці $\mathbf{P}$ за формулами:

$$\mathbf{A} = \mathbf{R}\mathbf{V},\ \mathbf{B} = \mathbf{P}\mathbf{V}^T\mathbf{V},$$

$$p_{uk} = \max\left\{ p_{uk} \frac{a_{uk} - \lambda((1-\rho)p_{uk} + \rho \operatorname{sgn}(p_{uk}))}{b_{uk}}, 0 \right\},\ u \in \overline{1,m},\ k \in \overline{1,d}\ .$$

Оновлюється матриця $\mathbf{V}$ за формулами:

$$\mathbf{A} = \mathbf{R}^T\mathbf{P},\ \mathbf{B} = \mathbf{V}\mathbf{P}^T\widehat{\mathbf{P}},$$

$$v_{ik} = \max\left\{v_{ik}\frac{a_{ik} - \lambda((1-\rho)v_{ik} + \rho\,\mathrm{sgn}(v_{ik}))}{b_{ik}}, 0\right\},\ i \in \overline{1,n},\ k \in \overline{1,d}.$$

Для методів SVM та NMF остаточний рейтинг розраховується у вигляді:

$$\widehat{r}_{uk} = \frac{\sum_{c=1}^{n}\mathrm{Cosine}(c,k)r_{uc}}{\sum_{c=1}^{n}\mathrm{Cosine}(c,k)},\ \mathrm{Cosine}(c,k) = \frac{\sum_{c=1}^{d}\sum_{k=1}^{d}(v_{ic})(v_{ik})}{\sqrt{\sum_{c=1}^{d}(v_{ic})^2}\sqrt{\sum_{k=1}^{d}(v_{ik})^2}}.$$

9.2. Методи контентної фільтрації

У методах контентної фільтрації явні і неявні рейтинги, пов'язані з конкретним користувачем, і опис предметів, що містить атрибути предметів, використовуються для формування рекомендацій. Прикладами таких методів у машинному навчанні є класифікатори найближчих сусідів, дерева рішень, методи знаходження взаємозв'язків, байєсівські класифікатори (наприклад, у разі аналізу текстових даних, якщо ключові слова з вагою BoW, то використовується мультиноміальний наївний Байєс, додатковий наївний Байєс, категоріальний наївний Байєс).

Зауваження. У разі класифікатора найближчих сусідів для прискорення обчислень рекомендується попередньо провести кластеризацію (наприклад, у разі аналізу текстових даних, якщо ключові слова з вагою BoW, то використовується метод LDA), щоб розбити предмети на кластери й обчислювати схожість тільки в тому кластері, до якого належить предмет.

Методи контентної фільтрації корисні, коли деякий користувач вказав високий рейтинг для деякого предмета, але для цього предмета відсутні рейтинги, зазначені іншими користувачами, тому методи колаборативної фільтрації не можуть використовуватися. Однак якщо опис цього предмета містить такі самі атрибути, як і інші предмети, то ці інші предмети можна порекомендувати цьому користувачу.

Методи контентної фільтрації ефективні для нових предметів, коли для цих предметів немає рейтингових даних, але є атрибути. Якщо інші предмети з аналогічними атрибутами мають рейтинги, то модель може використовувати ці рейтинги в поєднанні з атрибутами нових предметів, щоб давати рекомендації.

Рейтинги, пов'язані з іншими користувачами, зазвичай не використовуються під час обчислення оцінок рейтингів, пов'язаних із цим користувачем.

Методи контентної фільтрації особливо добре підходять для формування рекомендацій у текстових та неструктурованих областях (наприклад, для роботи з вебсторінками, для аналізу новин і статей).

Методи контентної фільтрації мають такі недоліки:

1. Навчена модель специфічна для конкретного користувача (наприклад, предмет з атрибутами, відмінними від атрибутів предметів, раніше замовлених цим користувачем, ніколи не буде обраний). Це приводить до зменшення різноманітності рекомендованих предметів, що є небажаним.

2. Методи контентної фільтрації неефективні під час формування рекомендацій для нового користувача. Це пов'язано з тим, що навчена модель повинна використовувати рейтинги, пов'язані з цим користувачем.

Методи класифікації тексту та регресійного моделювання є найбільш використовуваними методами контентної фільтрації.

На рис. 9.6 наведений приклад використання методів контентної фільтрації під час аналізу текстових даних фільму «Бетмен проти Супермена» на IMDB.

Рис. 9.6 – Приклад аналізу текстових даних фільму «Бетмен проти Супермена» на IMDB

Рекомендаційні системи контентної фільтрації, що працюють із текстовими даними, містять такі компоненти:

1. Попередня обробка.

1.1. З текстових даних виводяться слова.

1.2. Проводиться очищення (відбір ключових слів) згідно з такими етапами:

- видалення стоп-слів. Стоп-слова не належать до конкретного предмета, але є загальною частиною лексики мови. Стоп-слова зазвичай є високочастотними словами, саме тому можуть бути віднесені до ключових слів. Зазвичай артиклі, прийменники, сполучники і займенники розглядаються як стоп-слова (наприклад, в англійській мові слова «a», «an» і «the» відносять до стоп-слів);
- стемінг (зі слів витягуються корені, щоб об'єднати однокореневі слова);
- витяг фраз (виявлення словосполучень типу «хот-дог»).

1.3. Відібрані ключові слова перетворюються на вектори, і кожному слову присвоюється ваговий коефіцієнт у вигляді BoW або TF-IDF:

$$BoW(t,d) = n_t,$$

$$tf - idf(t,d,D) = tf(t,d)idf(t,D),$$

$$tf(t,d) = \frac{n_t}{\sum_{k=1}^{l_d} n_k},\ idf(t,D) = \ln\frac{|D|}{|\{d_i \in D \mid t \in d_i\}|},$$

де *BoW* (Bag-of-words – мішок слів) – кількість входжень певного слова в документ;

tf (*term frequency* – частота слова) – відношення кількості входжень певного слова до загальної кількості слів документа. Отже, оцінюється важливість слова в межах окремого документа,

idf (*inverse document frequency* – зворотна частота документа) – інверсія частоти, з якою певне слово зустрічається в документах колекції. Облік *idf* зменшує кількість широковживаних слів. Для кожного унікального слова в межах певної колекції документів існує лише одне значення *idf*,

n_t – кількість входжень слова t до документа d;

l_d – кількість слів у документі d;

|D| – кількість документів у колекції;

$|\{d_i \in D \mid t \in d_i\}|$ – кількість документів з колекції D, в яких зустрічаються слова t (коли $n_t \neq 0$).

2. Навчання моделі. Явні і неявні рейтинги й атрибути предметів (ключові слова зі значенням *BoW* або *tf-idf*) використовуються як навчальні дані для навчальних моделей, причому атрибути предметів використовуються як навчальні вхідні дані, а рейтинги – як навчальні вихідні дані. Цей етап аналогічний до навчання моделей класифікації або апроксимації.

3. Прогнозування. На цьому етапі навчальна модель використовується для формування рекомендацій з предметів для конкретних користувачів (обчислюються оцінки рейтингів предметів).

Зауваження. Вибір інформативних ознак може виконуватися методами машинного навчання (наприклад, у разі аналізу текстових даних, якщо ключові слова зі значенням *BoW*, то для відбору ключових слів використовуються методи на основі χ^2-тесту (критерію згоди Пірсона), *MRMR* (minimum redundancy – maximum relevance) на основі взаємної ентропії, ReliefF (Neighborhood Component Feature Selection) на основі найближчого сусідства).

9.3. Методи на основі знань

Брак методів колаборативної фільтрації – у разі малого обсягу даних, нових предметів або нових користувачів неефективний. Брак методів контентної фільтрації – у разі малого обсягу даних, нових користувачів неефективний.

Методи на основі знань корисні, коли:

- користувач хоче інтерактивно зазначити свої вимоги;
- важко отримати рейтинг, пов’язаний із користувачем;
- рейтинг, пов’язаний з користувачем, змінюється з плином часу.

Рейтинги, пов’язані з цим користувачем, не використовуються в цих методах.

Методи на основі знань добре адаптовані до конкретної предметної галузі. У рекомендаційних системах на основі знань користувачу надається більший контроль над процесом формування рекомендації через зазначення вимог. Рекомендаційні системи на основі знань можуть бути на основі обмежень (constraint-based system) або на основі прикладів (case-based system). У рекомендаційних системах на основі обмежень (рис. 9.7) користувач спочатку вказує

обмеження на атрибути предметів, після чого система перетворює ці обмеження до свого внутрішнього подання і за допомогою правил з бази знань, що відображають перетворені обмеження на предмети (наприклад, обмеження у внутрішньому перегляді «вартість < 10000 $ ∧ марка = Шкода» відображається у множині автомобілів, що задовольняють це обмеження), повертає результат, а потім користувач може змінювати обмеження (наприклад, додати колір автомобіля). У рекомендаційних системах на основі прикладів (рис. 9.8) користувач спочатку вказує цільові предмети, на які схожі бажані для нього предмети, після чого система перетворює опис цільових предметів до свого внутрішнього подання та за допомогою правил із бази знань, що відображають перетворений опис цільових предметів на предмети (наприклад, опис цільового предмета у внутрішньому уявленні «машина Джеймса Бонда» відображається множина автомобілів, що задовольняють цей цільовий предмет), повертає результат, а потім користувач може змінювати атрибути цільових предметів (наприклад, додати марку автомобіля Джеймса Бонда), причому такі зміни називаються критикою.

Брак методів на основі знань – налаштований лише на певну предметну область, на відміну від методів колаборативної та контентної фільтрації.

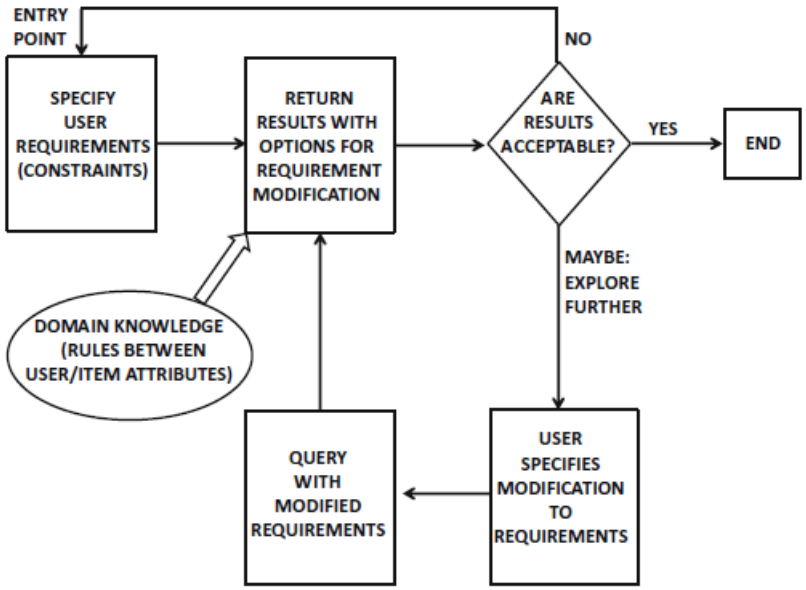


Рис. 9.7 – Інтерактивний процес у рекомендаційній системі на основі обмежень

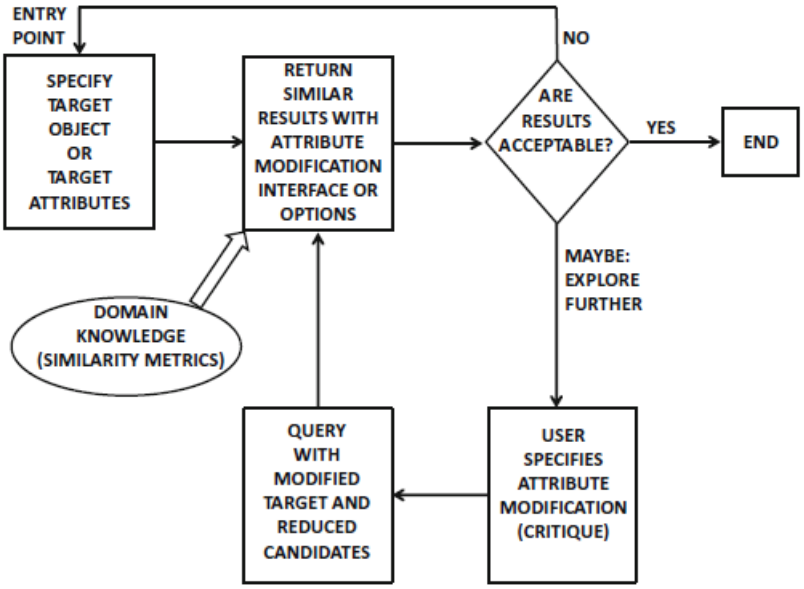


Рис. 9.8 – Інтерактивний процес у рекомендаційній системі на основі прикладів

9.4. Контекстно-залежні методи

Контекст користувача може бути виявлений різними способами. У деяких випадках контекст легко отримати (наприклад, GPS-приймач на мобільному телефоні вказує місце розташування клієнта, а позначка часу транзакції клієнта вказує час), і такі методи називаються методами неявного збору. В інших випадках контекст не так легко отримати (наприклад, використовуються опитування, методи інтелектуального аналізу даних, або логічний висновок).

У традиційній рекомендаційній системі з множиною користувачів U і множиною предметів I декартовий добуток $U \times I$ відображається в рейтингу. Результатом цього відображення є неповна рейтингова матриця розмірністю $|U| \times |I|$. У контекстно-залежній рекомендаційній системі з множиною користувачів U, множиною предметів I і множиною контекстів C декартовий добуток $U \times I \times C$ відображається в рейтингу. У випадку кількох типів контексту (наприклад, час і локація) декартовий добуток $U \times I \times C1 \times \ldots \times C_n$ буде показано у рейтингу.

На рис. 9.9 представлена тривимірна рейтингова матриця, що містить п’ять користувачів, чотири предмети (фільми) і контекст (час).

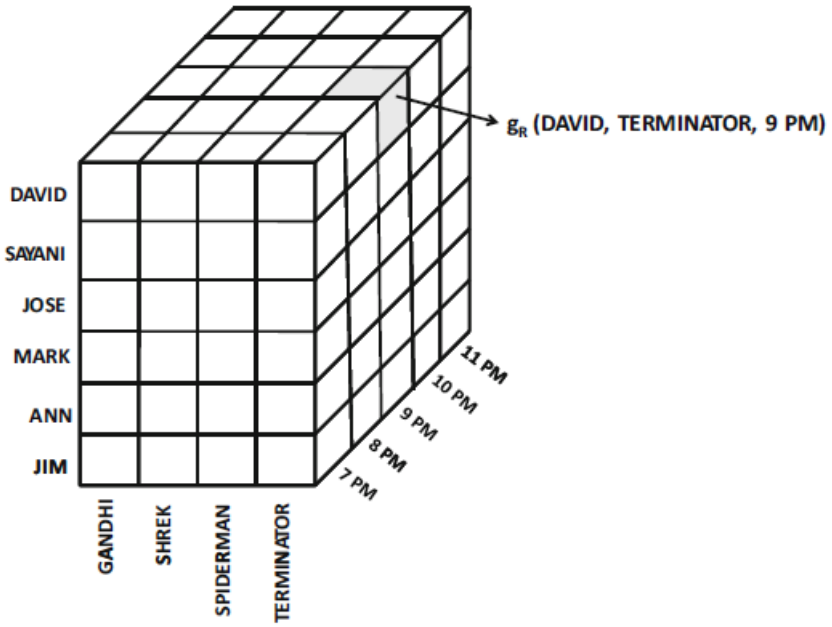


Рис. 9.9 – Тривимірна рейтингова матриця

Існує три основні способи виконання контекстно-залежних рекомендацій:

1. Передфільтрація. Знижується розмірність w-мірного куба до традиційної двовимірної матриці рейтингів перед застосуванням методу колаборативної фільтрації (наприклад, беруться зрізи (сегменти) за значеннями контексту або відбувається агрегування шляхом усереднення на інтервалі значень контексту).

2. Постфільтрація:

- на першому етапі контекст ігнорується (наприклад, у разі явних рейтингів відбувається агрегування шляхом усереднення по всьому контекстному виміру без урахування значень контексту), і результатом першого етапу є традиційна двовимірна матриця рейтингів;
- на другому етапі отримана матриця рейтингів коригується шляхом видалення стовпчиків, що відповідають предметам, які нерелевантні в цьому контексті. Для більшої ефективності коригування будується модель, яка дає змогу отримати ймовірнісну оцінку $P(u,i,c)$ того, що користувачеві u сподобається предмет i в контексті c.

3. Контекстне моделювання. Здійснюється модифікація методів колаборативної або контентної фільтрації шляхом включення контексту як додаткової ознаки. Цей підхід вимагає значних обчислювальних ресурсів, але дає найкращий результат, коли доступний великий обсяг даних.

ЛАБОРАТОРНІ РОБОТИ

Лабораторна робота № 1

Тема: Вивчення основ поповнення знань на основі псевдофізичної логіки.

Перша частина

Завдання: поповнення знань на основі псевдофізичної логіки часу та логічного виведення (Додаток А).

1. Представити обраний текст з одного, двох або трьох речень на основі псевдофізичної логіки часу.
2. Поповнити структуру обраного тексту на основі псевдофізичної логіки часу та логічного виведення.

Приклад для псевдофізичної логіки часу

Представлено такий текст: «Літак здійснив посадку о 15 год 20 хв. Через 10 хв був поданий трап, і потім пасажири покинули літак. Прийшов автобус, і через 2 хв пасажири були вже в будівлі аеровокзалу». У тексті виділяються події:

p_1 – літак здійснив посадку;

p_2 – подача трапа;

p_3 – вихід пасажирів;

p_4 – подача автобуса;

p_5 – поява пасажирів у будівлі аеровокзалу.

Структура тексту представлена у вигляді:

$$ST = (\mu_к(p_1)R^1_{4,M}(t)) \wedge (\mu_к(p_1)R^1_{3,M}(10, \text{хв})\mu_п(p_2)) \wedge (p_3 R^2_2 p_2) \wedge (p_5 R^2_2 p_4) \wedge$$
$$\wedge (\mu_к(p_4)R^1_{3,M}(2, \text{хв})\mu_п(p_5)),$$

де t = 15 год 20 хв;

$\mu_п$ – маркер початку інтервальної події;

$\mu_к$ – маркер кінця інтервальної події;

M – індикатор метричної логіки;

верхній індекс означає клас відношень; 1 – псевдофізична логіка точкових часових відношень; 2 – псевдофізична логіка інтервальних часових відношень, індекси відношень позначені згідно з додатком А.

Поповнимо структуру тексту на основі правил виведення новими фактами:

1. $(\mu_{к}(p_1)R^1_{4,M}(t)),(\mu_{к}(p_1)R^1_{3,M}(10,\text{хв})\mu_{п}(p_2)) \to \mu_{п}(p_2)R^1_4(t'))$,
   $t' =$15год 30хв,
2. $(\mu_{к}(p_1)R^1_{3,M}(10,\text{хв})\mu_{п}(p_2)) \to p_1R^2_1p_2$,
3. $p_1R^2_1p_2 \to p_2R^2_2p_1$,
4. $p_3R^2_2p_2 \to p_2R^2_1p_3$,
5. $p_1R^2_1p_2, p_2R^2_1p_3 \to p_1R^2_1p_3$,
6. $(\mu_{к}(p_4)R^1_{3,M}(2,\text{хв})\mu_{п}(p_5)) \to p_4R^2_1p_5$.

Приєднавши виведені факти до структури тексту, отримаємо:

$$ST^* = ST \cup \{\mu_{п}(p_2)R^1_{4,M}(t') \wedge (p_1R^2_1p_2) \wedge (p_2R^2_2p_1) \wedge (p_2R^2_1p_3) \wedge (p_1R^2_1p_3) \wedge (p_4R^2_1p_5)\}.$$

Отримано таке доповнення до початкового тексту: «Трап був поданий о 15 год 30 хв. Посадка літака суворо передує подачі трапу. Подача трапу суворо слідує за посадкою літака. Подача трапу суворо передує виходу пасажирів. Посадка літака суворо передує виходу пасажирів. Подача автобуса суворо передує появі пасажирів у будівлі аеровокзалу».

Друга частина

Завдання: поповнення знань на основі псевдофізичної логіки простору та логічного виведення.

1. Представити обраний текст з одного, двох або трьох речень на основі псевдофізичної логіки простору.

2. Поповнити структуру обраного тексту на основі псевдофізичної логіки простору та логічного виведення.

Приклад для псевдофізичної логіки простору

Представлено такий текст: «Капелюх знаходиться в середині капелюшної коробки. Капелюшна коробка знаходиться у шафі». У тексті виділяються об'єкти:

O_1 – капелюшна коробка;

O_2 – шафа;

O_3 – капелюх.

Структура тексту представлена у вигляді:

$$ST = (O_1 R_7^5 O_2) \wedge (O_3 R_{34}^5 O_1).$$

Верхній індекс означає клас відношень, 5 – псевдофізична логіка взаємного розташування об'єктів, індекси відношень позначені згідно з додатком А.

Поповнимо структуру тексту на основі правил виведення новими фактами:

$$O_1 R_7^5 O_2 \wedge O_3 R_{34}^5 O_1 \to O_3 R_{34}^5 O_2.$$

Приєднавши виведені факти до структури тексту, отримаємо:

$$ST^* = ST \cup \{(O_3 R_{34}^5 O_2)\}.$$

Отримано таке доповнення до початкового тексту: «Капелюх знаходиться у шафі».

Третя частина

Завдання: поповнення знань на основі каузальної псевдофізичної логіки та логічного виведення.

1. Представити обраний текст з одного, двох або трьох речень на основі каузальної псевдофізичної логіки.

2. Поповнити структуру обраного тексту на основі каузальної псевдофізичної логіки та логічного виведення.

Приклад для каузальної псевдофізичної логіки

Представлено такий текст: «Робочий робив нарізку болта на верстаті». У тексті виділяються події:

p_1 – робочий робив нарізку болта на верстаті;

p_2 – робочий бере болт;

p_3 – робочий встановлює болт на верстат.

Структура тексту ST представлена у вигляді:

$$ST = (p_2 R_3^6 p_1) \wedge (p_3 R_3^6 p_1) \wedge (p_2 R_3^6 p_3).$$

R_3^6 – обумовлює причинне відношення між подіями p і q (Додаток А), верхній індекс означає клас відношень; 6 – псевдофізична логіка каузальних відношень (причино-наслідкових відношень); індекси відношень позначені згідно з додатком А.

Поповнимо структуру тексту ST на основі правил виведення новими фактами:

$p_2 R_3^6 p_3 \rightarrow p_2 R_1^1 p_3$.

R_1^1 – подія p відбувається раніше події q (Додаток А).

Приєднавши виведені факти до структури тексту, отримаємо поповнення структури тексту ST:

$$ST^* = ST \cup \{p_2 R_1^1 p_3\}.$$

Отримано таке доповнення до початкового тексту: «Робочий бере болт раніше, ніж встановлює болт на верстат».

Четверта частина

Завдання: поповнення знань на основі псевдофізичної логіки дії та логічного виведення.

1. Представити обраний текст з одного, двох або трьох речень на основі псевдофізичної логіки дій.

2. Поповнити структуру обраного тексту на основі псевдофізичної логіки дій та логічного виведення.

Приклад для псевдофізичної логіки дій

Поставлено такий текст: «Він увійшов у вагон, і поїзд доїхав до Києва за 3 години».

У тексті виділяються дії:

d_1 – увійшов;

d_2 – доїхав;

d_3 – знаходиться в.

У тексті виділяються суб'єкти:

S_1 – він.

У тексті виділяються об'єкти:

O_1 – вагон;

O_2 – поїзд.

У тексті виділяються локалізації:

l_1 – Київ.

Структура тексту представлена у вигляді:

$$ST = d_1(S_1, O_1, t_1) \wedge (O_1 R_{11}^5 O_2) \wedge d_2(O_2, l_1, t_1 + \Delta t),$$ де Δt = 3 години.

Поповнимо структуру тексту на основі правил виводу новими фактами:

$$d_1(S_1, O_1, t_1) \to d_1 R_2^7 d_3(S_1, O_1, t_1 + \Delta t),$$
$$d_1(S_1, O_1, t_1), (O_1 R_{11}^5 O_2) \to d_1(S_1, O_2, t_1),$$
$$d_1(S_1, O_1, t_1) \to d_3(S_1, O_2, t_1 + \Delta t),$$
$$d_3(S_1, O_2, t_1 + \Delta t), d_2(O_2, l_1, t_1 + \Delta t) \to d_2(S_1, l_1, t_1 + \Delta t).$$

Приєднавши виведені факти до структури тексту, отримаємо:

$$ST^* = ST \cup \{d_1 R_2^7 d_3(S_1, O_1, t_1 + \Delta t) \wedge d_1(S_1, O_2, t_1) \wedge d_3(S_1, O_2, t_1 + \Delta t) \wedge d_2(S_1, l_1, t_1 + \Delta t)\}.$$

Отримано таке доповнення до початкового тексту: «Результатом входу є знаходження його у поїзді протягом 3-х годин. Він увійшов у поїзд. Він знаходився у поїзді 3 години. Він доїхав до Києва за 3 години».

Лабораторна робота № 2

Тема: Вивчення основ побудови експертних систем зі знаннями у вигляді продукційних правил (Додаток Б, В).

Перша частина

Завдання: розробити експертну систему, що використовує пряме виведення з використанням фактів та шаблонів фактів у вигляді одномірних предикатів, продукційних правил.

Перший варіант – факти визначаються без використання оператора deffacts

```
!pip install clipspy
import clips
# визначити шаблони фактів
DEFTEMPLATE_STRING1 = """
(deftemplate fact1
   "Fact1"
   (slot active_person (type SYMBOL)))
"""
DEFTEMPLATE_STRING2 = """
(deftemplate fact2
   "Fact2"
   (slot love_skiing (type SYMBOL)))
"""
DEFTEMPLATE_STRING3 = """
(deftemplate fact3
   "Fact3"
   (slot love_sea (type SYMBOL)))
"""
DEFTEMPLATE_STRING4 = """
(deftemplate fact4
   "Fact4"
   (slot vacation_winter (type SYMBOL)))
"""
DEFTEMPLATE_STRING5 = """
(deftemplate fact5
   "Fact5"
   (slot vacation_summer (type SYMBOL)))
```

```
"""
DEFTEMPLATE_STRING6 = """
(deftemplate fact6
   "Fact6"
   (slot goto_mountains (type SYMBOL)))
"""
# визначити правила
DEFRULE_STRING1 = """
(defrule rule1
   "Rule1"
   (fact4 (vacation_winter TRUE))
   (fact1 (active_person TRUE))
   =>
   (assert (fact6 (goto_mountains TRUE)))
   (println "go to mountains"))
"""
DEFRULE_STRING2 = """
(defrule rule2
   "Rule2"
   (fact3 (love_sea TRUE))
   =>
   (assert (fact5 (vacation_summer TRUE)))
   (println "summer vacation"))
"""
DEFRULE_STRING3 = """
(defrule rule3
   "Rule3"
   (fact2 (love_skiing TRUE))
   =>
   (assert (fact4 (vacation_winter TRUE)))
   (println "winter vacation"))
"""
environment = clips.Environment()
# визначити шаблони фактів
environment.build(DEFTEMPLATE_STRING1)
environment.build(DEFTEMPLATE_STRING2)
environment.build(DEFTEMPLATE_STRING3)
environment.build(DEFTEMPLATE_STRING4)
```

```
environment.build(DEFTEMPLATE_STRING5)
environment.build(DEFTEMPLATE_STRING6)
# визначити правила
environment.build(DEFRULE_STRING1)
environment.build(DEFRULE_STRING2)
environment.build(DEFRULE_STRING3)
# повернути шаблони фактів
template1 = environment.find_template("fact1")
template2 = environment.find_template("fact2")
# визначити факти
fact1 = template1.assert_fact(active_person=clips.Symbol("TRUE"))
fact2 = template2.assert_fact(love_skiing=clips.Symbol("TRUE"))
# друкувати початковий стан бази фактів
print("initial database states")
for fact in environment.facts():
   print(fact)
print()
# пряме виведення в режимі трасування та друк правил
print("forward inference")
for rule in environment.rules():
   rule.watch_activations = True
   rule.watch_firings = True
environment.run()
for rule in environment.rules():
   rule.matches(verbosity = clips.common.Verbosity.TERSE)
   print(rule)
print()
# друкувати заключний стан бази фактів
print("resulting database states")
for fact in environment.facts():
   print(fact)
```

Другий варіант – факти визначаються з використанням оператора deffacts

```
!pip install clipspy
import clips
# визначити шаблони фактів
DEFTEMPLATE_STRING1 = """
(deftemplate fact1
```

```
   "Fact1"
   (slot active_person (type SYMBOL)))
"""
DEFTEMPLATE_STRING2 = """
(deftemplate fact2
   "Fact2"
   (slot love_skiing (type SYMBOL)))
"""
DEFTEMPLATE_STRING3 = """
(deftemplate fact3
   "Fact3"
   (slot love_sea (type SYMBOL)))
"""
DEFTEMPLATE_STRING4 = """
(deftemplate fact4
   "Fact4"
   (slot vacation_winter (type SYMBOL)))
"""
DEFTEMPLATE_STRING5 = """
(deftemplate fact5
   "Fact5"
   (slot vacation_summer (type SYMBOL)))
"""
DEFTEMPLATE_STRING6 = """
(deftemplate fact6
   "Fact6"
   (slot goto_mountains (type SYMBOL)))
"""
# визначити правила
DEFRULE_STRING1 = """
(defrule rule1
   "Rule1"
   (fact4 (vacation_winter TRUE))
   (fact1 (active_person TRUE))
   =>
   (assert (fact6 (goto_mountains TRUE)))
   (println "go to mountains"))
"""
```

```
DEFRULE_STRING2 = """
(defrule rule2
   "Rule2"
   (fact3 (love_sea TRUE))
   =>
   (assert (fact5 (vacation_summer TRUE)))
   (println "summer vacation"))
"""
DEFRULE_STRING3 = """
(defrule rule3
   "Rule3"
   (fact2 (love_skiing TRUE))
   =>
   (assert (fact4 (vacation_winter TRUE)))
   (println "winter vacation"))
"""
# визначити факти
DEFFACT_STRING1 = """
(deffacts f1
   "Fact1"
   (fact1 (active_person TRUE)))
"""
DEFFACT_STRING2 = """
(deffacts f2
   "Fact2"
   (fact2 (love_skiing TRUE)))
"""
environment = clips.Environment()
# визначити шаблони фактів
environment.build(DEFTEMPLATE_STRING1)
environment.build(DEFTEMPLATE_STRING2)
environment.build(DEFTEMPLATE_STRING3)
environment.build(DEFTEMPLATE_STRING4)
environment.build(DEFTEMPLATE_STRING5)
environment.build(DEFTEMPLATE_STRING6)
# визначити правила
environment.build(DEFRULE_STRING1)
environment.build(DEFRULE_STRING2)
```

```
environment.build(DEFRULE_STRING3)
# визначити факти
environment.build(DEFFACT_STRING1)
environment.build(DEFFACT_STRING2)
# скинути середовище, щоб використовувати факти, які визначені deffacts
environment.reset()
# друкувати початковий стан бази фактів
print("initial database states")
for fact in environment.facts():
   print(fact)
print()
# пряме виведення в режимі трасування та друк правил
print("forward inference")
for rule in environment.rules():
   rule.watch_activations = True
   rule.watch_firings = True
environment.run()
for rule in environment.rules():
   rule.matches(verbosity = clips.common.Verbosity.TERSE)
   print(rule)
print()
# друкувати заключний стан бази фактів
print("resulting database states")
for fact in environment.facts():
   print(fact)
```

Для обох варіантів був отриманий такий результат

```
initial database states
(fact1 (active_person TRUE))
(fact2 (love_skiing TRUE))
forward inference
FIRE    1 rule3: f-2
==> Activation 0      rule1: f-3,f-1
winter vacation
FIRE    2 rule1: f-3,f-1
go to mountains
```

(defrule MAIN::rule1 "Rule1" (fact4 (vacation_winter TRUE)) (fact1 (active_person TRUE)) => (assert (fact6 (goto_mountains TRUE))) (println "go to mountains"))

(defrule MAIN::rule2 "Rule2" (fact3 (love_sea TRUE)) => (assert (fact5 (vacation_summer TRUE))) (println "summer vacation"))

(defrule MAIN::rule3 "Rule3" (fact2 (love_skiing TRUE)) => (assert (fact4 (vacation_winter TRUE))) (println "winter vacation"))

resulting database states
(fact1 (active_person TRUE))
(fact2 (love_skiing TRUE))
(fact4 (vacation_winter TRUE))
(fact6 (goto_mountains TRUE))

Внаслідок прямого логічного виведення на підставі фактів «людина – активна» і «любить кататися на лижах» в БД надходить факт «їхати вгору», який виступає як ціль.

Друга частина

Завдання: розробити експертну систему, що використовує пряме виведення з обмеженнями на підумови та застосуванням глобальних змінних, функцій, фактів та шаблонів фактів у вигляді багатомірних предикатів, продукційних правил.

Перший варіант – факти визначаються без використання оператора deffacts

```
!pip install clipspy
import clips
# визначити глобальні змінні
DEFGLOBAL_STRING1 = """
(defglobal ?*count* = 0)
"""
# визначити функції
DEFFUNC_STRING1 = """
(deffunction inc (?a)
   "Function1"
   (bind ?*count* (+ ?a 1)))
"""
# визначити шаблони фактів
DEFTEMPLATE_STRING1 = """
(deftemplate person
   "Fact1"
   (slot name (type STRING))
   (slot surname (type STRING))
   (slot experience (type INTEGER) (range 0 50) (default 0))
   (slot gender (allowed-symbols male female))
   (slot birthdate (type SYMBOL))
   (multislot phone (cardinality 1 2)))
"""
DEFTEMPLATE_STRING2 = """
(deftemplate person_accept
   "Fact2"
   (slot name (type STRING))
   (slot surname (type STRING))
   (slot accept (allowed-symbols TRUE FALSE)))
"""
```

```
    # визначити правила
    DEFRULE_STRING1 = """
    (defrule rule1
       "Rule1"
       (person (name ?name) (surname ?surname) (experience ?experience&:(>= ?experience 5)&:(<= ?experience 20)))
       =>
       (assert (person_accept (name ?name) (surname ?surname) (accept TRUE)))
       (inc ?*count*)
       (println ?name " " ?surname " is accepted")
       (println "count of accepted=" ?*count*))
    """
    environment = clips.Environment()
    # визначити глобальні змінні
    environment.build(DEFGLOBAL_STRING1)
    # визначити функції
    environment.build(DEFFUNC_STRING1)
    # визначити шаблоні фактів
    environment.build(DEFTEMPLATE_STRING1)
    environment.build(DEFTEMPLATE_STRING2)
    # визначити правила
    environment.build(DEFRULE_STRING1)
    # повернути шаблони фактів
    template = environment.find_template("person")
    # визначити факти
    fact = template.assert_fact(name="John", surname="Doe", experience=5, gender=clips.Symbol("male"), birthdate=clips.Symbol("01/01/1970"), phone=tuple(["111", "222"]))
    # друкувати функції
    print("functions")
    for function in environment.functions():
       print(function)
    print()
    # друкувати глобальні змінні
    print("globals")
    for g in environment.globals():
       print(g)
    print()
```

```
# друкувати початковий стан бази фактів
print("initial database states")
for fact in environment.facts():
   print(fact)
print()
# пряме виведення в режимі трасування та друк правил
print("forward inference")
for rule in environment.rules():
   rule.watch_activations = True
   rule.watch_firings = True
environment.run()
for rule in environment.rules():
   rule.matches(verbosity = clips.common.Verbosity.TERSE)
   print(rule)
print()
# друкувати заключний стан бази фактів
print("resulting database states")
for fact in environment.facts():
   print(fact)
```

Другий варіант – факти визначаються з використанням оператора deffacts

```
!pip install clipspy
import clips
# визначити глобальні змінні
DEFGLOBAL_STRING1 = """
(defglobal ?*count* = 0)
"""
# визначити функції
DEFFUNC_STRING1 = """
(deffunction inc (?a)
   "Function1"
   (bind ?*count* (+ ?a 1)))
"""
# визначити шаблони фактів
DEFTEMPLATE_STRING1 = """
(deftemplate person
   "Fact1"
   (slot name (type STRING))
```

```
    (slot surname (type STRING))
    (slot experience (type INTEGER) (range 0 50) (default 0))
    (slot gender (allowed-symbols male female))
    (slot birthdate (type SYMBOL))
    (multislot phone (cardinality 1 2)))
"""
DEFTEMPLATE_STRING2 = """
(deftemplate person_accept
    "Fact2"
    (slot name (type STRING))
    (slot surname (type STRING))
    (slot accept (allowed-symbols TRUE FALSE)))
"""
# визначити правила
DEFRULE_STRING1 = """
(defrule rule1
    "Rule1"
    (person (name ?name) (surname ?surname) (experience ?experience&:
(>= ?experience 5)&:(<= ?experience 20)))
    =>
    (assert (person_accept (name ?name) (surname ?surname) (accept TRUE)))
    (inc ?*count*)
    (println ?name " " ?surname " is accepted")
    (println "count of accepted=" ?*count*))
"""
# визначити факти
DEFFACT_STRING1 = """
(deffacts f1
    "Fact1"
    (person (name "John") (surname "Doe") (experience 5) (gender male)
(birthdate 01/01/1970) (phone "111" "222")))
"""
environment = clips.Environment()
# визначити глобальні змінні
environment.build(DEFGLOBAL_STRING1)
# визначити функції
environment.build(DEFFUNC_STRING1)
# визначити шаблоні фактів
```

```
environment.build(DEFTEMPLATE_STRING1)
environment.build(DEFTEMPLATE_STRING2)
# визначити правила
environment.build(DEFRULE_STRING1)
# визначити факти
environment.build(DEFFACT_STRING1)
# скинути середовище, щоб використовувати факти, які визначені deffacts
environment.reset()
# друкувати функції
print("functions")
for function in environment.functions():
    print(function)
print()
# друкувати глобальні змінні
print("globals")
for g in environment.globals():
    print(g)
print()
# друкувати початковий стан бази фактів
print("initial database states")
for fact in environment.facts():
    print(fact)
print()
# пряме виведення в режимі трасування та друк правил
print("forward inference")
for rule in environment.rules():
    rule.watch_activations = True
    rule.watch_firings = True
environment.run()
for rule in environment.rules():
    rule.matches(verbosity = clips.common.Verbosity.TERSE)
    print(rule)
print()
# друкувати заключний стан бази фактів
print("resulting database states")
for fact in environment.facts():
    print(fact)
```

Для обох варіантів був отриманий такий результат:

functions

(deffunction MAIN::inc (?a) "Function1" (bind ?*count* (+ ?a 1)))

globals

(defglobal MAIN ?*count* = 0)

initial database states

(person (name "John") (surname "Doe") (experience 5) (gender male) (birthdate 01/01/1970) (phone "111" "222"))

forward inference

FIRE 1 rule1: f-1

John Doe is accepted

count of accepted=1

(defrule MAIN::rule1 "Rule1" (person (name ?name) (surname ?surname) (experience ?experience&:(>= ?experience 5)&:(<= ?experience 20))) => (assert (person_accept (name ?name) (surname ?surname) (accept TRUE))) (inc ?*count*) (println ?name " " ?surname " is accepted") (println "count of accepted=" ?*count*))

resulting database states

(person (name "John") (surname "Doe") (experience 5) (gender male) (birthdate 01/01/1970) (phone "111" "222"))

(person_accept (name "John") (surname "Doe") (accept TRUE))

Внаслідок прямого логічного виведення на підставі факту «ім’я Джон, прізвище Доу, стаж 5 років, стать чоловіча, дата народження 01.01.1970, телефон 111 та 222» та обмеження «стаж не менше 5 та не більше 20» в БД надходить факт «ім’я Джон, прізвище Доу, прийнятий», який виступає як ціль та друкується як поточна кількість прийнятих на роботу – 1.

Лабораторна робота № 3

Тема: Вивчення основ побудови експертних систем з ненадійними знаннями (Додаток Г).

Перша частина

Завдання: розробити систему діагностики захворювання, засновану на алгоритмі Мамдані для нечіткого виведення.

1. Виконати генерацію універсумів для вхідних та вихідної змінних.
2. Створити функції належності.
3. Виконати візуалізацію функцій належності.
4. Виконати фазифікацію вхідних змінних.
5. Виконати агрегування підумов нечітких продукцій.
6. Виконати активізацію нечітких продукцій.
7. Виконати візуалізацію активізації нечітких продукцій.
8. Виконати агрегування заключень нечіткого виведення.
9. Виконати дефазифікацію.
10. Виконати візуалізацію агрегування заключень нечіткого виведення та дефазифікації.

```
!pip install scikit-fuzzy
import numpy, skfuzzy, matplotlib
x1_val = номер_варіанту # значення першої вхідної змінної
x2_val = номер варіанту # значення другої вхідної змінної

## Генерація універсумів
# генерація універсуму для першої вхідної змінної
x1 = numpy.arange(0, 26, 1)
# генерація універсуму для другої вхідної змінної
x2 = numpy.arange(0, 26, 1)
# генерація універсуму для вихідної змінної
y = numpy.arange(0, 10, 1)

## Створення функцій належності (дві сигмоїдні та одна Гауса) для трьох змінних
# створення функцій належності для першої вхідної змінної
x1mf_lo = skfuzzy.sigmf(x=x1, b=6, c=-0.8)
x1mf_md = skfuzzy.gaussmf(x=x1, mean=12.5, sigma=4)
```

```
x1mf_hi = skfuzzy.sigmf(x=x1, b=19, c=0.8)
# створення функцій належності для другої вхідної змінної
x2mf_lo = skfuzzy.sigmf(x=x2, b=6, c=-0.8)
x2mf_md = skfuzzy.gaussmf(x=x2, mean=12.5, sigma=4)
x2mf_hi = skfuzzy.sigmf(x=x2, b=19, c=0.8)
# створення функцій належності для вихідної змінної
ymf_lo = skfuzzy.sigmf(x=y, b=2, c=-2)
ymf_md = skfuzzy.gaussmf(x=y, mean=5, sigma=2)
ymf_hi = skfuzzy.sigmf(x=y, b=8, c=2)

## Візуалізація функцій належності
fig, (ax0, ax1, ax2) = matplotlib.pyplot.subplots(nrows=3, figsize=(8, 9))
# створення графіків функцій належності для першої вхідної змінної
ax0.plot(*(x1, x1mf_lo, "b"), linewidth=1.5, label="Low")
ax0.plot(*(x1, x1mf_md, "g"), linewidth=1.5, label="Middle")
ax0.plot(*(x1, x1mf_hi, "r"), linewidth=1.5, label="High")
ax0.set_title("Temperature")
ax0.legend()
# створення графіків функцій належності для другої вхідної змінної
ax1.plot(*(x2, x2mf_lo, "b"), linewidth=1.5, label="Low")
ax1.plot(*(x2, x2mf_md, "g"), linewidth=1.5, label="Middle")
ax1.plot(*(x2, x2mf_hi, "r"), linewidth=1.5, label="High")
ax1.set_title("Blood")
ax1.legend()
# створення графіків функцій належності для вихідної змінної
ax2.plot(*(y, ymf_lo, "b"), linewidth=1.5, label="Low")
ax2.plot(*(y, ymf_md, "g"), linewidth=1.5, label="Middle")
ax2.plot(*(y, ymf_hi, "r"), linewidth=1.5, label="High")
ax2.set_title("Influenza days")
ax2.legend()
matplotlib.pyplot.tight_layout()

## Фазифікація вхідних змінних
# фазифікація першої вхідної змінної
x1_lo = skfuzzy.interp_membership(x=x1, xmf=x1mf_lo, xx=x1_val)
x1_md=skfuzzy.interp_membership(x=x1,xmf=x1mf_md,xx=x1_val)
x1_hi = skfuzzy.interp_membership(x=x1, xmf=x1mf_hi, xx=x1_val)
# фазифікація другої вхідної змінної
```

```
x2_lo = skfuzzy.interp_membership(x=x2, xmf=x2mf_lo, xx=x2_val)
x2_md=skfuzzy.interp_membership(x=x2,xmf=x2mf_md,xx=x2_val)
x2_hi = skfuzzy.interp_membership(x=x2, xmf=x2mf_hi, xx=x2_val)

## Агрегування підумов нечітких продукцій за допомогою min
# агрегування підумов першої нечіткої продукції
cond_rule1 = numpy.fmin(x1_lo, x2_lo)
# cond_rule1 = numpy.multiply(x1_lo, x2_lo)
# агрегування підумов другої нечіткої продукції
cond_rule2 = numpy.fmin(x1_md, x2_md)
# cond_rule2 = numpy.multiply(x1_md, x2_md)
# агрегування підумов третьої нечіткої продукції
cond_rule3 = numpy.fmin(x1_hi, x2_hi)
# cond_rule3 = numpy.multiply(x1_level_hi, x2_hi)
## Обчислення заключень на основі  min
F1 = F2 = F3 = 1
# обчислення першого заключення
y_activation_lo = numpy.fmin(cond_rule1*F1, ymf_lo)
# y_activation_lo = numpy.multiply(cond_rule1*F1, ymf_lo)
# обчислення другого заключення
y_activation_md = numpy.fmin(cond_rule2*F2, ymf_md)
# y_activation_md = numpy.multiply(active_rule2*F2, ymf_md)
# обчислення третього заключення
y_activation_hi = numpy.fmin(cond_rule3*F3, ymf_hi)
# y_activation_hi = numpy.multiply(cond_rule3*F3, ymf_hi)

## Візуалізація обчислення заключень
y0 = numpy.zeros_like(y)
fig, ax0 = matplotlib.pyplot.subplots(figsize=(8, 3))
# створення графіка результату обчислення першого заключення
ax0.fill_between(x=y, y1=y0, y2=y_activation_lo, facecolor="b", alpha=0.7)
ax0.plot(*(y, ymf_lo, "b"), linewidth=0.5, linestyle="--", )
# створення графіка результату обчислення другого заключення
ax0.fill_between(x=y, y1=y0, y2=y_activation_md, facecolor="g", alpha=0.7)
ax0.plot(*(y, ymf_md, "g"), linewidth=0.5, linestyle="--")
# створення графіка результату обчислення третього заключення
ax0.fill_between(x=y, y1=y0, y2=y_activation_hi, facecolor="r", alpha=0.7)
ax0.plot(*(y, ymf_hi, "r"), linewidth=0.5, linestyle="--")
```

```
ax0.set_title("Consequent agregation")
matplotlib.pyplot.tight_layout()

## Агрегування заключень у вигляді max
aggregated = numpy.fmax(y_activation_lo, numpy.fmax(y_activation_md, y_activation_hi))
# aggregated = 1-numpy.multiply(1-y_activation_lo, numpy.multiply (1-y_activation_md, 1-y_activation_hi))

## Дефазифікація
result = skfuzzy.defuzz(x=y, mfx=aggregated, mode="centroid")

## Візуалізація агрегування заключень та дефазифікації
fig, ax0 = matplotlib.pyplot.subplots(figsize=(8, 3))
# створення графіка результату обчислення першого заключення
ax0.plot(*(y, ymf_lo, "b"), linewidth=0.5, linestyle="--")
# створення графіка результату обчислення другого заключення
ax0.plot(*(y, ymf_md, "g"), linewidth=0.5, linestyle="--")
# створення графіка результату обчислення третього заключення
ax0.plot(*(y, ymf_hi, "r"), linewidth=0.5, linestyle="--")
# створення графіка агрегування висновків нечіткого висновку
tip0 = numpy.zeros_like(y)
ax0.fill_between(x=y, y1=y0, y2=aggregated, facecolor="Orange", alpha=0.7)
# створення графіка дефазифікації
y_activation = skfuzzy.interp_membership(y, aggregated, result)
ax0.plot(*([result, result], [0, y_activation], "k"), linewidth=1.5, alpha=0.9)
ax0.set_title("Consequent agregation and defuzzification (line)")
matplotlib.pyplot.tight_layout()
matplotlib.pyplot.show()
print(result)
```

Був отриманий такий результат:

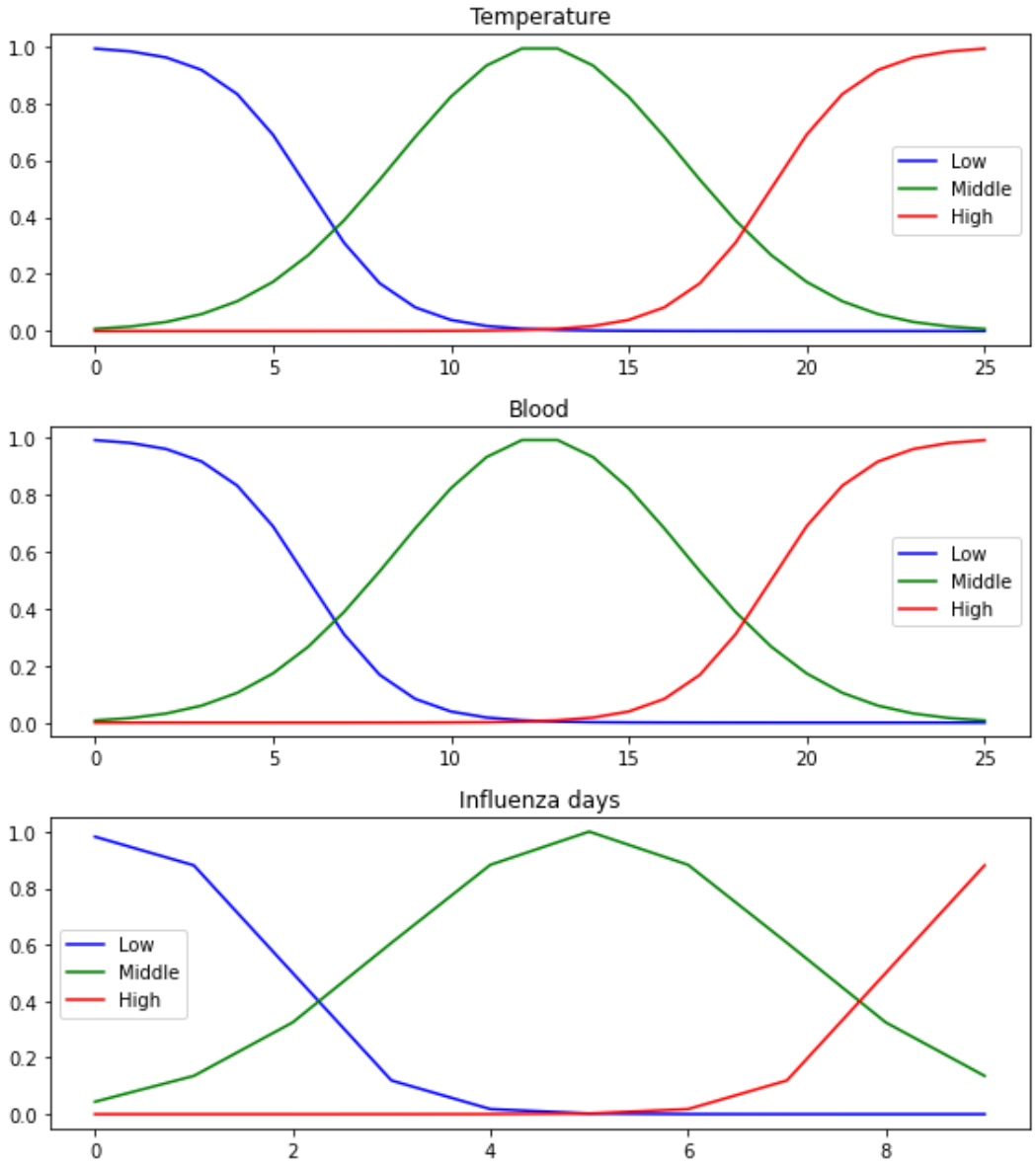


Рис. 1 – Функції належності вхідних і вихідної змінних

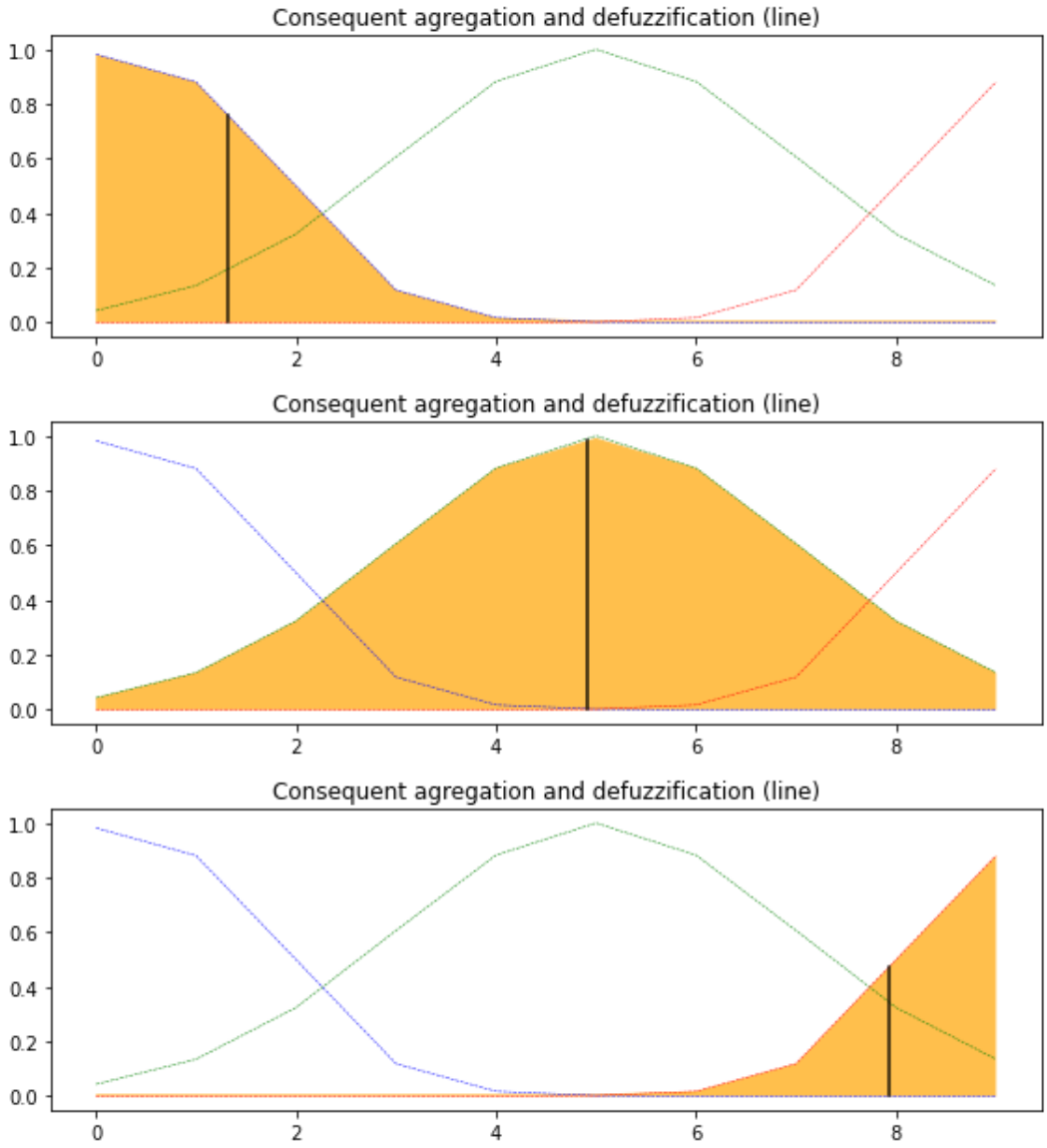


Рис. 2 – Агрегування заключень та дефазифікація (X1=0, X2=0; X1=12.5, X2=12.5; X1=25, X2=25)

Внаслідок нечіткого прямого логічного виведення на підставі чіткого значення першої вхідної змінної x1_val та чіткого значення другої вхідної змінної x2_val буде отримане чітке значення вихідної змінної result.

Друга частина

Завдання: розробити систему діагностики захворювання, засновану на механізмі виведення в ЕС PROSPECTOR (для виведення за ЕС PROSPECTOR застосування повинні бути відомі усі апріорні ймовірності, гіпотези повинні бути несумісні, а факти – незалежні, гарна теоретична база, з додаванням нової гіпотези, необхідно перераховувати всі ймовірності).

1. Вибрати захворювання H.

2. Визначити всю множину симптомів $\{E_j\}$, $j \in \{1,...,J\}$, пов'язаних із цим захворюванням. Кількість симптомів J не менше трьох.

3. Для кожного симптому E_j скласти таблицю виду.

Симптом E_j	**Захворювання** H	
	Є	**Відсутнє**
Ні (кількість людей) Є (кількість людей)	c_j a_j	d_j b_j
ВСЬОГО (кількість людей)	n_1	n_2

4. Визначити апріорні шанси на користь захворювання:

$$O(H) = \frac{n_1}{n_2},$$

де n_1 – кількість людей із захворюванням H;

n_2 – кількість людей без захворювання H.

5. Визначити правдоподібності для кожного симптому:

$$P(E_j \mid H) = \frac{a_j}{n_1},\ P(E_j \mid \overline{H}) = \frac{b_j}{n_2},\ P(\overline{E}_j \mid H) = \frac{c_j}{n_1},\ P(\overline{E}_j \mid \overline{H}) = \frac{d_j}{n_2},$$

де a_j – кількість людей із симптомом E_j із захворюванням H;

b_j – кількість людей із симптомом E_j без захворювання H;

c_j – кількість людей без симптому E_j із захворюванням H;

d_j – кількість людей без симптому E_j без захворювання H.

6. Визначити фактори достатності для кожного симптому:

$$D_{E_j} = \frac{P(E_j \mid H)}{P(E_j \mid \overline{H})},\ D_{E_j} \in [1,\infty).$$

7. Визначити фактори необхідності для кожного симптому:

$$N_{E_j} = \frac{P(\overline{E}_j \mid H)}{P(\overline{E}_j \mid \overline{H})},\ N_{E_j} \in [0,1].$$

8. Визначити апостеріорні шанси на користь захворювання H, за умови, що всі симптоми E_j істинні:

$$O\left(H \left| \bigcap_{j=1}^{J} E_j \right.\right) = O(H)\prod_{j=1}^{J} D_{E_j} \text{ або } O\left(H \left| \bigcap_{j=1}^{J} E_j \right.\right) = \min(D_{E_1},...,D_{E_J})O(H).$$

9. Визначити апостеріорні шанси на користь захворювання H, за умови, що всі симптоми E_j хибні:

$$O\left(H \left| \bigcap_{j=1}^{J} \overline{E}_j \right.\right) = O(H)\prod_{j=1}^{J} N_{E_j} \text{ або } O\left(H \left| \bigcap_{j=1}^{J} \overline{E}_j \right.\right) = \min(N_{E_1},...,N_{E_j})O(H).$$

Внаслідок прямого логічного виведення на підставі правдоподібності для кожного симптому та апріорних шансів на користь захворювання будуть визначені апостеріорні шанси на користь захворювання, за умови, що всі симптоми істинні, й апостеріорні шанси на користь захворювання, за умови, що всі симптоми хибні. За наявності кількох захворювань обирається те, у якого апостеріорні шанси на користь захворювання вищі.

Третя частина

Завдання: розробити систему діагностики захворювання, засновану на механізмі виведення в ЕС MYCIN (складність оцінки ступенів істинності та хибності фактів і заключень, відсутність теоретичного обґрунтування).

1. Вибрати захворювання H.

2. Визначити всю множину симптомів $\{E_j\}$, $j \in \{1,...,J\}$, пов'язаних із цим захворюванням. Кількість симптомів J не менше трьох.

3. Для кожного симптому E_j задати ступінь істинності $MB(E_j)$ та ступінь хибності $MD(E_j)$, де $0 \le MB(E_j), MD(E_j) \le 1$.

4. Для кожного симптому E_j обчислити коефіцієнт впевненості:

$$CF(E_j) = MB(E_j) - MD(E_j).$$

5. Сформувати із симптомів E_j антецедент A та обчислити його коефіцієнт впевненості $CF(A)$. Під час формування антецедента? повинні використовуватися логічні зв'язки І та АБО.

За умови логічного зв'язку І між фактами E_1 та E_2 коефіцієнт впевненості обчислюється у вигляді:

$$CF(A) = CF(E_1 \wedge E_2) = \min(CF(E_1), CF(E_2)).$$

За умови логічного зв'язку АБО між фактами E_1 та E_2 коефіцієнт впевненості обчислюється у вигляді:

$$CF(A) = CF(E_1 \vee E_2) = \max(CF(E_1), CF(E_2)).$$

6. Для правила R задати ступінь істинності $MB(R)$ та ступінь хибності $MD(R)$, де $0 \le MB(R), MD(R) \le 1$.

7. Для правила R обчислити коефіцієнт впевненості:

$$CF(R) = MB(R) - MD(R).$$

8. Обчислити коефіцієнт впевненості консеквента:

$$CF(B) = CF(A)CF(R).$$

Внаслідок прямого логічного виведення на підставі ступеня істинності та ступеня хибності для кожного симптому та ступеня істинності і ступеня хибності для правила буде визначений коефіцієнт впевненості консеквентна. У разі наявності кількох захворювань обирається те, у якого коефіцієнт впевненості консеквентна вищий.

Четверта частина

Завдання: розробити систему діагностики захворювання, засновану на механізмі виведення, який базується на теорії Демпстера–Шефера (дає інтервальні оцінки, необхідно задавати ступені довіри і правдоподібності).

Вибрати захворювання H.

1. Визначити всю множину симптомів $\Omega = \{E_1, ..., E_n\}$, пов'язаних із цим захворюванням. Кількість симптомів n не менше трьох.

2. Для всіх множин $C \subset \Omega$ задати базові ймовірності $m(C)$.

3. Для всіх множин $C \subset \Omega$ обчислити їх нижні ймовірності:

$$Bl(C) = \sum_{\hat{C} \subseteq C} m(\hat{C}).$$

4. Для всіх множин $C \subset \Omega$ обчислити їх верхні ймовірності:

$$Pl(C) = 1 - Bl(\overline{C}) = \sum_{\hat{C} \subseteq \overline{C}} m(\hat{C}).$$

5. Сформувати із симптомів E_j антецедент A та обчислити його нижню ймовірність $Bl(A)$. Під час формування антецедента повинні використовуватися логічні зв'язки І та АБО.

За умови логічного зв'язку І між фактами E_1 та E_2 нижня ймовірність обчислюється у вигляді:

$$Bl(E_1 \wedge E_2) = \min(Bl(E_1), Bl(E_2)).$$

За умови логічного зв'язку АБО між фактами E_1 та E_2 нижня ймовірність обчислюється у вигляді:

$$Bl(E_1 \vee E_2) = \max(Bl(E_1), Bl(E_2)).$$

6. Для правила R задати базову ймовірність $m(R)$.

7. Обчислити коефіцієнт впевненості консеквента:

$$m(B) = Bl(A) \cdot m(R).$$

Внаслідок прямого логічного виведення на підставі базових ймовірностей для елементів булеану множини симптомів та базової ймовірності для правила буде визначений коефіцієнт впевненості консеквентна. За наявності кількох захворювань обирається те, у якого коефіцієнт впевненості консеквентна вищий.

Лабораторна робота №4

Тема: Вивчення колаборативної фільтрації (Додаток Д).

Перша частина

Завдання: обчислити відсутню оцінку рейтингу на основі коефіцієнта кореляції Пірсона та скоригованої косинусної подоби.

Задано 5-бальну рейтингову матрицю $R = [r_{ui}]$ розмірності $m \times n$, де $m=n=5$, у вигляді такої таблиці:

Таблиця 1

Рейтингова таблиця

	Item1	**Item2**	**Item3**	**Item4**	**Item5**
User1	2	4	2	2	4
User2	1	3	3	5	1
User3	4	5	2	3	3
User4	1	1	5	2	1
User5	1	3	3	2	1

Елемент r_{ui}, рейтинг якого треба оцінити, визначається з наступної таблиці та відповідає номеру студента в журналі (номер варіанта).

Таблиця 2

Визначення елемента, який потрібно оцінити за номером варіанта

	Item1	**Item2**	**Item3**	**Item4**	**Item5**
User1	1	2	3	4	5
User2	6	7	8	9	10
User3	11	12	13	14	15
User4	16	17	18	19	20
User5	21	22	23	24	25

```
!pip install scikit-surprise
import numpy, surprise, pandas
number_variant=5 # номер варіанта 5
# Створення таблиці
df = pandas.DataFrame(data=[["user1", "item1", 2], ["user1", "item2", 4], ["user1", "item3", 2], ["user1", "item4", 2], ["user1", "item5", 4], ["user2", "item1", 1], ["user2", "item2", 3], ["user2", "item3", 3], ["user2", "item4", 5], ["user2", "item5", 1], ["user3", "item1", 4], ["user3", "item2", 5], ["user3", "item3", 2], ["user3", "item4", 3], ["user3", "item5", 3], ["user4", "item1", 1],
```

["user4", "item2", 1], ["user4", "item3", 5], ["user4", "item4", 2], ["user4", "item5", 1], ["user5", "item1", 1], ["user5", "item2", 3], ["user5", "item3", 3], ["user5", "item4", 2], ["user5", "item5", 1]], columns = ["user_id", "item_id", "rating"])

```
# ["user1", "item5", 4] => ["user1", "item5", 0] для number_variant=5
df.iloc[number_variant-1][2]=0
# Створення читача таблиці
reader = surprise.Reader(name=None, line_format="user item rating", sep=None, rating_scale=(1, 5), skip_lines=0)
# Створення навчального та тестового датасетів
Dataset = surprise.Dataset.load_from_df(df[["user_id", "item_id", "rating"]], reader)
train_Dataset, valid_Dataset = surprise.model_selection.split.train_test_split(data=Dataset, test_size=0.2, train_size=0.8, random_state=None, shuffle=True)
print("---------------------------------")
# Обчислення оцінки рейтингу на основі скоригованої косинусної подібності
model = surprise.prediction_algorithms.knns.KNNBasic(k=3, min_k=1, sim_options={"name":"cosine","user_based":False,"min_support":1}, verbose=False)
model.fit(trainset=train_Dataset)
test_pred = model.test(testset=valid_Dataset, verbose=False)
print(test_pred[number_variant-1])
# Обчислення RMSE
surprise.accuracy.rmse(predictions=test_pred, verbose=True)
print("---------------------------------")
# Обчислення оцінки рейтингу на основі коефіцієнта кореляції Пірсона
model = surprise.prediction_algorithms.knns.KNNWithMeans(k=3, min_k=1, sim_options={"name":"pearson","user_based":True,"min_support":1}, verbose=False)
model.fit(trainset=train_Dataset)
test_pred = model.test(testset=valid_Dataset, verbose=False)
print(test_pred[number_variant-1])
# Обчислення RMSE
surprise.accuracy.rmse(predictions=test_pred, verbose=True)
print("---------------------------------")
```

Був отриманий такий результат:

user: user1 item: item5 r_ui = 4.00 est = 3.36 {'actual_k': 3, 'was_impossible': False}

RMSE: 0.8774

---

user: user1 item: item5 r_ui = 4.00 est = 3.48 {'actual_k': 2, 'was_impossible': False}

RMSE: 0.4958

Внаслідок колаборативної фільтрації на підставі рейтингової матриці буде визначений відсутній рейтинг (у цьому прикладі для user1 та item5 він дорівнює 4).

Друга частина

Завдання: обчислити відсутню оцінку рейтингу з урахуванням SVD та NMF.

Задано 5-бальну рейтингову матрицю $R = [r_{ui}]$ розмірності $m \times n$, де $m=n=5$, у вигляді такої таблиці:

Таблиця 3

Рейтингова таблиця

	Item1	**Item2**	**Item3**	**Item4**	**Item5**
User1	2	4	2	2	4
User2	1	3	3	5	1
User3	4	5	2	3	3
User4	1	1	5	2	1
User5	1	3	3	2	1

Елемент r_{ui}, рейтинг якого треба оцінити, визначається з наступної таблиці та відповідає номеру студента в журналі (номер варіанта).

Таблиця 4

Визначення елемента, який потрібно оцінити за номером варіанта

	Item1	**Item2**	**Item3**	**Item4**	**Item5**
User1	1	2	3	4	5
User2	6	7	8	9	10
User3	11	12	13	14	15
User4	16	17	18	19	20
User5	21	22	23	24	25

```
!pip install scikit-surprise
import numpy, surprise, pandas
number_variant=5 # номер варіанта 5
# Створення таблиці
df = pandas.DataFrame(data=[["user1", "item1", 2], ["user1", "item2", 4], ["user1", "item3", 2], ["user1", "item4", 2], ["user1", "item5", 4], ["user2", "item1", 1], ["user2", "item2", 3], ["user2", "item3", 3], ["user2", "item4", 5], ["user2", "item5", 1], ["user3", "item1", 4], ["user3", "item2", 5], ["user3", "item3", 2], ["user3", "item4", 3], ["user3", "item5", 3], ["user4", "item1", 1], ["user4", "item2", 1], ["user4", "item3", 5], ["user4", "item4", 2], ["user4", "item5", 1], ["user5", "item1", 1], ["user5", "item2", 3], ["user5", "item3", 3], ["user5", "item4", 2], ["user5", "item5", 1]], columns = ["user_id", "item_id", "rating"])
# ["user1", "item5", 4] => ["user1", "item5", 0] для number_variant=5
df.iloc[number_variant-1][2]=0
```

```
# Створення читача таблиці
reader = surprise.Reader(name=None, line_format="user item rating", sep=None, rating_scale=(1, 5), skip_lines=0)
# Створення навчального та тестового датасетів
Dataset = surprise.Dataset.load_from_df(df[["user_id", "item_id", "rating"]], reader)
train_Dataset, valid_Dataset = surprise.model_selection.split.train_test_split (data=Dataset, test_size=0.2, train_size=0.8, random_state=None, shuffle=True)
print("---------------------------------")
# Обчислення оцінки рейтингу на основі SVD
model = surprise.prediction_algorithms.matrix_factorization.SVD (n_factors=3, n_epochs=20, biased=False, init_mean=0, init_std_dev=0.1, lr_all=0.005, random_state=0, verbose=False)
model.fit(trainset=train_Dataset)
test_pred = model.test(testset=valid_Dataset, verbose=False)
print(test_pred[number_variant-1])
# Обчислення RMSE
surprise.accuracy.rmse(predictions=test_pred, verbose=True)
print("---------------------------------")
# Обчислення оцінки рейтингу на основі NMF
model = surprise.prediction_algorithms.matrix_factorization.NMF( n_factors=3, n_epochs=20, biased=False, random_state=0, init_low=0, init_high=1, verbose=False)
model.fit(trainset=train_Dataset)
test_pred = model.test(testset=valid_Dataset, verbose=False)
print(test_pred[number_variant-1])
# Обчислення RMSE
surprise.accuracy.rmse(predictions=test_pred, verbose=True)
print("---------------------------------")
```

Був отриманий такий результат:

```
user: user1      item: item5      r_ui = 4.00   est = 1.02   {'was_impossible': False}
RMSE: 1.8891
---------------------------------
user: user1      item: item5      r_ui = 4.00   est = 5.00   {'was_impossible': False}
RMSE: 1.1468
```

Внаслідок колаборативної фільтрації на підставі рейтингової матриці буде визначений відсутній рейтинг (у цьому прикладі для user1 та item5 він дорівнює 4).

ЛІТЕРАТУРА

1. Технології підтримання прийняття рішень: навч. посіб. / за заг. ред. В. В. Пасічника. Львів: Вид-во Львівської політехніки, 2010. 252 с.

2. Месюра В. І., Яровий А. А., Арсенюк І. Р. Експертні системи. Частина 1: навч. посіб. Вінниця: ВНТУ, 2006. 114 с.

3. Бакан Г. М. Вступ до теорії експертних систем та баз знань. Київ: ВПЦ «Київський університет», 2005. 90 с.

4. Гнатієнко Г. М., Снитюк В. Є. Експертні технології прийняття рішень: монографія. Київ: ТОВ «Маклаут», 2008. 444 с.

5. Литвин В. В., Пасічник В. В., Яцишин Ю. В. Інтелектуальні системи: підручник. Львів: «Новий світ-2000», 2008. 406 с.

6. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень: навч. посіб. Запоріжжя: ЗНТУ, 2008. 341 с.

7. Кацадзе Т. Л. Експертні системи прийняття рішень в енергетиці: навч. посіб. Київ: ЛОГОС, 2014. 173 с.

8. Системи штучного інтелекту: навч. посіб. за наук. ред. В. В. Пасічника; МОНМС України. 2-ге вид., виправ. та доп. Львів: Магнолія, 2006, 2013. 279 с.

9. Expert Systems / edited by Petrica Vizureanu. Vukovar, Croatia: Intech, 2010. 238 p.

10. Siler W., Buckley J. J., Fuzzy Expert Systems and Fuzzy Reasoning. New Jersey: John Wiley & Sons, 2005. 405 p.

11. Gupta I., Nagpal G. Artificial Intelligence and Expert Systems. Dulles, Virginia: Mercury Learning and Information LLC, 2020. 409 p.

12. Aggarwal C. C. Recommender Systems: The Textbook. Cham: Springer, 2016. 498 p.

13. Falk K. Practical Recommender Systems. Shelter Island, New York: Manning Publications Co., 2019. 406 p.

ДОДАТКИ

ДОДАТОК А
ВІДНОШЕННЯ І ПРАВИЛА ДЛЯ ПСЕВДОФІЗИЧНОЇ ЛОГІКИ

1. Псевдофізична логіка точкових часових відношень:

а) нечітка логіка точкових часових відношень

R_0^1 – подія p відбувається одночасно з подією q,

R_1^1 – подія p відбувається раніше події q,

R_2^1 – подія p відбувається пізніше події q,

$R_{3,F}^1(n,\omega)$ – подія p відбувається раніше події q приблизно на $n\omega$ одиниць за шкалою L, де ω – одиниця вимірювання на шкалі L, $n=1,2,3\ldots$,

$R_{4,F}^1(t^*)$ – подія p реалізується приблизно в момент t^* на шкалі L,

$R_{5,F}^1(\tau)$ – подія p реалізується приблизно з періодом τ на шкалі L;

б) метрична логіка точкових часових відношень

$R_{3,M}^1(n,\omega)$ – подія p відбувається раніше події q на $n\omega$ одиниць за шкалою L, де ω – одиниця вимірювання на шкалі L, $n=1,2,3\ldots$,

$R_{4,M}^1(t^*)$ – подія p реалізується в момент t^* на шкалі L,

$R_{5,M}^1(\tau)$ – подія p реалізується з періодом τ на шкалі L.

2. Псевдофізична логіка інтервальних часових відношень:

а) нечітка логіка інтервальних часових відношень

R_0^2 – подія p збігається з подією q,

R_1^2 – подія p строго передує q,

R_2^2 – подія p суворо слідує за q,

R_3^2 – подія p перетинається з q,

R_4^2 – подія p лежить усередині q,

R_5^2 – подія p лежить усередині q так, що їх початки збігаються,

R_6^2 – подія p лежить усередині q так, що їх кінці збігаються,

R_7^2 – подія p безпосередньо передує q,

$R^2_{8,F}(n,\omega)$ – подія p строго передує q приблизно на $n\omega$ одиниць за шкалою L,

$R^2_{9,F}(n,\omega)$ – подія p перетинається q так, що відстань між їх початками приблизно дорівнює $n\omega$ за шкалою L,

$R^2_{10,F}(n,\omega)$ – подія p перетинається q так, що відстань між їх кінцями приблизно дорівнює $n\omega$ за шкалою L,

$R^2_{11,F}(t^*,\Delta t)$ – початок події p реалізується приблизно в момент t^*, а тривалість p має величину приблизно Δt за шкалою L,

$R^2_{12,F}(\tau)$ – подія p реалізується приблизно з періодом τ на шкалі L;

б) метрична логіка інтервальних часових відношень

$R^2_{8,M}(n,\omega)$ – подія p строго передує q на $n\omega$ одиниць за шкалою L,

$R^2_{9,M}(n,\omega)$ – подія p перетинається з q так, що відстань між їх початками дорівнює $n\omega$ за шкалою L,

$R^2_{10,M}(n,\omega)$ – подія p перетинається з q так, що відстань між їх кінцями дорівнює $n\omega$ за шкалою L,

$R^2_{11,M}(t^*,\Delta t)$ – початок події p реалізується в даний момент t^*, а тривалість p має величину Δt за шкалою L,

$R^2_{12,M}(\tau)$ – подія p реалізується з періодом τ на шкалі L.

Правила виводу

Тип 1

$p_i R^1_0 p_j \to p_j R^1_0 p_i$,

$p_i R p_j, p_j R p_k \to p_i R p_k, R \in \{R^1_0, R^1_1, R^1_2\}$,

$p_i R^2_0 p_j, p_j R p_k \to p_i R p_k, R \in \{R^2_1, R^2_2\}$.

Тип 2

$p_i R^1_3(n,\omega) p_j, p_j R^1_3(m,\omega) p_k \to p_i R^1_3(n+m,\omega) p_k$,

$$p_i R^1_3(n,\omega) p_j, p_i R^1_3(m,\omega) p_k \to \begin{cases} p_k R^1_3(n-m,\omega) p_j, & m < n \\ p_j R^1_3(m-n,\omega) p_k, & m \geq n \end{cases}.$$

Тип 3

$p_i R_3^1(n,\omega) p_j \to p_i R_1^1 p_j$,

$$p_i R_3^1(n,\omega) p_j, p_i R_3^1(m,\omega) p_k \to \begin{cases} p_k R_1^1 p_j & m < n \\ p_k R_0^1 p_j & m = n \\ p_k R_2^1 p_j & m > n \end{cases}.$$

$$p_i R_4^1(t_1), p_j R_4^1(t_2) \to \begin{cases} p_i R_1^1 p_j & t_1 < t_2 \\ p_i R_0^1 p_j & t_1 = t_2 \\ p_i R_2^1 p_j & t_1 > t_2 \end{cases}$$, де t_i - мітки шкали години,

$p_i R_0^1 p_j, p_j R_4^1(t_1) \to p_i R_4^1(t_1)$,

$p_i R_4^1(t_1), p_i R_3^1(n,\omega) p_j \to p_i R_4^1(t_2),\ t_2 = t_1 \oplus N, N = n\omega$,

$p_i R_5^1(\tau), p_j R_4^1(t) \to p R_0^1 p_i,\ \tau = n\omega, t = kn\omega$.

Тип 4

$p_i R_0^2 p_j \to p_j R_0^2 p_i;\ p_i R_1^2 p_j \to p_j R_2^2 p_i;\ p_i R_2^2 p_j \to p_j R_1^2 p_i$,

$p_i R p_j \to p_i R_3^2 p_j,\ R \in \{R_4^2, R_5^2, R_6^2\};\ p_i R_7^2 p_j \to p_i R_1^2 p_j$,

$p_i R p_j, p_j R p_k \to p_i R p_k,\ R \in \{R_0^2, R_1^2, R_2^2\}$,

$p_i R_0 p_j, p_j R p_k \to p_i R p_k,\ R \in \{R_1^2, ..., R_7^2\}$,

$p_i R_1^2 p_j, p_k R p_j \to p_i R_1^2 p_k,\ R \in \{R_4^2, R_5^2, R_6^2\}$,

$p_i R_2^2 p_j, p_k R p_j \to p_i R_2^2 p_k,\ R \in \{R_4^2, R_5^2, R_6^2\}$,

$p_i R_3^2 p_j, p_j R p_k \to p_i R_3^2 p_k,\ R \in \{R_4^2, R_5^2, R_6^2\}$,

$p_i R_3^2 p_j, p_i R p_k \to p_j R_3^2 p_k,\ R \in \{R_4^2, R_5^2, R_6^2\}$,

$p_i R_4^2 p_j, p_j R_5^2 p_k \to p_i R_4^2 p_k;\ p_i R_4^2 p_j, p_i R_5^2 p_k \to p_j R_3^2 p_k$,

$p_i R_4^2 p_j, p_k R p_i \to p_k R_4^2 p_j,\ R \in \{R_5^2, R_6^2\}$,

$p_i R_4^2 p_j, p_j R_6^2 p_k \to p_i R_4^2 p_k;\ p_i R_4^2 p_j, p_i R_6^2 p_k \to p_j R_3^2 p_k$,

$p_i R p_j, p_k R p_i \to p_k R p_j,\ R \in \{R_4^2, R_5^2, R_6^2\}$,

$p_i R p_j, p_i R p_k \to p_j R_3^2 p_k,\ R \in \{R_4, R_5, R_6\}$,

$p_i R p_j, p_j R p_k \to p_i R p_k,\ R \in \{R_4^2, R_5^2, R_6^2\}$,

$p_i R_5^2 p_j, p_j R_6^2 p_k \to p_i R_4^2 p_k;\ p_i R_5^2 p_j, p_i R_6^2 p_k \to p_j R_3^2 p_k$,

$p_i R_5^2 p_j, p_k R_6^2 p_j \to p_i R_3^2 p_k$,

$p_i R_7^2 p_j, p_i R_3^2 p_j \to p_i R_4^2 p_k;\ p_j R_1^2 p_k, p_j R_7^2 p_k \to p_i R_1^2 p_k,$

$p_i R p_j, p_j R_7^2 p_k \to p_i R p_k,\ R \in \{R_4^2, R_5^2, R_6^2\},$

$p_i R_7^2 p_j, p_j R_7^2 p_k \to p_i R_1^2 p_k;\ p_i R_7^2 p_j, p_i R_7^2 p_k \to p_j R_3^2 p_k.$

Тип 5

$p_i R_8^2(n,\omega) p_j, p_j R_8^2(m,\omega) p_k \to p_i R_1^2 p_k,$

$p_i R_9^2(n,\omega) p_j, p_j R_9^2(m,\omega) p_k \to \mu_н(p_i) R_3^1(n+m,\omega) \mu_н(p_k),$

$p_i R_{10}^2(n,\omega) p_j, p_j R_{10}^2(m,\omega) p_k \to \mu_к(p_i) R_3^1(n+m,\omega) \mu_к(p_k),$

$$p_i R_9^2(n,\omega) p_j, p_i R_9^2(m,\omega) p_k \to \begin{cases} \mu_н(p_k) R_3^1(n-m,\omega) \mu_н(p_j), & m \le n \\ \mu_н(p_j) R_3^1(m-n,\omega) \mu_н(p_k), & m > n \end{cases};$$

$$p_i R_{10}^2(n,\omega) p_j, p_i R_{10}^2(m,\omega) p_k \to \begin{cases} \mu_к(p_k) R_3^1(n-m,\omega) \mu_к(p_j), & m \le n \\ \mu_к(p_j) R_3^1(m-n,\omega) \mu_к(p_k), & m > n \end{cases};$$

$$p_i R_{11}^2(t_1,\Delta t_1), p_j R_{11}^2(t_2,\Delta t_2), \to \begin{cases} p_i R_5^2 p_j, & t_1 = t_2, \Delta t_1 < \Delta t_2 \\ p_j R_5^2 p_i, & t_1 = t_2, \Delta t_1 > \Delta t_2 \\ p_i R_0^2 p_j, & t_1 = t_2, \Delta t_1 = \Delta t_2 \end{cases};$$

$$p_i R_{11}^2(t_1,\Delta t_1), p_j R_{11}^2(t_2,\Delta t_2), \to \begin{cases} \mu_н(p_i) R_1^1 \mu_н(p_j), & t_1 < t_2 \\ \mu_н(p_j) R_1^1 \mu_н(p_i), & t_1 > t_2 \\ p_i R_1^2 p_j, & t_1 + \Delta t_1 < t_2 \\ p_j R_3^2 p_i, & t_1 + \Delta t_1 > t_2 \\ p_i R_7^2 p_j, & t_1 + \Delta t_1 = t_2 \end{cases};$$

$p_i R_8^2(n,\omega) p_j, p_j R_9^2(m,\omega) p_k \to p_i R_8^2(n+m,\omega) p_k.$

Тип 6

$p_i R_8^2(0,\omega) p_j \to p_i R_7^2 p_j;\ p_i R_7^2 p_j \to p_i R_8^2(0,\omega) p_j,$

$p_i R_8^2(n,\omega) p_j \to p_i R_1^2 p_j,$

$p_i R_0^2 p_j, p_j R_{11}^2(t,\Delta t) \to p_j R_{11}^2(t,\Delta t);\ p_i R_{11}^2(t,\Delta t), p_j R_{11}^2(t,\Delta t) \to p_i R_0^2 p_j,$

$p_i R_8^2(n,\omega) p_j, p_j R_9^2(m,\omega) p_k \to p_i R_1^2 p_k,$

$$p_i R_8^2(n,\omega) p_j, p_i R_{10}^2(m,\omega) p_k \to \begin{cases} p_k R_1^2 p_j, & m \le n \\ p_k R_7^2 p_j, & m = n \\ p_k R_3^2 p_j, & m < n \end{cases};$$

$p_i R_8^2(n,\omega) p_j \to p_i R_1^2 p_j$; $p_i R_9^2(n,\omega) p_j \to p_i R_3^2 p_j$;

$p_i R_{10}^2(n,\omega) p_j \to p_i R_3^2 p_j$,

$p_i R_{12}^2(\tau), p_j R_{11}^2(t,\Delta t) \to p_i R_3^2 p_j$, $\tau = n\omega, t = kn\omega, k > 0, n > 0$.

Тип 7

$p_1 R_1^2 p_2 \leftrightarrow (\mu_н(p_1) R_1^1 \mu_н(p_2)) \wedge (\mu_к(p_1) R_1^1 \mu_н(p_2)) \wedge (\mu_к(p_1) R_1^1 \mu_к(p_2))$,

$p_1 R_2^2 p_2 \leftrightarrow (\mu_н(p_1) R_2^1 \mu_н(p_2)) \wedge (\mu_н(p_1) R_2^1 \mu_к(p_2)) \wedge (\mu_к(p_2) R_1^1 \mu_к(p_1))$,

$p_1 R_0^2 p_2 \leftrightarrow (\mu_н(p_1) R_0^1 \mu_н(p_2)) \wedge (\mu_к(p_1) R_0^1 \mu_к(p_2))$,

$p_1 R_4^2 p_2 \leftrightarrow (\mu_н(p_2) R_1^1 \mu_н(p_1)) \wedge (\mu_к(p_2) R_1^1 \mu_к(p_1))$,

$p_1 R_5^2 p_2 \leftrightarrow (\mu_н(p_1) R_0^1 \mu_н(p_2)) \wedge (\mu_к(p_1) R_1^1 \mu_к(p_2))$,

$p_1 R_6^2 p_2 \leftrightarrow (\mu_н(p_2) R_1^1 \mu_н(p_1)) \wedge (\mu_к(p_2) R_0^1 \mu_к(p_1))$,

$p_1 R_7^2 p_2 \leftrightarrow (\mu_к(p_1) R_0^1 \mu_н(p_2)) \wedge (\mu_н(p_1) R_1^1 \mu_н(p_2))$,

$p \to (\mu_н(p) R_1^1 \mu_к(p))$,

$p_1 R_9^2(n,\omega) p_2 \leftrightarrow \mu_к(p_1) R_3^1(n,\omega) \mu_н(p_2)$,

$p_1 R_9^2(n,\omega) p_2 \leftrightarrow \mu_н(p_1) R_3^1(n,\omega) \mu_н(p_2)$,

$p_1 R_{10}^2(n,\omega) p_2 \leftrightarrow \mu_к(p_1) R_3^1(n,\omega) \mu_к(p_2)$,

$p_1 R_{11}^2(t,\Delta t) \leftrightarrow (\mu_н(p_1) R_4^1(t)) \wedge (\mu_к(p_1) R_4^1(t+\Delta t))$,

$\mu_к(p_1) R_3^1(n,\omega) \mu_н(p_2) \to p_1 R_1^2 p_2$.

3. Псевдофізична логіка відстаней:

а) просторові відношення для нечіткої логіки відстаней

R_1^3 – впритул,

R_2^3 – дуже близько,

R_3^3 – близько,

R_4^3 – не далеко і не близько,

R_5^3 – далеко,

R_6^3 – дуже далеко,

R_7^3 – дуже-дуже далеко,

$R_{8,F}^3$ – приблизно в n одиницях;

б) просторові відношення для метричної логіки відстаней

$R^3_{8,M}$ – в n одиницях.

У логіці відстаней враховується також розмір об’єкта:

$r^*(x_i)$ – має розмір,

де

x_1 – нульовий,

x_2 – дуже маленький,

x_3 – маленький,

x_4 – середній,

x_5 – великий,

x_6 – дуже великий,

x_7 – дуже-дуже великий.

4. Псевдофізична логіка напрямків

а) просторові відношення для нечіткої логіки напрямків:

R^4_1 – справа,

R^4_2 – попереду-праворуч,

R^4_3 – вперед,

R^4_4 – попереду-зліва,

R^4_5 – ліворуч,

R^4_6 – ззаду-зліва,

R^4_7 – ззаду,

R^4_8 – ззаду-праворуч,

$R^4_{9,F}$ – приблизно під кутом A;

б) просторові відношення для метричної логіки напрямків:

$R^4_{9,M}$ – під кутом A.

5. Псевдофізична логіка взаємного розташування об'єктів

а) просторові відношення для нечіткої логіки взаємного розташування об'єктів:

$R^5_{0,F}$ – перебувати приблизно під кутом $A(O_1R^4_{9,F}O_2 \to O_1R^5_{0,F}O_2)$,

R^5_1 – мати горизонтальне розташування,

R^5_2 – мати вертикальне положення,

R^5_3 – перебувати всередині,

R^5_4 – перебувати поза,

R^5_5 – перебувати на поверхні,

R^5_6 – перебувати в центрі,

R^5_7 – перебувати в середині,

R^5_8 – бути там же, де і,

R^5_9 – бути ненульовою проєкцією,

R^5_{10} – перебувати пооколиці ε,

R^5_{11} – бути частиною цілого,

R^5_{12} – перебувати на одній прямій,

R^5_{13} – перебувати навколо,

R^5_{14} – бути на краю,

R^5_{15} – бути паралельно,

R^5_{16} – бути перпендикулярно,

R^5_{17} – бути симетричним,

R^5_{18} – перебувати приблизно в n одиницях від $O_1R^3_{8,F}O_2 \to O_1R^5_{18,F}O_2$,

R^5_{19} – мати точку опори на,

R^5_{20} – мати точку підвісу на,

R^5_{21} – стикатися,

R^5_{22} – знаходитися вище,

R^5_{23} – знаходитися нижче,

R^5_{24} – перебувати на однаковому рівні,

R^5_{25} – бути далі,

R_{26}^5 – бути ближче,

R_{27}^5 – бути рівновіддаленим,

R_{28}^5 – бути між,

R_{29}^5 – не стикатися $O_1\overline{R}_{21}^5O_2 \to O_1R_{29}^5O_2$,

R_{30}^5 – не симетрично $O_1\overline{R}_{17}^5O_2 \to O_1R_{30}^5O_2$,

R_{31}^5 – не перебувати на $O_1\overline{R}_5^5O_2 \to O_1R_{31}^5O_2$,

R_{32}^5 – перебувати над $O_1R_{22}^5O_2 \wedge O_1R_{29}^5O_2 \wedge O_1R_9^5O_2 \to O_1R_{32}^5O_2$,

R_{33}^5 – перебувати під $O_1R_{23}^5O_2 \wedge (O_1R_{21}^5O_2 \vee O_1R_{29}^5O_2) \wedge O_1R_9^5O_2 \to O_1R_{33}^5O_2$,

R_{34}^5 – перебувати в $O_1R_3^5O_2 \to O_1R_{34}^5O_2$,

R_{35}^5 – перебувати ззовні $O_1R_4^5O_2 \to O_1R_{35}^5O_2$,

R_{36}^5 – перебувати у $O_1R_{10}^5O_2 \to O_1R_{36}^5O_2$,

R_{37}^5 – перебувати позаду $O_1R_7^4O_2 \to O_1R_{37}^5O_2$,

R_{38}^5 – бути спереду $O_1R_3^4O_2 \to O_1R_{38}^5O_2$,

R_{39}^5 – перебувати за $O_1R_{37}^5O_2 \wedge O_1R_{10}^5O_2 \wedge (O_1R_{21}^5O_2 \vee O_1R_{29}^5O_2) \to O_1R_{39}^5O_2$,

R_{40}^5 – перебувати перед $O_1R_{38}^5O_2 \wedge O_1R_{10}^5O_2 \wedge (O_1R_{21}^5O_2 \vee O_1R_{29}^5O_2) \to O_1R_{40}^5O_2$,

R_{41}^5 – знаходитися збоку,

$(O_1R_1^4O_2 \vee O_1R_5^4O_2) \wedge O_1R_{10}^5O_2 \wedge (O_1R_{21}^5O_2 \vee O_1R_{29}^5O_2) \to O_1R_{41}^5O_2$,

R_{42}^5 – бути поруч $O_1R_{41}^5O_2 \to O_1R_{42}^5O_2$,

R_{43}^5 – бути біля $O_1R_{10}^5O_2 \to O_1R_{43}^5O_2$,

R_{44}^5 – бути разом $O_1R_8^5O_2 \to O_1R_{44}^5O_2$,

R_{45}^5 – бути елементом рядка $\underset{n}{\wedge}(O_{n-1}R_{40}^5O_n \vee O_nR_{39}^5O_{n-1}) \wedge O_nR_{12}^5O_{n-1} \to O_nR_{45}^5$, $n \geq 2$,

R_{46}^5 – бути вздовж $O_2(d_{\max})R_{15}^5O_1 \wedge (O_1R_{21}^5O_2 \vee O_1R_{29}^5O_2) \to O_1R_{46}^5O_2$,

R_{47}^5 – бути впоперек $O_2(d_{\min})R_{16}^5O_1 \wedge (O_1R_{21}^5O_2 \vee O_1R_{29}^5O_2) \to O_1R_{47}^5O_2$,

R_{48}^5 – бути навпроти,

$O_1R_{38}^5O_2(orientation) \wedge (O_1R_{21}^5O_2 \vee O_1R_{29}^5O_2) \to O_1R_{48}^5O_2$,

де $orientation$ = передня сторона,

R_{49}^5 – виснути $O_1R_2^5 \wedge O_1R_{20}^5O_2 \to O_1R_{49}^5O_2$,

R_{50}^5 – стояти $O_1R_2^5 \wedge O_1R_{19}^5O_2 \to O_1R_{50}^5O_2$,

R_{51}^5 – лежати $O_1R_1^5 \wedge O_1R_{19}^5O_2 \to O_1R_{51}^5O_2$,

R^5_{52} – бути присунутим $O_1R^5_{21}O_2 \to O_1R^5_{52}O_2$,

R^5_{53} – бути відсунутим $O_1R^5_{10}O_2 \to O_1R^5_{53}O_2$,

R^5_{54} – перебувати справа $O_1R^4_1O_2 \to O_1R^5_{54}O_2$,

R^5_{55} – перебувати зліва $O_1R^4_5O_2 \to O_1R^5_{55}O_2$,

R^5_{56} – перебувати по один бік,

$(O_1R^4_1H \wedge O_2R^4_1H) \vee (O_1R^4_5H \wedge O_2R^4_5H) \to O_1R^5_{56}O_2$,

де H – точка відліку,

R^5_{57} – перебувати по різні боки,

$(O_1R^4_1H \wedge O_2R^4_5H) \vee (O_1R^4_5H \wedge O_2R^4_1H) \to O_1R^5_{57}O_2$;

б) просторові відношення для метричної логіки взаємного розташування об'єктів:

$R^5_{0,M}$ – перебувати під кутом $A(O_1R^4_{9,M}O_2 \to O_1R^5_{0,M}O_2)$,

$R^5_{18,M}$ – перебувати в *n* одиницях від $(O_1R^3_{8,M}O_2 \to O_1R^5_{18,M}O_2)$.

Правила виводу

Тип 1

$O_1R^5_iO_2 \to O_1R^5_jO_2$;

$O_1R^5_{40}O_2 \to O_1R^5_{26}O_2$; $O_1R^5_{40}O_2 \to O_1R^5_{38}O_2$; $O_1R^5_{31}O_2 \to O_1R^5_{22}O_2$;

$O_1R^5_{32}O_2 \to O_1R^5_{22}O_2$; $O_1R^5_{31}O_2 \to O_1R^5_{21}O_2$; $O_1R^5_{47}O_2 \to O_1R^5_{21}O_2$;

$O_1R^5_{15}O_2 \to O_1R^5_{29}O_2$; $O_1R^5_{32}O_2 \to O_1R^5_{29}O_2$; $O_1R^5_{33}O_2 \to O_1R^5_{23}O_2$;

$O_1R^5_{35}O_2 \to O_1R^5_4O_2$; $O_1R^5_{37}O_2 \to O_1R^5_{25}O_2$; $O_1R^5_{34}O_2 \to O_1R^5_3O_2$;

$O_1R^5_{36}O_2 \to O_1R^5_{10}O_2$; $O_1R^5_{39}O_2 \to O_1R^5_{37}O_2$; $O_1R^5_{51}O_2 \to O_1R^5_1O_2$;

$O_1R^5_{51}O_2 \to O_1R^5_{21}O_2$;

$O_1RO_2 \to O_1R^5_{34}O_2$, $R \in \{R^5_3, R^5_6, R^5_7\}$; $O_1RO_2 \to O_1R^5_{10}O_2$, $R \in \{R^5_{42}, R^5_{43}, R^5_{53}\}$;

$O_1R^5_{49}O_2 \to O_1RO_2$, $R \in \{R^5_2, R^5_5, R^5_{21}\}$; $O_1R^5_{50}O_2 \to O_1RO_2$, $R \in \{R^5_2, R^5_{19}, R^5_{21}\}$.

Тип 2

$O_1R^5_iO_2 \to O_2R^5_jO_1$;

$O_1R^5_3O_2 \to O_2R^5_4O_1$; $O_1R^5_{22}O_2 \to O_2R^5_{23}O_1$; $O_1R^5_{23}O_2 \to O_2R^5_{22}O_1$;

$O_1R^5_{25}O_2 \to O_2R^5_{26}O_1$; $O_1R^5_{26}O_2 \to O_2R^5_{25}O_1$; $O_1R^5_{32}O_2 \to O_2R^5_{33}O_1$;

$O_1R_{33}^5O_2 \to O_2R_{32}^5O_1;\ O_1R_{37}^5O_2 \to O_2R_{26}^5O_1;\ O_1R_{37}^5O_2 \to O_2R_{28}^5O_1;$
$O_1R_{39}^5O_2 \to O_2R_{28}^5O_1; O_1R_{40}^5O_2 \to O_2R_{25}^5O_1.$

Тип 3

$O_1R_i^3O_2 \to O_1R_j^5O_2,$
$O_1R_1^3O_2 \to O_1R_{21}^5O_2;\ O_1R_2^3O_2 \to O_1R_{10}^5O_2;\ O_1RO_2 \to O_1R_{29}^5O_2, R \in \{R_2^3,...,R_7^3\}.$

Тип 4

$O_1R_i^5O_2 \to O_1R_j^3O_2,$
$O_1R_8^5O_2 \to O_1R_1^3O_2 \vee O_1R_{i2}^{\#}O_3;\ O_1R_{21}^5O_2 \to O_1R_1^3O_2;\ O_1R_{36}^5O_2 \to O_1R_2^3O_2;$
$O_1R_{42}^5O_2 \to O_1R_2^3O_2;\ O_1R_{43}^5O_2 \to O_1R_2^3O_2; O_1R_{52}^5O_2 \to O_1R_1^3O_2.$

Тип 5

$O_1R_i^5O_2 \wedge O_3R_j^5O_1 \to O_3R_k^5O_2,$
$O_1R_6^5O_2 \wedge O_3RO_1 \to O_3RO_2,\ R \in \{R_6^5, R_7^5\};$
$O_1R_7^5O_2 \wedge O_3RO_1 \to O_3R_{34}^5O_2,\ R \in \{R_6^5, R_7^5, R_8^5, R_{10}^5, R_{21}^5, R_{31}^5,..., R_{34}^5, R_{42}^5, R_{43}^5\};$
$O_1R_{11}^5O_2 \wedge O_3RO_1 \to O_3R_8^5O_2,\ R \in \{R_{31}^5, R_{32}^5, R_{33}^5, R_{36}^5\};$
$O_1R_{31}^5O_2 \wedge O_3R_{22}^5O_1 \to O_3R_{22}^5O_2;\ O_1R_{31}^5O_2 \wedge O_3R_{34}^5O_1 \to O_3R_8^5O_2;$
$O_1R_{31}^5O_2 \wedge O_3R_{31}^5O_1|_\omega \to O_3R_{32}^5O_2;\ O_1R_{33}^5O_2 \wedge O_3R_{33}^5O_1|_\omega \to O_3R_{33}^5O_2,$
де ω – ненульова проєкція O_3 на O_1;
$O_2R_{33}^5O_1 \wedge O_3R_{22}^5O_1 \to O_3R_{22}^5O_2;$
$O_1R_7^5O_2 \wedge O_3R_{34}^5O_1 \to O_3RO_2,\ R \in \{R_3^5, R_4^5, R_6^5, R_7^5, R_{31}^5,..., R_{36}^5, R_{42}^5, R_{43}^5, R_{48}^5\}.$

Тип 6

$O_1R_{31}^5O_2 \wedge O_3R_{36}^5O_1 \wedge O_3R_{31}^5O_2 \to O_3R_8^5O_2;$
$O_1R_{32}^5O_2 \wedge O_1R_{31}^5O_2 \wedge O_3R_{21}^5O_2 \to O_1R_{31}^5O_2;$
$O_1R_i^3O_2 \wedge O_2R_i^3O_3 \wedge O_1R_3^4O_2 \wedge O_2R_3^4O_3 \to O_3R_{25}^5O_1,\ i \in \overline{1,7};$
$O_1R_i^3O_2 \wedge O_2R_i^3O_3 \wedge O_1R_7^4O_2 \wedge O_2R_7^4O_3 \to O_3R_{26}^5O_1,\ i \in \overline{1,7};$
$O_1RO_2 \wedge ... \wedge O_{n-1}RO_n|_\chi \to O_1RO_n,\ R_i \in \{R_{32}, R_{33}\},$
де χ означає ненульову проєкцію O_n на O_1,
$O_1RO_2 \wedge ... \wedge O_{n-1}RO_n|_\nu \to O_nR_{45}^5,\ R \in \{R_{39}^5, R_{40}^5\},$
де ν – означає перебувати на одній прямій.

6. Псевдофізична каузальна логіка

R_0^6 – загальне умовне відношення, що відображає залежність будь-якого роду між подіями p і q,

R_1^6 – необхідне і достатнє причинне відношення між подіями p і q,

R_2^6 – достатнє причинне відношення між подіями p і q,

R_3^6 – обумовлює причинне відношення між подіями p і q,

$R_{1,1}^6$ – p є причина q,

$R_{1,2}^6$ – q є наслідок p.

Правила виводу

$p_i R p_j \to p_i R_0^6 p_j, R \in \{R_1^6, R_2^6, R_3^6\}$,

$p_i R_4^6 p_j \to p_j R_5^6 p_i$,

$p_i \to p_j R_3^6 p_i$,

$p_i R p_j, p_j R p_k \to p_i R p_k, R \in \{R_0^6, R_1^6, R_2^6\}$,

$p_i R p_j, p_j R_1^6 p_k \to p_i R p_k, R \in \{R_0^6, R_2^6, R_3^6\}$,

$p_i R p_j, p_j R_2^6 p_k \to p_i R p_k, R \in \{R_0^6, R_3^6\}$,

$p_i R p_j, p_k R p_j \to (p_i \wedge p_k) R p_j, R \in \{R_0^6, R_1^6, R_2^6, R_3^6\}$,

$p_i R p_j \to p_i R_1^1 p_j, R \in \{R_1^6, R_2^6, R_3^6\}$,

$p_i R p_j, p_j R_4^1(t^*) \to p_i R_4^1(t^*), R \in \{R_1^6, R_2^6, R_3^6\}$.

7. Псевдофізична логіка дій

R_1^7 – дія d_i є частиною дії d_j,

R_2^7 – дія d_i приводить до результату (дії) d_j,

R_3^7 – дія d_i супроводжується дією d_j,

R_4^7 – суб'єкт дія $d_i(S_k)$ має мету (дія) d_j,

R_5^7 – суб'єкт дія $d_i(S_k)$ має мотив (дія) d_j,

Правила виводу (найпоширеніші)

$d_i(t_i) R d_j(t_j) \to t_i R_1^1 t_j$, $R \in \{R_1^6, R_2^6, R_3^6\}$,

$d_i(t_i)R_3^7 d_j(t_j) \to t_i R_3^2 t_j$,

$d_i(S_i, O_j, I_k, t_m) \to d_j(O_j, I_k, t_m)$, де d_j – перебувати біля,

$d_i(S_i, O_j, t_m) \to d_j(S_i, O_j, t_m)$, де d_j – перебувати біля,

$d_i R_2^7 d_j, d_j R_2^7 d_k \to d_i R_2^6 d_k$,

$d_i(S_j) \to d_i R_4^7 d_k$,

$d_i(S_i, l_j, t_k) \to d_i R_2^7 d_j(S_i, l_j, t_m)$,

де $t_m = t_k + \Delta t$, d_i – переміститися всередину (увійти),

d_j – перебувати в,

$d_i(S_i, l_j, t_k) \to d_i R_2^7 d_j(S_i, l_j, t_m)$,

де $t_m = t_k + \Delta t$, d_i – переміститися зсередини (вийти),

d_j – перебувати поза,

$d_i(S_i, l_p, t_k), l_p R_{11}^5 l_m \to d_i(S_i, l_m, t_k)$, де d_i – переміститися всередину,

$d_i(O_i, l_m, t_k), d_j(S_p, O_i, t_k) \to d_i(S_p, l_m, t_k)$,

де d_i – переміститися до,

d_j – перебувати в.

ДОДАТОК Б
ОСНОВНІ ПОНЯТТЯ ТА ФУНКЦІЇ МОВИ CLIPS

CLIPS (C Language Integrated Production System) є мовою програмування для створення продукційних ЕС із прямим висновком.

Відображення типів та констант між CLIPS та Python (табл. Б1).

Таблиця Б1

Відображення примітивних типів між CLIPS та Python

CLIPS	Python
INTEGER	int
FLOAT	float
STRING	str
SYMBOL	Symbol
MULTIFIELD	tuple
FACT_ADDRESS	Fact
INSTANCE_NAME	InstanceName
INSTANCE_ADDRESS	Instance
EXTERNAL_ADDRESS	ffi.CData

FACT_ADDRESS – адреса факту у базі даних фактів;

INSTANCE_NAME – ім’я екземпляра класу;

INSTANCE_ADDRESS – адреса об’єкта класу;

EXTERNAL_ADDRESS – адреса структури даних;

FACT_ADDRESS, INSTANCE_ADDRESS, EXTERNAL_ADDRESS не можуть бути задані константою і можуть бути отримані лише за допомогою функції.

Таблиця Б2

Відображення констант між CLIPS та Python

CLIPS	Python
nil	None
TRUE	True
FALSE	False

Основні функції CLIPS

Таблиця Б3

Математичні функції CLIPS у вигляді *(функція)*

Функція	Опис
pi	Отримання значення числа π

Таблиця Б4

Тригонометричні функції CLIPS у вигляді *(функція <вираз>)*

Функція	Опис
acos	Арккосинус
acosh	Гіперболічний арккосинус
acot	Арккотангенс
acoth	Гіперболічний арккотангенс
acsc	Арккосеканс
acsch	Гіперболічний арккосеканс
asec	Арксеканс
asech	Гіперболічний арксеканс
asin	Арксинус
asinh	Гіперболічний арксинус
atan	Арктангенс
atanh	Гіперболічний арктангенс
cos	Косинус
cosh	Гіперболічний косинус
cot	Котангенс
coth	Гіперболічний котангенс
csc	Косеканс
csch	Гіперболічний косеканс
sec	Секанс
sech	Гіперболічний секанс
sin	Синус
sinh	Гіперболічний синус
tan	Тангенс
tanh	Гіперболічний тангенс

Таблиця Б5

Математичні функції CLIPS у вигляді *(функція <вираз>)*

Функція	Опис
abs	Абсолютне значення
float	Перетворення на тип float
integer	Перетворення на тип integer
deg-grad	Перетворення із градусів на сектори
deg-rad	Перетворення із градусів на радіани
grad-deg	Перетворення із секторів на градуси
rad-deg	Перетворення з радіан на градуси
sqrt	Обчислення квадратного кореня
exp	Обчислення експонентів
log	Обчислення логарифму
log10	Обчислення десяткового логарифму
round	Округлення числа

Таблиця Б6

Математичні функції CLIPS у вигляді *(функція <вираз> <вираз>+)*

Функція	Опис
+	Додавання
–	Віднімання
*	Множення
/	Ділення
div	Цілочисельне ділення
max	Максимальне числове значення
min	Мінімальне числове значення

Таблиця Б7

Математичні функції CLIPS у вигляді *(функція <вираз> <вираз>)*

Функція	Опис
**	Обчислення ступеня числа
mod	Обчислення залишку від розподілу

Таблиця Б8

Опції порівняння CLIPS у вигляді
(функція <вираз> <вираз>+)

Функція	Опис
=	Перевірка рівності
<>	Перевірка нерівності
>	Перевірка більше
>=	Перевірка більше чи однаково
<	Перевірка менше
<=	Перевірка менше чи однаково
eq	Перевірка рівності та типів аргументів
neq	Перевірка нерівності

Таблиця Б9

Логічні функції CLIPS у вигляді
(функція <вираз> <вираз>+)

Функція	Опис
and	Логічне І
or	Логічне АБО

Таблиця Б10

Логічні функції CLIPS у вигляді
(функція <вираз>)

Функція	Опис
not	Логічне НЕ

Таблиця Б11

Функції перевірки приналежності до типу CLIPS у вигляді
(функція <вираз>)

Функція	Опис
numberp	Перевірка, чи належить аргумент до типу float або integer
floatp	Перевірка, чи належить аргумент до типу float
integerp	Перевірка, чи належить аргумент до типу integer
lexemep	Перевірка, чи належить аргумент до типу symbol або string
stringp	Перевірка, чи належить аргумент до типу string
symbolp	Перевірка, чи належить аргумент до типу symbol
Wordp	Синонім функції symbolp
evenp	Перевірка цілого числа на парність
oddp	Перевірка цілого числа на непарність
multifieldp	Перевірка, чи є аргумент складовим полем
sequencep	Синонім функції multifieldp
pointerp	Перевірка, чи належить аргумент до типу external-address

Таблиця Б12

Функції неформатованого виведення на консоль CLIPS у вигляді *(функція <вираз>*)*

Функція	Опис
print	Неформатоване виведення на консоль
println	Неформатоване виведення на консоль із поверненням каретки та переведення рядка

Таблиця Б13

Функції форматованого виведення на консоль CLIPS у вигляді *(функція <стринг_формату> <вираз>*)*

Функція	Опис
format	Форматоване виведення на консоль

Стринг-формату виглядає так:
%[sign][0][width][precision] type

Таблиця Б14

Опція знака sign

sign	Опис
"+"	Знак використовується і для позитивних, і для негативних чисел
"–"	Знак використовується лише для негативних чисел (за замовчуванням)
" "	Для позитивних чисел використовується пробіл, а для негативних чисел використовується знак мінус

Опція "0" вказує, що заповнювачем є символ "0".

Опція ширини поля *width* є десятковим цілим числом і визначає мінімальну ширину поля. За замовчуванням визначається вмістом, тобто відбувається заповнення символом " " (за замовчуванням) або "0", причому для числових типів додавання нуля відбувається з урахуванням знака.

Опція точності precision є десятковим числом і вказує, скільки цифр має відображатися після десяткової точки для значення типу float, якщо опція type="f", або до та після десяткової точки для значення типу float, якщо опція type="g". Для нечислових типів ця опція показує максимальний обсяг поля, тобто скільки символів буде використано із вмісту поля. Ця опція не допускається для типу int.

Таблиця Б15

Опція типу

Функція	Опис
" d "	Десятковий формат
" o "	Вісімковий формат
" x "	Шістнадцятковий формат
" e "	Експонентний формат
" f "	Формат із фіксованою точкою
" g "	Загальний формат
"c"	Символ
"s"	Стринг
"r"	Повернення каретки
"n"	Переклад рядка
"%"	Символ "%"

Таблиця Б16

Функції роботи зі стрингами CLIPS

Функція	Опис
str-cat	Конкатенує зазначені аргументи у стринг (str-cat <expression>*)
sym-cat	Конкатенує зазначені аргументи на символ (sym-cat <expression>*)
sub-string	Вилучення підстрингу зі стрингу у вказаному діапазоні (sub-string <integer-expression> <integer-expression> <string-expression>)
str-index	Отримання за вказаним підстрингом його позиції у стрингу (str-index <lexeme-expression> <lexeme-expression>)
upcase	Отримання стринга у верхньому регістрі (upcase <string-or-symbol-expression>)
lowcase	Отримання стринга в нижньому регістрі (lowcase <string-or-symbol-expression>)
str-compare	Порівняння двох стрингів (str-compare <string-or-symbol-expression> <string-or-symbol-expression>)
str-length	Отримання довжини стринга (str-length <string-or-symbol-expression>)
string-to-field	Отримання зі стрингу даних примітивного типу (string-to-field <string-or-symbol-expression>)

Таблиця Б17

Функції роботи з мультиполями (multifield) CLIPS

Функція	Опис
create$	Створення мультиполя (create$ <expression>*)
nth$	Отримання за вказаним номером елемента мультиполя (nth$ <integer-expression> <multifield-expression>)
member$	Отримання за вказаним елементом мультиполя його номера (member$ <expression> <multifield-expression>)
subsetp	Перевіряє, чи не є одне мультиполе підмножиною іншого мультиполя (subsetp <multifield-expression> <multifield-expression>)
delete$	Видалення елементів мультиполя у вказаному діапазоні (delete$ <multifield-expression> <begin-integer-expression> <end-integer-expression>)
explode$	Створення мультиполя зі стрингу (explode$ <string-expression>)
implode$	Створення стрингу з мультиполя (implode$ <multifield-expression>)
subseq$	Вилучення підпослідовності з мультиполя у вказаному діапазоні (subseq$ <multifield-value> <begin-integer-expression> <end-integer-expression>)
replace$	Заміна елементів мультиполя у вказаному діапазоні на вказані елементи (replace$ <multifield-expression> <begin-integer-expression> <end-integer-expression> <single-or-multi-field-expression>+)
insert$	Додавання в мультиполі із зазначеного номера вказаних елементів (insert$ <multifield-expression> <integer-expression> <single-or-multi-field-expression>+)
first$	Отримання першого елемента мультиполя (first$ <multifield-expression>)
rest$	Отримання залишку (крім першого елемента) мультиполя (rest$ <multifield-expression>)
length$	Отримання кількості елементів мультиполя (length$ <multifield-expression>)
delete-member$	Вилучення зазначених елементів мультиполя (delete-member$ <multifield-expression> <expression>+)
replace-member$	Заміна вказаних елементів мультиполя у діапазоні на вказані елементи (replace-member$ <multifield-expression> <substitute-expression> <search-expression>+)

Оператори (statement) CLIPS

defrule – визначає правило,
deffacts – визначає факт чи факти,
deftemplate – визначає шаблон невпорядкованих фактів,
defglobal – визначає глобальну змінну,
deffunction – визначає функцію користувача,
defclass – визначає клас,

definstances – визначає екземпляр (об’єкт) класу,

defmessage-handler – визначає оброблювача повідомлення, яке надходить об’єкту класу,

defgeneric – визначає generic-функцію, яка, на відміну від звичайної функції, може бути перевантажена,

defmethod – визначає метод, який обробляє відповідний варіант навантаження,

defmodule – визначає модуль (частина програми).

Надалі обмежимося першими п’ятьма операторами для створення продукційної експертної системи із прямим виведенням.

Факти та шаблони

Факти є списком атомарних змінних, які описані або позиційно (впорядковані факти), або на ім’я (невпорядковані або шаблонні факти).

Шаблон – це формальний опис даних, представлених невпорядкованим фактом.

Впорядкований факт являє собою інформацію як впорядковану послідовність полів, розділених пробілами, тобто є впорядкованим мультиполем. Перше поле вказує на ставлення, яке застосовується до інших полів. Впорядковані факти не використовують шаблон. Впорядковані факти обмежені з погляду підтримуваних характеристик. Прикладом упорядкованого факту є (product "bread" "milk" "eggs"), де product – відношення.

Невпорядковані факти являють собою інформацію як словник (ключем є ім’я слота шаблону факту, значенням є значення слота шаблону факту). Для невпорядкованих фактів потрібне визначення шаблонів. Невпорядковані факти гнучкі, оскільки вони підтримують характеристики обмеження для типів даних, значення за замовчуванням та інші. Невпорядковані факти можуть бути змінені після визначення.

Шаблон задається за допомогою такого оператора:

(deftemplate <deftemplate-name> [<comment>] <slot-definition>*)

<slot-definition> ::= <single-slot-definition> | <multislot-definition>

<single-slot-definition> ::= (slot <slot-name> <template-attribute>*)

<multislot-definition> ::= (multislot <slot-name> <template-attribute>*)

<template-attribute> ::= <default-attribute> | <constraint-attribute>

<default-attribute> ::= (default ?DERIVE | ?NONE | <expression>*) | (default-dynamic <expression>*),

де атрибут slot використовується для опису даних, представлених у вигляді поля;

атрибут multislot використовується для опису даних, представлених у вигляді мультиполя;

атрибут default-dynamic задає як значення за замовчуванням вираз, що оцінюються кожного разу, коли визначається факт, що відповідає цьому слоту шаблону;

атрибут default задає статичне значення за замовчуванням. Якщо для цього використовується:

– вираз, то воно оцінюється один раз під час визначення шаблону, і отримана оцінка є значенням за замовчуванням і зберігається разом із шаблоном;

– ключове слово ?DERIVE, значення за замовчуванням автоматично обчислюється на основі обмежень для слота;

– ключове слово ?NONE, значення за замовчуванням має бути явно призначене під час визначення факту.

Якщо атрибути default та default-dynamic не вказані, то приймається default ?DERIVE

<constraint-attribute> визначає обмеження на слоти:

– обмеження на тип значень слота:

(type <type-specification>),

– обмеження на константні значення слота:

(allowed-symbols <symbol-list>)
(allowed-strings <string-list>)
(allowed-lexemes <lexeme-list>)
(allowed-integers <integer-list>)
(allowed-floats <float-list>)
(allowed-numbers <number-list>)
(allowed-instance-names <instance-list>)
(allowed-classes <class-name-list>)
(allowed-values <value-list>),

– обмеження на діапазон значень слота:

(range <range-specification> <range-specification>),

– обмеження на кількість полів мультислота (вказується мінімальна та максимальна кількість):

(cardinality <cardinality-specification> <cardinality-specification>).

Прикладами шаблону неупорядкованого факту є:

```
(deftemplate person "FactTemplate"
   (slot name) (slot age))
(deftemplate hoo "FactTemplate"
```

(slot day) (slot open) (slot close)).

Один або кілька впорядкованих чи невпорядкованих фактів задаються за допомогою такого оператора:

(deffacts <deffacts-name> [<comment>] <RHS-pattern>*),

де <RHS-pattern> є фактом.

Прикладами впорядкованого та невпорядкованих фактів є:

(deffacts fact "Facts"

(product "bread" "milk" "eggs"))

(person (name "John") (age 30))

(hoo (day Monday) (open "8:00 am") (close "6:00 pm"))).

Глобальні та локальні змінні

Глобальні змінні діють у всій програмі. *Локальні змінні* діють тільки всередині операторів deffunction та defrule, в яких вони визначені. Ці змінні позначаються як ?*змінна** та ?*змінна* відповідно.

Змінні можуть мати значення «поле» або «мультиполе».

Одна чи кілька глобальних змінних задаються за допомогою такого оператора:

(defglobal [<defmodule-name>] <global-assignment>*)

<global-assignment> ::= <global-variable> = <expression>

<global-variable> ::= ?*<symbol>*.

Наприклад, (defglobal ?*count* = 0).

Функція bind встановлює нове значення змінної:

(bind <variable> <expression>*).

Наприклад, (bind ?*a* (+ ?a ?c))), (bind ?a (+ ?b ?c)))).

Щоб встановити змінну у вихідне значення, до функції bind не передається нове значення.

Наприклад, (bind ?*a*), (bind ?a).

Правила

Правило задається за допомогою такого оператора:

(defrule <rule-name> [<comment>]

[<declaration>] ; Rule Properties

<conditional - element>*; умова правила

=>

<action>*); заключення правила

Властивості правила задаються оператором declare

<declaration> ::= (declare <rule-property>+)

<rule-property> ::= (salience <integer-expression>) | (auto-focus <boolean-symbol>)

<boolean-symbol> ::= TRUE | FALSE,

де атрибут salience задає значимість правила, атрибут auto-focus у разі значення TRUE визначає автоматичне фокусування, яке виконується під час активізації правила на модулі, в якому визначено правило.

Умову правила представлено у вигляді

<conditional-element> ::= <pattern-CE> | <assigned-pattern-CE> | <not-CE> | <and-CE> | <or-CE> | <logical-CE> | <test-CE> | <exists-CE> | <forall-CE>

<assigned-pattern-CE> ::= ?<variable-symbol> <- <pattern-CE>

<not-CE> ::= (not <conditional-element>)

<and-CE> ::= (and <conditional-element> <conditional-element>+)

<or-CE> ::= (or <conditional-element> <conditional-element>+)

<logical-CE> ::= (logical <conditional-element>+)

<test-CE> ::= (test <function-call>)

<exists-CE> ::= (exists <conditional-element> <conditional-element>+)

<forall-CE> ::= (forall <conditional-element> <conditional-element>+),

де logical неявно реалізує логічне I,

test викликає перевірочну функцію,

exists реалізує квантор існування,

forall реалізує квантор загальності,

<assigned-pattern-CE> використовується для отримання адреси факту, щоб вказати, на який факт слід впливати. Наприклад, в умові правила ?f <- (person), а в заключенні правила (retract ?f),

<pattern-CE> містить обмеження фактів, які відповідають умовам.

Наприклад,

```
(defrule rule "Rule"
   (product "bread" "milk" "eggs")
=>
(assert ((product "bread" "milk" "eggs")))
(defrule closed-today "Rule"
(not (hoo (day = (weekday)))))
=>
(println "We are closed today")),
```

де weekday – ім’я функції користувача, заданої оператором deffunction, hoo – ім’я шаблону, що містить слот day.

Існують такі обмеження фактів:

- обмеження на основі літерала визначає точне значення (літеру), яке відповідає полю або слоту шаблону. Складається лише з констант. Наприклад, (product "bread" "milk" "eggs");
- обмеження на основі метасимволу (wildcard) вказує, що поле або слот шаблону можуть відповідати будь-чому. Використовуються мета-символи? (для одного поля або слота) та $? (для одного або більше полів або слотів). Наприклад, (product $? "milk" $?);
- обмеження на основі змінної використовується для збереження значення поля або слота шаблону, щоб його можна було використовувати пізніше в лівій частині правила в умовах або в правій частині правила як аргумент дії. Наприклад, (product $? var1 "milk" $? var2);
- обмеження на основі з'єднувачів, що використовує логічні зв'язки & (І), | (АБО), ~ (НЕ). Наприклад,

(product $? "milk" | "butter" $?), (product ?product&~"bread"&~"eggs")

- обмеження на основі предикатів, що використовує предикатні функції (перед їх викликом ставиться :) та логічні зв'язки. Наприклад, (person (age ?age&:(>= ?age 13)&:(<= ?age 19))) використовує дві предикатні функції;
- обмеження на основі значення, яке повертається, що використовує одиночне значення примітивного типу, що повертається функцією. Наприклад, (hoo (day =(weekday))), де weekday – ім'я функції користувача, заданої оператором deffunction, hoo – ім'я шаблону, що містить слот day.

Функції

Функція користувача задається за допомогою такого оператора:

(deffunction <name> [<comment>]
(<regular-parameter>* [<wildcard-parameter>])
<action>*)
<regular-parameter>::= <single-field-variable>
<wildcard-parameter>::= <multifield-variable>,
де <regular-parameter> є аргументом, який має своїм значенням поле,
<wildcard-parameter> є аргументом, який має своїм значенням мультиполе,
<action> є дією або виразом, які виконуються під час виклику функції.
Наприклад,
(deffunction weekday () "Function"
(nth$ 7 (local-time))),

де local-time є функцією, яка повертає локальний час у вигляді мультиполя (рік, місяць, день, година, секунда, день тижня, день початку року, прапор переведення годинника на літній час).

Стратегії вирішення конфліктів

1. Depth стратегія

Нещодавно активізовані правила розміщуються вище всіх правил тієї самої значущості (salience) у вигляді цілого числа (наприклад, якщо факт-1 активізує правило-1 і правило-2, а факт-2 активізує правило-3 і правило-4, то якщо факт-1 затверджується до факту-2, правило-3 і правило-4 будуть вищими за правило-1 і правило-2. Однак положення правила-1 щодо правила-2 і правила-3 щодо правила-4 буде довільним).

2. Breadth стратегія

Нещодавно активізовані правила розміщуються нижче за всі правила тієї самої значущості (наприклад, якщо факт-1 активізує правило-1 і правило-2, а факт-2 активізує правило-3 і правило-4, то якщо факт-1 затверджується до факту-2, правило-1 і правило-2 будуть вищими за правила-3 і правила-4. Однак положення правила-1 щодо правила-2 і правила-3 щодо правила-4 буде довільним).

3. Simplicity стратегія

Серед правил однакової значущості нещодавно активізовані правила розміщуються вище за всіх активізованих правил з такою ж або вищою специфічністю (specificity). *Специфіка правила* визначається кількістю порівнянь, які повинні бути виконані в лівій частині правила. Кожне порівняння з константою або раніше пов'язаною змінною додає одиницю специфічності. Кожен виклик функції, зроблений у лівій частині правила, додає одиниці до специфічності. Булеві функції and, or, not додають специфічності правилу. Виклики функцій, зроблені всередині виклику, не додають специфічності правилу.

4. Complexity стратегія

Серед правил однакової значущості нещодавно активізовані правила розміщуються вище всіх активізованих правил з рівною або меншою специфічністю.

5. LEX стратегія

Серед правил однакового значення нещодавно активізовані правила розміщуються за допомогою стратегії OPS5. Спочатку використовується давність (recency) підумов активізованого правила. Кожен факт та об'єкт класу позначаються тегом часу, щоб вказати їх новизну щодо будь-якого іншого факту та об'єкта класу. Підумовам активізованих правил відповідають факти з тегами часу, і підумови сортуються в порядку зменшення тегів часу. Активізовані правила з підумовами, які мають більш пізні теги часу, розміщуються перед активізованими правилами, у яких більш ранні теги часу. Щоб визначити порядок розміщення двох активізованих правил, порівнюються теги часу їх відсортованих підумов, починаючи з найбільших тегів часу. Порівняння про-

довжується доти, доки тег часу підумови одного активізованого правила не стане більшим за тег часу відповідної підумови іншого активізованого правила. Активізоване правило, яке має підумови з великим тегом часу, ставиться перед іншими активізованими правилами. Якщо одне активізоване правило має більше підумов, ніж інше активізоване правило, і всі порівнювані теги часу підумов ідентичні, то активізоване правило з великою кількістю тегів часу підумов розміщується перед іншим активованим правилом. Якщо два активізовані правила мають однакову давність, то активізоване правило з вищою специфічністю розміщується над активізованим правилом з нижчою специфічністю. У CLIPS, на відміну від OPS5, підумови з not мають псевдотеги часу, які використовуються стратегією розв'язання конфліктів LEX. Тег часу підумов з not завжди менше тег часу умов без not.

Наприклад, підумови правил у конфліктній множині:

rule-6: f-1, f-4

rule-5: f-1, f-2, f-3, *

rule-1: f-1, f-2, f-3

rule-2: f-3, f-1

rule-4: f-1, f-2, *

rule-3: f-2, f-1

будуть упорядковані у вигляді:

rule -6: f -4, f -1

rule-5: f-3, f-2, f-1, *

rule-1: f-3, f-2, f-1

rule-2: f-3, f-1

rule-4: f-2, f-1, *

rule-3: f-2, f-1,

де * означає псевдотег часу підумови з not,

f-*номер* означає тег часу підумови.

6. MEA стратегія

Серед правил однакового значення нещодавно активізовані правила розміщуються за допомогою стратегії OPS5. Спочатку використовується тег часу першої підумови активізованого правила. Активізоване правило, яке має підумову з тегом часу, що перевищує тег часу першої підумови іншого активізованого правила, поміщається перед іншим активізованим правилом. Якщо обидва активізовані правила мають однаковий тег часу першої підумови, то для розміщення активізованого правила використовується стратегія LEX. Як і у випадку зі стратегією LEX, підумови з not мають псевдотеги часу.

Наприклад, для попереднього прикладу підумови правила у конфліктній множині будуть упорядковані у вигляді:

rule-2: f-3, f-1

rule-3: f-2, f-1

rule-6: f-1, f-4

rule-5: f-1, f-2, f-3, *

rule-1: f-1, f-2, f-3

rule-4: f-1, f-2. *

7. Random стратегія

Кожному активізованому правилу присвоюється випадкове число, що використовується для визначення його місця серед активізованих правил однакової значимості. Це випадкове число зберігається за умови зміни стратегії, тому той самий порядок відтворюється під час повторного вибору випадкової стратегії (серед активізованих правил, які були при початковій зміні стратегії).

ДОДАТОК В
ОСНОВНІ КЛАСИ МОДУЛЯ CLIPSPY, ЩО ВИКОРИСТОВУЮТЬСЯ ДЛЯ ЧІТКОГО ВИВОДУ

class clips.environment.Environment

Клас навколишнього середовища CLIPS.

Методи

clips.environment.Environment.batch_star(path)

Оцінює пакет команд із зазначеного файла. Еквівалент функції CLIPS batch*, де path – шлях до файла типу str.

clips.environment.Environment.build(construct)

Створює одиночну конструкцію CLIPS. Еквівалент функції CLIPS build, де construct – конструкція типу str.

clips.environment.Environment.clear()

Очищує навколишнє середовище CLIPS. Еквівалент функції CLIPS clear.

clips.environment.Environment.eval(expression)

Оцінює вираз (у вигляді команди, константи чи виразу). Повертає результат оцінювання. Еквівалент функції CLIPS eval,

де expression – вираз типу str, що оцінюється.

clips.environment.Environment.load(path, binary=False)

Завантажує множину конструкцій CLIPS із файла. Еквівалент функції CLIPS load,

де path – шлях до файла типу str,

binary – якщо True, то конструкції завантажуються із файла у бінарному форматі. Конструкції повинні зберігатися і завантажуватися в тому самому форматі. За замовчуванням False.

clips.environment.Environment.reset()

Скидає довкілля CLIPS. Еквівалент функції CLIPS reset.

clips.environment.Environment.save(path, binary=False)

Зберігає множину конструкцій CLIPS у файлі. Еквівалент функції CLIPS save,

де path – шлях до файлатипу str,

binary – якщо True, то конструкції зберігаються у файлі бінарному форматі. За замовчуванням False.

class clips.facts.Facts(env)

Клас для фактів та шаблонів. Усі атрибути та методи цього класу доступні через клас clips.environment.Environment.

Атрибути

clips.facts.Facts.fact_duplication – якщо True, то факти можуть дублюватися. За замовчуванням False.

Методи

clips.facts.Facts.assert_string(string)

Додає факт у базу даних. Повертає об’єкт класу clips.facts.ImpliedFact або clips.facts.TemplateFact,

де string – факт, представлений у вигляді стрингу.

clips.facts.Facts.defined_facts()

Повертає ітератор для фактів, заданих в операторі deffacts.

clips.facts.Facts.facts()

Повертає ітератор для фактів.

clips.facts.Facts.find_defined_facts(name)

Повертає об’єкт класу clips.facts.DefinedFacts на ім’я,

де name – ім’я факту, визначеного в операторі deffacts типу str.

clips.facts.Facts.find_template(name)

Повертає об’єкт класу clips.facts.Template на ім’я,

де name – ім’я шаблону типу str.

clips.facts.Facts.load_facts(facts)

Завантажує багато фактів із файла. Еквівалент функції CLIPS load-facts,

де path – шлях до файла типу str.

clips.facts.Facts.save_facts(path, mode = clips.SaveMode.LOCAL_SAVE)

Зберігає багато фактів у файлі. Еквівалент функції CLIPS save-facts,

де path – шлях до файла типу str.

mode – режим збереження фактів (clips.SaveMode.LOCAL_SAVE=0 – зберігаються лише факти, пов’язані із шаблонами, визначеними у поточному модулі, clips. SaveMode.VISIBLE_SAVE=1 – зберігаються всі факти, помітні для поточного модуля). За замовчуванням clips.SaveMode.LOCAL_SAVE.

clips.facts.Facts.templates()

Повертає ітератор для шаблонів.

class clips.facts.DefinedFacts(env: <MagicMock name="mock.ffi.CData" id="139919262313808">, name: str)

Клас факту, визначеного в операторі deffacts. Доступ до об’єктів цього класу виконується за допомогою методів clips.facts.Facts.find_defined_facts() та clips.facts.Facts.defined_facts().

Атрибути

clips.facts.DefinedFacts.deletable – якщо True, то об’єкт цього класу можна видалити. За замовчуванням True.

clips.facts.DefinedFacts.module – модуль (об’єкт класу clips.modules. Module), у якому факт, визначений в операторі deffacts, визначений. Еквівалент команди CLIPS deffacts-module.

clips.facts.DefinedFacts.name – ім’я факту, визначеного в операторі deffacts типу str.

Методи

clips.facts.DefinedFacts.undefine()

Скасовує визначення об’єкта цього класу. Еквівалент функції CLIPS undeffacts. Об’єкт стає непридатним для використання після цього методу.

class clips.facts.Fact(env: <MagicMock name="mock.ffi.CData" id="139919262313808">, fact: <MagicMock name="mock.ffi.CData" id="139919262313808")

Клас факту. Доступ до об’єктів цього класу виконується за допомогою методу clips.facts.Facts.facts().

Атрибути

clips.facts.Fact.exists – якщо True, то факт знаходиться в базі даних. Еквівалент функції CLIPS fact-existp.

clips.facts.Fact.index – індекс факту у базі даних.

clips.facts.Fact.template – шаблон факту (об’єкт класу clips.facts.Template).

Методи

clips.facts.Fact.retract()

Видаляє факт із бази даних.

class clips.facts.ImpliedFact(env: <MagicMock name="mock.ffi.CData" id="139919262313808">, fact: <MagicMock name="mock.ffi.CData" id="1399192623138)

Клас упорядкованих фактів. Об’єкти цього класу безпосередньо не створюються а модифікуються. Доступ до об’єктів цього класу виконується за допомогою методу clips.facts.Facts.assert_string().

Приклад

```
fact = env.assert_string('(ordered-fact 1 2 3)')
assert fact[0] == 1 # True
assert list(fact) == [1, 2, 3] # True.
```

class clips.facts.Template(env: <MagicMock name="mock.ffi.CData" id="139919262313808">, name: str)

Клас шаблону неупорядкованого факту. У CLIPS шаблон визначено за допомогою оператора deftemplate. Об’єкти цього класу безпосередньо не створюються. Доступ до об’єктів цього класу виконується за допомогою методів clips.facts.Facts.find_template() та clips.facts.Facts.templates().

Атрибути

clips.facts.Template.deletable – якщо True, то об’єкт цього класу можна видалити. За замовчуванням True.

clips.facts.Template.implied – завжди False.

clips.facts.Template.module – модуль (об’єкт класу clips.modules.Module), у якому шаблон визначено. Еквівалент команди CLIPS deftemplate-module.

clips.facts.Template.name – ім’я шаблону типу str.

clips.facts.Template.slots – кортеж слотів шаблону.

Методи

clips.facts.Template.assert_fact(slots)**

Додає невпорядкований факт у базу даних. Еквівалент функції CLIPS assert. Повертає об’єкт класу clips.facts.TemplateFact,

де slots – словник слотів.

clips.facts.Template.facts()

Повертає ітератор для невпорядкованих фактів.

clips.facts.Template.undefine()

Скасовує визначення об’єкта цього класу. Еквівалент функції CLIPS undeftemplate. Об’єкт стає непридатним для використання після цього методу.

class clips.facts.TemplateFact(env: <MagicMock name="mock.ffi.CData" id="140398532084368">, fact: <MagicMock name="mock.ffi.CData" id="1403985320848)

Клас невпорядкованого факту. Об’єкти цього класу безпосередньо не створюються а модифікуються. Доступ до об’єктів цього класу виконується за допомогою методу clips.facts.Template.assert_fact().

clips.facts.TemplateFact.modify_slots(slots)**

Модифікує одне чи більше значень слотів. Еквівалент функції CLIPS modify,

де slots – словник слотів.

Приклад

```
template = env.find_template("person")
fact = template.assert_fact(name="John", age=30)
```

```
# fact = env.assert_string('(person(name "John") (age 30))')
assert fact["age"] == 30 # True
assert dict(fact) == {"name": "John", "age": 30} # True.
```

class clips.facts.TemplateSlot(env: <MagicMock name="mock.ffi.CData" id="139919262313808">, tpl: str, name: str)

Клас слота шаблон. Об’єкти цього класу безпосередньо не створюються. Доступ до об’єктів цього класу здійснюється за допомогою атрибута clips.facts.Template.slots за номером слота, починаючи з 0.

Атрибути

clips.facts.TemplateSlot.allowed_values – кортеж, що містить допустимі константні значення слота. Еквівалент функції CLIPS slot-allowed-values.

clips.facts.TemplateSlot.cardinality – кортеж, що містить мінімальну та максимальну кількість полів мультислота. Еквівалент функції CLIPS deftemplate-slot-cardinality.

clips.facts.TemplateSlot.default_type – тип за замовчуванням (clips.common. TemplateSlotDefaultType.DYNAMIC_DEFAULT=2 – динамічний, clips.common.TemplateSlotDefaultType.NO_DEFAULT=0 – відсутній, clips.common.TemplateSlotDefaultType.STATIC_DEFAULT=1 – статичний. Еквівалент функції CLIPS deftemplate-slot-defaultp.

clips.facts.TemplateSlot.default_value – значення за замовчуванням. Еквівалент функції CLIPS deftemplate-slot-default-value.

clips.facts.TemplateSlot.multifield – Якщо True, то слот є мультислотом.

clips.facts.TemplateSlot.name – ім’я слота типу str.

clips.facts.TemplateSlot.range – кортеж, що містить діапазон значень слота. Еквівалент функції CLIPS deftemplate-slot-range.

clips.facts.TemplateSlot.types – кортеж, що містить типи значень слота. Еквівалент функції CLIPS deftemplate-slot-types.

Приклад

```
template = env.find_template("person")
print(template.slots[1].range).
```

class clips.agenda.Agenda(env: <MagicMock name="mock.ffi.CData" id="139919262313808">)

Клас конфліктної множини (містить активізовані правила, відсортовані відповідно до значущості та стратегії вирішення конфлікту). Усі атрибути та методи цього класу доступні через клас clips.environment.Environment.

Атрибути

clips.agenda.Agenda.agenda_changed – якщо True, то відбулися зміни активізованих правил.

clips.agenda.Agenda.agenda.focus – модуль типу clips.modules.Module, пов'язаний із поточним фокусом. Еквівалент функції CLIPS get-focus.

clips.agenda.Agenda.salience_evaluation – поведінка оцінки значущості (clips.common.SalienceEvaluation.EVERY_CYCLE=2 – оцінює значимість під час визначення, активізації (умова правила задоволена) та спрацьовування правила, clips.common.SalienceEvaluation.WHEN_ACTIVATED=1 – оцінює значимість під час визначення та активізації правила, clips.common. SalienceEvaluation.WHEN_DEFINED=0 – оцінює значимість під час визначення правила). Еквівалент функції CLIPS get-salience-evaluation. За замовчуванням clips.common.SalienceEvaluation.WHEN_DEFINED.

clips.agenda.Agenda.agenda.strategy – стратегія вирішення конфлікту (clips.common.Strategy.BREADTH=1, clips.common.Strategy.COMPLEXITY=4, clips.common.Strategy.DEPTH=0, clips.common.Strategy.MEA=3, clips.common.Strategy.RANDOM=6, clips.common.Strategy.SIMPLICITY=5). Еквівалент функції CLIPS get-strategy. За замовчуванням clips.common. Strategy.DEPTH.

Методи

clips.agenda.Agenda.activations()

Повертає ітератор для активізованих правил.

clips.agenda.Agenda.clear_focus()

Видаляє всі модулі зі стека фокусу. Еквівалент функції CLIPS clear-focus-stack.

clips.agenda.Agenda.delete_activations()

Видаляє всі активізовані правила з конфліктної множини.

clips.agenda.Agenda.find_rule(name)

Повертає об'єкт класу clips.agenda.Rule на ім'я,

де name – ім'я правила типу str.

clips.agenda.Agenda.refresh(module = None)

Перераховує значимість активізованих правил у конфліктноій множині та переупорядковує конфліктну множину. Еквівалент функції CLIPS refresh-agenda,

де module – модуль (об'єкт класу clips.modules.Module). За замовчуванням None (очищаються всі модулі).

clips.agenda.Agenda.reorder(module = None)

Перевпорядковує активізовані правила у конфліктній множині. Повинен бути викликаний після зміни стратегії вирішення конфлікту,

де module – модуль (об’єкт класу clips.modules.Module). За замовчуванням None (впорядковуються всі модулі).

clips.agenda.Agenda.rules()

Повертає ітератор для правил.

clips.agenda.Agenda.run(limit = None)

Запускає правила з конфліктної множини. Повертає кількість запущених правил,

де limit – кількість правил, що запускаються з конфліктної множини. За замовчуванням None (усі правила).

class clips.agenda.Rule(env: <MagicMock name="mock.ffi.CData" id="139919262313808">, name: str)

Клас правила. У CLIPS правило визначено за допомогою оператора defrule. Об’єкти цього класу безпосередньо не створюються. Доступ до об’єктів цього класу виконується за допомогою методів clips.agenda.Agenda.find_rule() та clips.agenda.Agenda.rules ().

Атрибути

clips.agenda.Rule.deletable – якщо True, то об’єкт цього класу можна видалити. За замовчуванням True.

clips.agenda.Rule.module – модуль (об’єкт класу clips.modules.Module), у якому об’єкт цього класу визначений. Еквівалент функції CLIPS defrule-module.

clips.agenda.Rule.name – ім’я правила типу str.

clips.agenda.Rule.watch_activations – якщо True, відображаються активізовані правила. За замовчуванням False.

clips.agenda.Rule.watch_firings – якщо True, то відображаються правила, що спрацювали. За замовчуванням False.

Методи

clips.agenda.Rule.add_breakpoint()

Додає точку зупинки для правила. Еквівалент функції CLIPS add-break.

clips.agenda.Rule.matches(verbosity= clips.common.Verbosity.TERSE)

Показує часткове зіставлення та активізацію правил. Повертає триелементний кортеж, що містить кількість правильно зіставлених умов, кількість комбінацій правильно зіставлених умов, кількість активізацій,

де verbosity – ступінь деталізації друку інформації (clips.common. Verbosity. VERBOSE – друкується детальна інформація, clips.common.Verbosity.

SUCCINT – друкується коротка інформація, clips.common.Verbosity.TERSE – нічого не друкується). За замовчуванням clips.common.Verbosity.TERSE.

clips.agenda.Rule.refresh()

Очищує правило. Еквівалент функції CLIPS refresh.

clips.agenda.Rule.remove_breakpoint()

Видаляє точку зупинки для правила. Еквівалент функції CLIPS remove-break.

clips.agenda.Rule.undefine()

Скасовує визначення об’єкта цього класу. Еквівалент функції CLIPS undefrule. Об’єкт стає непридатним для використання після цього методу.

class clips.agenda.Activation(env: <MagicMock name="mock.ffi.CData" id=2139919262313808">, act: <MagicMock name="mock.ffi.CData" id="139919262313808")

Клас активізованого правила (умову правила задоволено). Об’єкти цього класу безпосередньо не створюються. Доступ до об’єктів цього класу виконується за допомогою методу clips.agenda.Agenda.activations().

Атрибути

clips.agenda.Activation.name – ім’я активізованого правила.

clips.agenda.Activation.salience – значимість активізованого правила типу int.

Методи

clips.agenda.Activation.delete()

Видаляє активізоване правило з конфліктної множини.

class clips.functions.Functions(env: <MagicMock name="mock.ffi.CData" id="140398532084368">)

Клас функцій, generics – функцій та методів. Об’єкти цього класу безпосередньо не створюються. Усі методи цього класу доступні через клас clips.environment.Environment.

Методи

clips.functions.Functions.call(function, *arguments)

Викликає функцію,

де function – ім’я функції типу str,

arguments – кортеж аргументів функції.

clips.functions.Functions.define_function(function, name = None)

Визначає функцію,

де function – об’єкт, що викликається,

name – ім’я функції типу str. За замовчуванням None (використовується ім’я функції Python).

clips.functions.Functions.find_function(name)

Повертає об’єкт класу clips.functions.Function на ім’я,

де name – ім’я функції типу str.

clips.functions.Functions.find_generic(name)

Повертає об’єкту класу clips.functions.Generic на ім’я,

де name – ім’я функції типу str.

clips.functions.Functions.functions()

Повертає ітератор для функцій.

clips.functions.Functions.generics()

Повертає ітератор для generics – функцій та методів.

class clips.functions.Function(env: <MagicMock name="mock.ffi.CData" id="140398532084368">, name: str)

Клас користувальницької функції. У CLIPS функція визначена за допомогою оператора deffunction. Об’єкти цього класу безпосередньо не створюються. Доступ до об’єктів цього класу виконується за допомогою методів clips.functions.Functions.find_function() та clips.functions.Functions.functions().

Атрибути

clips.functions.Function.deletable – якщо True, то об’єкт цього класу можна видалити. За замовчуванням True.

clips.functions.Function.module – модуль (об’єкт класу clips.modules. Module) у якому об’єкт цього класу визначений. Еквівалент функції CLIPS deffunction-module.

clips.functions.Function.name – ім’я функції.

clips.functions.Function.watch – якщо True, у разі виклику методу clips. agenda.Agenda.rules() відображається об’єкт цього класу. За замовчуванням False.

Методи

clips.functions.Function.undefine()

Скасовує визначення об’єкта цього класу. Еквівалент функції CLIPS undeffunction. Об’єкт стає непридатним для використання після цього методу.

class clips.modules.Modules(env: <MagicMock name="mock.ffi.CData" id="140398532084368">)

Клас глобальних змінних та модулів. Об'єкти цього класу безпосередньо не створюються. Усі атрибути та методи цього класу доступні через клас clips.environment.Environment.

Атрибути

clips.modules.Modules.current_module – поточний модуль (об'єкт класу clips.modules.Module). Еквівалент функції get-current-module.

clips.modules.Modules.globals_changed – якщо True, то хоча б одна глобальна змінна змінилася з моменту останньої перевірки.

clips.modules.Modules.reset_globals – якщо True, то включено поведінку скидання глобальних змінних. За замовчуванням True.

Методи

clips.modules.Modules.find_global(name)

Повертає об'єкт класу clips.functions.Global на ім'я,

де name – ім'я глобальної змінної типу str.

clips.modules.Modules.find_module(name)

Повертає об'єкт класу clips.functions.Module на ім'я,

де name – ім'я модуля типу str.

clips.modules.Modules.globals()

Повертає ітератор для глобальних змінних.

clips.modules.Modules.modules()

Повертає ітератор для модулів.

class clips.modules.Global(env: <MagicMock name="mock.ffi.CData" id="140398532084368">, name: str)

Клас глобальної змінної. У CLIPS глобальна змінна визначена за допомогою оператора defglobal. Доступ до об'єктів цього класу виконується за допомогою методів clips.modules.Modules.find_global() та clips.modules. Modules.globals().

Атрибути

clips.modules.Global.deletable – якщо True, то об'єкт цього класу можна видалити. За замовчуванням True.

clips.modules.Global.module – модуль (об'єкт класу clips.modules.Module), у якому об'єкт цього класу визначений. Еквівалент функції CLIPS defglobal-module.

clips.modules.Global.name – ім'я глобальної змінної.

clips.modules.Global.value – значення глобальної змінної.

clips.modules.Global.watch – якщо True, у разі виклику методу clips.agenda. Agenda.rules() відображається об’єкт цього класу. За замовчуванням False.

Методи

clips.modules.Global.undefine()

Скасовує визначення об’єкта цього класу. Еквівалент функції CLIPS undefglobal. Об’єкт стає непридатним для використання після цього методу.

ДОДАТОК Г
ОСНОВНІ ФУНКЦІЇ МОДУЛЯ SCIKIT-FUZZY, ЩО ВИКОРИСТОВУЮТЬСЯ ДЛЯ НЕЧІТКОГО ВИСНОВКУ

skfuzzy.interp_membership(x, xmf, xx)

Обчислює значення функції власності у цій точці. Повертає значення функції приналежності типу float,

де x – універсум у вигляді одновимірного масиву,

xmf – функція приналежності як одномірного масиву довжиною len(x),

xx – точка типу float, у якій обчислюється значення функції власності.

skfuzzy.defuzz(x, mfx, mode)

Виконує дефазифікацію. Повертає результат дефазифікації,

де x – універсум у вигляді одновимірного масиву,

mfx – функція приналежності як одномірного масиву довжиною len(x),

mode – ім’я методу ("centroid" – центр гравітації, "bisector" – центр площі, "mom" – метод середнього максимуму, "som" – метод лівого максимуму, "lom" – метод правого максимуму).

skfuzzy.dsigmf (x, b1, c1, b2, c2)

Обчислює різницю двох сигмоїдальних функцій.

y = f1 – f 2

f1(x) = 1 / (1 + exp [-c1 * (x - b1)])

f2(x) = 1 / (1 + exp [-c2 * (x - b2)])

Повертає функцію приналежності як одномірного масиву,

де x – універсум у вигляді одновимірного масиву,

b1 – параметр зсуву першої сигмоїди типу float, f1(b1) = 0.5,

c1 – параметр масштабу першої сигмоїди типу float,

b2 – параметр зсуву другої сигмоїди типу float, f2(b2) = 0.5,

c2 – параметр масштабування другої сигмоїди типу float.

skfuzzy.membership.psigmf(x, b1, c1, b2, c2)

Обчислює добуток двох сигмоїдальних функцій.

y = f1(x) * f2(x)

f1(x) = 1 / (1 + exp[- c1 * (x - b1)])

f2(x) = 1 / (1 + exp[- c2 * (x - b2)])

Повертає функцію приналежності як одномірного масиву,

де x – універсум у вигляді одновимірного масиву,

b1 – параметр зсуву першої сигмоїди типу float f1(b1) = 0.5,

c1 – параметр масштабу першої сигмоїди типу float,

b2 – параметр зсуву другої сигмоїди типу float, f2(b2) = 0.5,

c2 – параметр масштабування другої сигмоїди типу float.

skfuzzy.membership.sigmf(x, b, c)

Обчислює сигмоїдальну функцію.

y = 1 / (1 + exp[- c * (x - b)])

Повертає функцію приналежності як одномірного масиву,

де x – універсум у вигляді одновимірного масиву,

b – параметр зсуву типу float, f (b) = 0.5,

c – параметр масштабу типу float. Якщо c > 0, то функція S-подібна. Якщо c < 0, то функція Z-подібна.

skfuzzy.gauss2mf(x, mean1, sigma1, mean2, sigma2)

Обчислює двосторонню функцію Гауса (ліва частина – перша функція Гаусса до mean1, центральна частина (рівна 1) – між mean1 та mean2, права частина – друга функція Гаусса після mean2). Повертає функцію приналежності як одномірного масиву,

де x – універсум у вигляді одновимірного масиву,

mean1 – середнє значення (параметр зсуву) типу float першої функції Гаусса, mean1 <= mean2,

sigma1 – середньоквадратичне відхилення (параметр масштабу) типу float першої функції Гауса,

mean2 – середнє значення (параметр зсуву) типу float другої функції Гауса, mean1 <= mean2,

sigma 2 – середньоквадратичне відхилення (параметр масштабу) типу float другої функції Гауса.

skfuzzy.gaussmf(x, mean, sigma)

Обчислює функцію Гауса. Повертає функцію приналежності як одномірного масиву,

де x – універсум у вигляді одновимірного масиву,

mean – середнє значення (параметр зсуву) типу float,

sigma – середньоквадратичне відхилення (параметр масштабу) типу float.

skfuzzy.skfuzzy.membership.gbellmf(x, a, b, c)

Обчислює узагальнену дзвонову функцію.

y(x) = 1 / (1 + abs([x - c] / a) ** [2 * b])

Повертає функцію приналежності як одномірного масиву,

де x – універсум у вигляді одновимірного масиву,

a – параметр масштабу типу float,

b – параметр форми типу float,

c – параметр зсуву типу float.

skfuzzy.membership.trapmf(x, abcd)

Обчислює трапецеїдальну функцію.

Повертає функцію приналежності як одномірного масиву,

де x – універсум у вигляді одновимірного масиву,

abcd – чотириелементний масив [a, b, c, d], a <= b <= c <= d.

skfuzzy.membership.trimf(x, abc)

Обчислює трикутну функцію.

Повертає функцію приналежності як одномірного масиву,

де x – універсум у вигляді одновимірного масиву,

abc – триелементний масив [a, b, c], a <= b <= c.

skfuzzy.membership.pimf(x, a, b, c, d)

Обчислює добуток функцій skfuzzy.membership.smf() та skfuzzy. membership.zmf(). Повертає функцію приналежності як одномірного масиву,

де x – універсум у вигляді одновимірного масиву,

a – точка типу float, з якою функція починає зростати, a <= b,

b – точка типу float, з якою функція звертається до 1, a <= b,

c – точка типу float, з якою функція починає зменшуватися, c<= d,

d – точка типу float, з якою функція звертається до 0, c <= d.

skfuzzy.membership.smf(x, a, b)

Обчислює S -подібну сплайн-функцію.

y = 0, якщо x <= a,

y = 2 * ((x – b) / (b – a)) ** 2, якщо a <x <= (a + b) / 2,

y = 1 – 2 * ((x – a) / (b – a)) ** 2, якщо (a + b) / 2 <x <b,

y = 1, якщо x>= b.

Повертає функцію приналежності як одномірного масиву,
де x – універсум у вигляді одновимірного масиву,
a – точка типу float, з якою функція починає зростати, a <= b,
b – точка типу float, з якою функція звертається до 1, a <= b.

skfuzzy.membership.zmf(x, a, b)
Обчислює Z-подібну сплайн-функцію.
y = 1, якщо x <= a,
y = 1 – 2 * ((x – a) / (b – a)) ** 2, якщо a <x <= (a + b) / 2,
y = 2 * ((x – b) / (b – a)) ** 2, якщо (a + b) / 2 <x <b,
y = 0, якщо x>= b.
Повертає функцію приналежності як одномірного масиву,
де x – універсум у вигляді одновимірного масиву,
a – точка типу float, з якою функція починає зменшуватися, a <= b,
b – точка типу float, з якою функція звертається до 0, a <= b.

ДОДАТОК Д
ОСНОВНІ КЛАСИ ТА ФУНКЦІЇ МОДУЛЯ SURPRISE, ЩО ВИКОРИСТОВУЮТЬСЯ ДЛЯ ОБЧИСЛЕННЯ РЕЙТИНГУ

МЕТОДИ ОБЧИСЛЕННЯ РЕЙТИНГУ

class surprise.prediction_algorithms.matrix_factorization.SVD (n_factors=100, n_epochs=20, biased=True, init_mean=0, init_std_dev=0.1, lr_all=0.005, reg_all=0.02, lr_bu=None, lr_bi=None, lr_pu=None, lr_qi=None, reg_bu=None, reg_bi=None, reg_pu=None, reg_qi=None, random_state=None, verbose=False).

Створює модель SVD,

де n_factors – кількість чинників. За замовчуванням 100,

n_epochs – кількість ітерацій процедури SGD. За замовчуванням 20,

biased – якщо True, то використовується поріг. За замовчуванням True,

init_mean – середнє нормального розподілу для ініціалізації векторів факторів. За замовчуванням 0,

init_std_dev – середньоквадратичне відхилення нормального розподілу для ініціалізації векторів факторів. За замовчуванням 0.1,

lr_all – параметр, що визначає швидкість навчання для всіх параметрів. За замовчуванням 0.005,

reg_all – параметр регуляризації для всіх параметрів. За замовчуванням 0.02,

lr_bu – параметр, що визначає швидкість навчання для bu. Має пріоритет над аргументом lr_all. За замовчуванням None,

lr_bi – параметр, що визначає швидкість навчання для bi. Має пріоритет над аргументом lr_all. За замовчуванням None,

lr_pu – параметр, що визначає швидкість навчання для pu. Має пріоритет над аргументом lr_all. За замовчуванням None,

lr_qi – параметр, що визначає швидкість навчання для qi. Має пріоритет над аргументом lr_all. За замовчуванням None,

reg_bu – параметр регуляризації для bu. Має пріоритет над аргументом reg_all. За замовчуванням None,

reg_bi – параметр регуляризації для bi. Має пріоритет над аргументом reg_all. За замовчуванням None,

reg_pu – параметр регуляризації для pu. Має пріоритет над аргументом reg_all. За замовчуванням None,

reg_qi – параметр регуляризації для qi. Має пріоритет над аргументом reg_all. За замовчуванням None,

random_state – генератор випадкових чисел. Може бути числом типу int (початкове число для генератора випадкових чисел) чи об'єктом класу numpy.random.RandomState. Може бути корисним для методу surprise. prediction_algorithms.matrix_factorization.SVD.fit(). За замовчуванням None,

verbose – якщо True, то друкує поточну епоху. За замовчуванням False.

Атрибути

surprise.prediction_algorithms.matrix_factorization.SVD.pu

Об'єкт класу numpy.ndarray з формою (n_users, n_factors), що містить фактори користувачів, де n_users – кількість користувачів.

surprise.prediction_algorithms.matrix_factorization.SVD.qi

Об'єкт класу numpy.ndarray з формою (n_items, n_factors), що містить фактори предметів, де n_items – кількість предметів.

surprise.prediction_algorithms.matrix_factorization.SVD.bu

Об'єкт класу numpy.ndarray з формою (n_users,), що містить пороги користувача, де n_users – кількість користувачів.

surprise.prediction_algorithms.matrix_factorization.SVD.bi

Об'єкт класу numpy.ndarray з формою (n_items,), що містить пороги предмета, де n_items – кількість предметів.

Успадковані методи від класу surprise.prediction_algorithms.algo_base. AlgoBase

fit(trainset)

Налаштовує модель (модифікує об'єкт), використовуючи вказані навчальні дані, і повертає модифікований об'єкт,

де trainset – навчальні дані.

predict(uid, iid, r_ui = None, clip = True, verbose = False)

Обчислює рейтинг для вказаного користувача та предмета. Якщо обчислення рейтингу неможливе (наприклад, відсутній користувач та/або предмет), то повертається глобальне середнє рейтингів. Повертає об'єкт класу surprise. prediction_algorithms.predictions.Prediction (містить ідентифікатор користувача, ідентифікатор предмета, правильний рейтинг, оцінку рейтингу),

де uid – ідентифікатор користувача,

iid – ідентифікатор предмета,

r_ui – правильний рейтинг типу float. За замовчуванням None.

clip – якщо True, то кліпує рейтинг за рейтинговою шкалою. Наприклад, якщо обчислений рейтинг дорівнює 0.5 або 5.5, а шкала рейтингів дорівнює [1,5], то обчислений рейтинг встановлюється 1 або 5 відповідно. За замовчуванням True.

verbose – якщо True, то друкує деталі обчислення рейтингу. За замовчуванням False.

test(testset, verbose = False)

Тестує модель, використовуючи вказані тестові дані. Повертає список об'єктів класу surprise.prediction_algorithms.predictions.Prediction, які містять усі обчислені рейтинги,

де testset – тестові дані, повернуті ітератором перехресної перевірки, функцією surprise.model_selection.split.train_test_split() або методом surprise. Trainset.build_testset(),

verbose – якщо True, то друкує деталі обчислення рейтингу. За замовчуванням False.

class surprise.prediction_algorithms.matrix_factorization.NMF (n_factors=15, n_epochs=50, biased=False, reg_pu=0.06, reg_qi=0.06, reg_bu=0.02, reg_bi=0.02, lr_bu=1, random_state=None, verbose=False).

Створює модель NMF,

де n_factors – кількість чинників. За замовчуванням 15,

n_epochs – кількість ітерацій процедури SGD. За замовчуванням 50,

biased – якщо True, то використовується поріг. За замовчуванням False,

reg_pu – параметр регуляризації для користувачів u. За замовчуванням 0.06,

reg_qi – параметр регуляризації для предметів λi. За замовчуванням 0.06,

reg_bu – параметр регуляризації для bu. Тільки у разі аргументу biased = True. За замовчуванням 0.02,

reg_bi – параметр регуляризації для bi. Тільки у разі аргументу biased = True. За замовчуванням 0.02,

lr_bu – параметр, що визначає швидкість навчання для bu. Тільки у разі аргументу biased = True. За замовчуванням 0.005,

lr_bi – параметр, що визначає швидкість навчання для bi. Тільки у разі аргументу biased = True. За замовчуванням 0.005,

init_low – нижня межа для випадкової ініціалізації факторів. Має бути більше 0, щоб фактори були невід'ємними. За замовчуванням 0,

init_high – верхня межа для випадкової ініціалізації факторів. За замовчуванням 1,

random_state – генератор випадкових чисел. Може бути числом типу int (початкове число для генератора випадкових чисел) чи об'єктом класу numpy. random.RandomState. Може бути корисним для методу surprise.prediction_ algorithms.matrix_factorization.NMF.fit(). За замовчуванням None,

verbose – якщо True, то друкує поточну епоху. За замовчуванням False.

Атрибути

surprise.prediction_algorithms.matrix_factorization.NMF.pu – об'єкт класу numpy.ndarray з формою (n_users, n_factors), що містить фактори користувачів, де n_users – кількість користувачів, n_factors – кількість факторів.

surprise.prediction_algorithms.matrix_factorization.NMF.qi – об'єкт класу numpy.ndarray з формою (n_items, n_factors), що містить фактори предметів, де n_items – кількість предметів, n_factors – кількість факторів.

surprise.prediction_algorithms.matrix_factorization.NMF.bu – об'єкт класу numpy.ndarray з формою (n_users,), що містить пороги користувача, де n_users – кількість користувачів.

surprise.prediction_algorithms.matrix_factorization.NMF.bi – об'єкт класу numpy.ndarray з формою (n_items,), що містить пороги предмета, де n_items – кількість предметів.

Наслідувані методи від класу surprise.prediction_algorithms.algo_base. AlgoBase

fit(trainset)

Налаштовує модель (модифікує об'єкт), використовуючи вказані навчальні дані. Повертає модифікований об'єкт,

де trainset – навчальні дані.

predict(uid, iid, r_ui = None, clip = True, verbose = False)

Обчислює рейтинг для вказаного користувача та предмета. Якщо обчислення рейтингу неможливе (наприклад, відсутній користувач та/або предмет), то повертається глобальне середнє рейтингів. Повертає об'єкт класу surprise. prediction_algorithms.predictions.Prediction (містить ідентифікатор користувача, ідентифікатор предмета, правильний рейтинг, оцінку рейтингу),

де uid – ідентифікатор користувача,

iid – ідентифікатор предмета,

r_ui – правильний рейтинг типу float. За замовчуванням None,

clip – якщо True, то кліпує рейтинг за рейтинговою шкалою. Наприклад, якщо обчислений рейтинг дорівнює 0.5 або 5.5, а шкала рейтингів дорівнює [1,5], то обчислений рейтинг встановлюється 1 або 5 відповідно. За замовчуванням True,

verbose – якщо True, то друкує деталі обчислення рейтингу. За замовчуванням False.

test(testset, verbose = False)

Тестує модель, використовуючи вказані тестові дані. Повертає список об'єктів класу surprise.prediction_algorithms.predictions.Prediction, які містять усі обчислені рейтинги,

де testset – тестові дані, повернуті ітератором перехресної перевірки, функцією surprise.model_selection.split.train_test_split() або методом surprise. Trainset.build_testset(),

verbose – якщо True, то друкує деталі обчислення рейтингу. За замовчуванням False.

Зауваження. Існує також клас surprise.prediction_algorithms.matrix_ factorization.SVDpp(), який реалізує метод SVD++

class surprise.prediction_algorithms.knns.KNNBasic(k=40, min_k=1, sim_options={}, verbose=True, **kwargs)

Створює модель k найближчих сусідів (KNN),

де k – максимальна кількість сусідів. За замовчуванням 40,

min_k – мінімальна кількість сусідів.

Якщо сусідів недостатньо, то обчислений рейтинг встановлюється у глобальне середнє рейтингів. За замовчуванням 1,

sim_options – словник міри подоби. За замовчуванням {},

verbose – якщо True, то друкує, трасуючи повідомлення оцінки порогів, подібності та інших. За замовчуванням True.

Наслідувані методи від класу surprise.prediction_algorithms.algo_base. AlgoBase

fit(trainset)

Налаштовує модель (модифікує об’єкт), використовуючи вказані навчальні дані, і повертає модифікований об’єкт,

де trainset – навчальні дані.

predict(uid, iid, r_ui = None, clip = True, verbose = False)

Обчислює рейтинг для вказаного користувача та предмета. Якщо обчислення рейтингу неможливе (наприклад, відсутній користувач та/або предмет), то повертається глобальне середнє рейтингів. Повертає об’єкт класу surprise. prediction_algorithms.predictions.Prediction (містить ідентифікатор користувача, ідентифікатор предмета, правильний рейтинг, оцінку рейтингу),

де uid – ідентифікатор користувача,

iid – ідентифікатор предмета,

r_ui – правильний рейтинг типу float. За замовчуванням None,

clip – якщо True, то кліпує рейтинг за рейтинговою шкалою. Наприклад, якщо обчислений рейтинг дорівнює 0.5 або 5.5, а шкала рейтингів дорівнює [1,5], то обчислений рейтинг встановлюється 1 або 5 відповідно. За замовчуванням True,

verbose – якщо True, то друкує деталі обчислення рейтингу. За замовчуванням False.

test(testset, verbose = False)

Тестує модель, використовуючи вказані тестові дані. Повертає список об'єктів класу surprise.prediction_algorithms.predictions.Prediction, які містять усі обчислені рейтинги,

де testset – тестові дані, повернуті ітератором перехресної перевірки, функцією surprise.model_selection.split.train_test_split() або методом surprise. Trainset.build_testset(),

verbose – якщо True, то друкує деталі обчислення рейтингу. За замовчуванням False.

compute_similarities()

Створює матрицю подібності.

get_neighbors(iid, k)

Повертає список k найближчих сусідів користувача чи предмета,

де iid – ідентифікатор користувача чи предмета,

k – кількість найближчих сусідів.

class surprise.prediction_algorithms.knns.KNNWithMeans(k=40, min_k=1, sim_options={}, verbose=True, **kwargs)

Створює модель k найближчих сусідів (KNN) із додаванням середніх рейтингів,

де k – максимальна кількість сусідів. За замовчуванням 40,

min_k – мінімальна кількість сусідів. Якщо сусідів недостатньо, то обчислений рейтинг встановлюється у глобальне середнє рейтингів. За замовчуванням 1,

sim_options – словник міри подібності. За замовчуванням {},

verbose – якщо True, то друкує, трасуючи повідомлення оцінки порогів, подібності та інших. За замовчуванням True.

Успадковані методи від класу surprise.prediction_algorithms.algo_base.AlgoBase

fit(trainset)

Налаштовує модель (модифікує об'єкт), використовуючи вказані навчальні дані, і повертає модифікований об'єкт,

де trainset – навчальні дані.

predict(uid, iid, r_ui = None, clip = True, verbose = False)

Обчислює рейтинг для вказаного користувача та предмета. Якщо обчислення рейтингу неможливе (наприклад, відсутній користувач та/або предмет),

то повертається глобальне середнє рейтингів. Повертає об’єкт класу surprise.prediction_algorithms.predictions.Prediction (містить ідентифікатор користувача, ідентифікатор предмета, правильний рейтинг, оцінку рейтингу),

де uid – ідентифікатор користувача,

iid – ідентифікатор предмета,

r_ui – правильний рейтинг типу float. За замовчуванням None,

clip – якщо True, то кліпує рейтинг за рейтинговою шкалою. Наприклад, якщо обчислений рейтинг дорівнює 0.5 або 5.5, а шкала рейтингів дорівнює [1,5], то обчислений рейтинг встановлюється 1 або 5 відповідно. За замовчуванням True,

verbose – якщо True, то друкує деталі обчислення рейтингу. За замовчуванням False.

test(testset, verbose = False)

Тестує модель, використовуючи вказані тестові дані. Повертає список об’єктів класу surprise.prediction_algorithms.predictions.Prediction, які містять усі обчислені рейтинги,

де testset – тестові дані, повернуті ітератором перехресної перевірки, функцією surprise.model_selection.split.train_test_split() або методом surprise.Trainset.build_testset(),

verbose – якщо True, то друкує деталі обчислення рейтингу. За замовчуванням False.

compute_similarities()

Створює матрицю подібності.

get_neighbors(iid, k)

Повертає список k найближчих сусідів користувача чи предмета,

де iid – ідентифікатор користувача чи предмета,

k – кількість найближчих сусідів.

Зауваження. Словник для міри подібності містить такі ключі:

"name" – ім’я міри подібності ("cosine" – скоригована косинусна подібність, "msd" – подібність середньоквадратичної різниці, "pearson" – коефіцієнт кореляції Пірсона, "pearson_baseline" – стислий (shrunk) коефіцієнт кореляції Пірсона (у разі невеликої кількості рейтингів)). За замовчуванням "msd",

"user_based" – якщо "user_based" = "True", то подібність обчислюється між користувачами. Якщо "user_based" = "False", то подібність обчислюється між предметами. За замовчуванням True,

"min_support" – мінімальна кількість загальних предметів (якщо "user_based" = "True") чи мінімальна кількість загальних користувачів (якщо "user_

based" = "False"), нижче якого міра подібності стає нульовою. Наприклад, якщо |Iuv| <min_support, то sim (u, v) = 0,

"shrinkage" – параметр стиснення (shrunk) (якщо тільки "name"="pearson_baseline similarity"). За замовчуванням 100.

Зауваження. Існують також класи:

surprise.prediction_algorithms.knns.KNNWithZScore – подібний до класу surprise.prediction_algorithms.knns.KNNWithMeans, але ще враховує нормалізацію z-оцінки кожного користувача.

surprise.prediction_algorithms.knns.KNNBaseline – подібний до класу surprise.prediction_algorithms.knns.KNNWithMeans, але замість середнього рейтингу використовує базовий рейтинг.

КЛАС ЧИТАННЯ ФАЙЛА

class surprise.reader.Reader(name=None, line_format="user item rating", sep=None, rating_scale=(1, 5), skip_lines=0)

Створює читача для синтаксичного аналізу рейтингових файлів. У кожному рядку цього файла міститься user, item, rating, і можливо, timestamp у зазначеному порядку та із зазначеними символами-розділювачами,

де name – ім'я вбудованого набору даних ("ml-100k", "ml-1m", "jester"). За замовчуванням None,

line_format – стринг формату рядка файла рейтингів. Містить поля у потрібному порядку. В аргументі line_format поля завжди розділені пробілом. За замовчуванням "user item rating",

sep – символ-розділювач полів аргументу line_format. За замовчуванням None (";"),

rating_scale – рейтингова шкала у вигляді кортежу. За замовчуванням (1, 5),

skip_lines – кількість рядків, що пропускаються з початку файла. За замовчуванням 0 (без пропусків рядків).

КЛАСИ ТА ФУНКЦІЇ УПРАВЛІННЯ НАБОРОМ ДАНИХ

class surprise.dataset.Dataset(reader)

Базовий клас для завантаження наборів даних із зазначеним читачем (об'єкт класу surprise.Reader.Reader). Не можна створювати об'єкти цього класу безпосередньо, а слід використовувати один із таких методів для завантаження наборів даних.

Методи

surprise.dataset.Dataset.load_builtin(name="ml-100k", prompt=True)

Завантажує вбудований набір даних. Якщо ім'я набору даних неправильне, виникає виняток ValueError. Повертає об'єкт класу surprise.dataset. Dataset,

де name – ім'я вбудованого набору даних ("ml-100k", "ml-1m", "jester"). За замовчуванням "ml-100k",

prompt – якщо True, то видає повідомлення перед завантаженням набору даних на диск (якщо він відсутній на диску). За замовчуванням True.

surprise.dataset.Dataset.load_from_df(df, reader)

Завантажує набір даних із кадру даних. Повертає об'єкт класу surprise. dataset.Dataset,

де df – кадр даних (об'єкт класу pandas.DataFrame), що містить рейтинги. Містить стовпці користувача, предмета, рейтингу, що йдуть лише у цьому порядку,

reader – читач як об’єкт класу surprise.reader.Reader (під час створення цього об’єкта слід вказати лише аргумент rating_scale).

surprise.dataset.Dataset.load_from_file(file_path, reader)

Завантажує набір даних із зазначеного файла. Повертає об’єкт класу surprise.dataset.Dataset,

де file_path – шлях до файла рейтингів типу str,

reader – читач як об’єкт класу surprise.reader.Reader (під час створення цього об’єкта не вказується аргумент name).

surprise.dataset.Dataset.load_from_folds(folds_files, reader)

Завантажує навчальні та тестові набори даних із зазначених файлів. Наприклад, набір даних "ml-100k", що читається як об’єкт класу surprise. reader.Reader, можна завантажити у вигляді частин [("u1.base", "u1.test"), …, ("u5.base", "u5.test")]. Використовується разом з ітератором перехресної перевірки surprise.model_selection.split.PredefinedKFold для поділу навчальних та тестових наборів даних. Повертає об’єкт класу surprise.dataset.Dataset,

де folds_files – список кортежів. Кожен кортеж має форму (path_to_ train_file, path_to_test_file),

reader – читач як об’єкт класу surprise.reader.Reader.

surprise.model_selection.split.train_test_split(data, test_size=0.2, train_size=None, random_state=None, shuffle=True)

Поділяє набір даних на перевірочний та тестовий. Повертає навчальний та тестовий набір даних,

де data – набір даних як об’єкта класу surprise.dataset.Dataset для поділу;

test_size – обсяг тестового набору даних. Якщо число типу float, то становить частку рейтингів у тестовому наборі даних. Якщо число типу int, то становить кількість рейтингів у тестовому наборі даних. Якщо None, то test_size встановлюється як різниця розміру аргументу data та аргументу train_size. За замовчуванням 0.2;

train_size – розмір навчального набору даних. Якщо число типу float, то становить частку рейтингів у навчальному наборі даних. Якщо число типу int, то становить кількість рейтингів у навчальному наборі даних. Якщо None, то train_size встановлюється як різниця розміру аргументу data та аргументу test_size. За замовчуванням None;

random_state – генератор випадкових чисел. Може бути числом типу int (початкове число для генератора випадкових чисел) чи об’єктом класу numpy.random.RandomState. Може бути корисним для багаторазового виклику цього методу. За замовчуванням None (використовується, якщо аргумент shuffle = True);

shuffle – якщо True, то тасує дані. За замовчуванням True.

Зауваження. Існують також такі класи ітераторів перехресної перевірки:

surprise.model_selection.split.KFold – базовий ітератор,

surprise.model_selection.split.LeaveOneOut – ітератор, у якому користувач має тільки один рейтинг у тестовому наборі даних,

surprise.model_selection.split.PredefinedKFold – ітератор, який використовується, коли набір даних завантажується за допомогою методу surprise.dataset.Dataset.load_from_folds(),

surprise.model_selection.split.RepeatedKFold – ітератор, що повторює вказану кількість разів ітераторів surprise.model_selection.split.KFold,

surprise.model_selection.split.ShuffleSplit – ітератор із випадковими навчальними та тестовими наборами даних.

Зауваження. Існує також така функція перехресної перевірки: surprise.model_selection.validation.cross_validate().

МЕТРИКИ ТОЧНОСТІ ОБЧИСЛЕННЯ РЕЙТИНГУ

surprise.accuracy.mae(predictions, verbose=True)

Обчислює середню абсолютну помилку (MAE). Якщо аргумент predictions порожній, виникає виняток ValueError,

де predictions – список об’єктів класу surprise.prediction_algorithms. predictions.Prediction, повернутих методом test();

verbose – якщо True, то друкує обчислене значення. За замовчуванням True.

surprise.accuracy.mse(predictions, verbose=True)

Обчислює середню квадратичну помилку (MSE). Якщо аргумент predictions порожній, виникає виняток ValueError,

де predictions – список об’єктів класу surprise.prediction_algorithms. predictions.Prediction, повернутих методом test();

verbose – якщо True, то друкує обчислене значення. За замовчуванням True.

surprise.accuracy.rmse(predictions, verbose=True)

Обчислює корінь середньої квадратичної помилки (RMSE). Якщо аргумент predictions порожній, виникає виняток ValueError,

де predictions – список об’єктів класу surprise.prediction_algorithms. predictions.Prediction, повернутих методом test();

verbose – якщо True, то друкує обчислене значення. За замовчуванням True.

Зауваження. Існує також такий метод: surprise.accuracy.fcp() – обчислює частки конкордантних пар (Fraction of Concordant Pairs).

ДЛЯ НОТАТОК

Навчальне видання

Нескородєва Тетяна Василівна
Федоров Євген Євгенович
Січко Тетяна Василівна
Нескородєва Анастасія Романівна

ЕКСПЕРТНІ ТА РЕКОМЕНДАЦІЙНІ СИСТЕМИ

Навчальний посібник

Редактор О. А. Солдатова
Технічний редактор Т. О. Важеніна-Гопрак

Підписано до друку 09.05.2023
Формат 60 х 84/16. Папір офсетний.
Друк – цифровий. Умовн. друк. арк. 13,02
Тираж 30. Зам. 2

Донецький національний університет імені Василя Стуса
21021, м. Вінниця, 600-річчя, 21
Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру
серія ДК № 5945 від 15.01.2018