МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Звенігородський О.С., Зінченко О.В., Чичкарьов Є.А., Кисіль Т.М.

Artificial Intelligence

ШТУЧНИЙ ІНТЕЛЕКТ

Вступний курс

**Навчальний посібник для студентів
спеціальностей 121 Інженерія програмного забезпечення,
122 Комп'ютерні науки, 123 Комп'ютерна інженерія, 124 Системний аналіз,
125 Кібербезпека, 126 Інформаційні системи та технології,
172 Електронні комунікації та радіотехніка**

Київ 2022

УДК 004.8

Звенігородський О.С., Зінченко О.В., Чичкарьов Є.А., Кисіль Т.М.

Штучний інтелект. Вступний курс: Навчальний посібник. – К.: ДУТ, 2022. – 193 с.

Рецензенти: **Савченко В.А**., доктор технічних наук, професор, директор Навчально-наукового інституту Захисту інформації Державного університету телекомунікацій.

Корнага Я.І., доктор технічних наук, доцент, доцент кафедри технічної безпеки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Навчальний посібник містить викладення навчальної дисципліни «Штучний інтелект» для студентів, що навчаються на бакалаврських програмах спеціальностей 121 Інженерія програмного забезпечення, 122 Комп'ютерні науки, 123 Комп'ютерна інженерія, 124 Системний аналіз, 125 Кібербезпека, 126 Інформаційні системи та технології, 172 Електронні комунікації та радіотехніка. У посібнику розглянуті напрями розвитку штучного інтелекту: подання знань, нейронні мережі, машинне навчання і розпізнавання образів, еволюційні обчислення. Для кожного напряму розглянуті найбільш відомі моделі, методи і алгоритми, що застосовують в сучасних програмних застосунках для вирішення задач прийняття рішень, розпізнавання, прогнозування, оптимізації. Наведені приклади застосування засобів і методів штучного інтелекту в кібербезпеці та телекомунікаціях. Навчальний посібник може використовуватись студентами ВНЗ, що вивчають дисципліни з інформаційних технологій, програмування, телекомунікацій, системного аналізу, економіки, бізнесу.

УДК 004.8

Скорочення та умовні позначення

ADALINE	ADAptive LINer Element
ADF	Augmented Dickey-Fuller test
CNN	Convolutional Neural Network
DFA	Detrended Fluctuation Analysis
DNN	Deep neural network
FRL	Frame Representation Language
GAIA	Global Artificial Intelligence Accelerator
IEC	International Electrotechnical Commission
ISO	International Organization of Standardization
KIF	Knowledge Interchange Format
LSTM	Long Short Term Memory
ML	Mashing Learning
NLP	Natural Language Processing
OWL	Web Ontology Language
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
W3C	World Wide Web Consortium
АСУ	Автоматизована система управління
БНМ	Багатошарові Нейронні Мережі
БД	База даних
ЕС	Експертна система
ІТМ	Інформаційно-телекомунікаційна мережа
МН	Машинне навчання
ПГ	Предметна галузь
СШІ	Системи штучного інтелекту
ШІ	Штучний інтелект
ШНМ	Штучні нейронні мережі
ШПЗ	Шкідливе програмне забезпечення

Зміст

ПЕРЕДМОВА

Дисципліна «Штучний інтелект. Вступний курс» передбачає аудиторні заняття і самостійну роботу із засвоєння теоретичних положень дисципліни, набуття практичних навичок на практичних заняттях та залік.

Головною метою викладання дисципліни є надання студентам базових теоретичних знань щодо методів та алгоритмів , що застосовуються в системах штучного інтелекту та набуття початкових практичних навиків проектування інтелектуальних моделей аналізу даних.

Штучний інтелект – це галузь досліджень, що перебуває на стику наук. Фахівці, що працюють у цій галузі, намагаються зрозуміти, яка поведінка вважається розумною (аналіз), і створити працюючі моделі цієї поведінки (синтез). Дослідники ставлять запитання про те, як за допомогою нових теорій і моделей навчитися розуміти принципи й механізми інтелектуальної діяльності.

Отже *завданнями* вивчення дисципліни «Штучний інтелект. Вступний кур» є:

- дослідження та осмислення фундаментальних понять штучного інтелекту;
- дослідження методів та моделей подання знань у системах штучного інтелекту (СШІ);
- дослідження принципів побудови СШІ, зокрема, експертних систем;
- формування навиків із самостійного оволодіння сучасними технологіями побудови інтелектуальних моделей, подання їх у загальній структурі СШІ.

У результаті вивчення даної навчальної дисципліни студент повинен *знати*:

- основні методи подання та використання знань;
- основи теорії логічного і нечіткого логічного виводу;
- сучасні програмні та інструментальні засоби для проектування СШІ;
- основи машинного навчання;
- основи штучних нейронних мереж;
- основи еволюційних обчислень.

Успішне освоєння дисципліни «Штучний інтелект. Вступний курс» базується на знаннях, отриманих із вищої та дискретної математики, теорії ймовірності, алгоритмізації та програмування, організації баз даних та знань.

Навчальний посібник розроблено відповідно до робочої навчальної програми дисципліни «Штучний інтелект», Освітньої програми «Штучний інтелект» першого (бакалаврського) рівня вищої освіти. Галузь знань 12 Інформаційні технології, спеціальність 122 «Комп’ютерні науки.

Навчальний посібник розроблено відповідно до вимог кредитно-модульної системи оцінювання знань.

ВСТУП

Про сьогоденні успіхи в застосуванні штучного інтелекту (ШІ) в різноманітних галузях людського буття сказано і написано дуже багато. Він став потужним засобом розвитку прогресу в науці, техніці, економіці, медицині, освіті. Знання і розуміння алгоритмів і методів ШІ стає поступово обов'язковим для фаховості випускників вищих навчальних закладів.

Основна частина даного навчального посібника складається з десяти розділів.

Розділ 1 розкриває суть основних понять та напрямків дослідження в галузі штучного інтелекту, наведено науки, на яких гуртуються методи і алгоритми ШІ.

Розділ 2 присвячений методам подання знань в системах ШІ. пошуку рішень у системах штучного інтелекту. Проаналізовано різницю між даними і знаннями, розглянута класифікація знань, що використовуються в ШІ. Розглянуті формальні моделі логічних і продукційних систем, фрейми, семантичні мережі і принципи онтологічних моделей, розглянуті програмні засоби реалізації моделей подання знань

Розділ 3 описує засади нечіткої логіки, терміни і поняття теорії, структура нечіткого виведення, алгоритми нечіткого виведення Мамдані і Сугено, переваги і недоліки нечітких систем.

У 4 розділі розглянуті терміни і означення експертних систем, їх класифікація, галузі застосування, структура, інструментальні засоби. Призначення та принципи побудови статичних та динамічних експертних систем, основні етапи розробки та базові функції. Визначено класи задач, для яких використання технології експертних систем є ефективним.

Розділ 5 присвячений основним напрямам машинного навчання. Розглядаються напрями і задачі машинного навчання, розглядаються терміни і поняття класичної теорії розпізнавання образів, етапи створення систем розпізнавання образів, класифікація методів розпізнавання за типом даних, обговорюються принципи ансамблевих методів розпізнавання: Bagging, Random Forest, Boosting Stacking.

Розділ 6 розкриває суть штучних нейронних мереж, розглядається декілька моделей штучних нейронів, найбільш застосовувані активаційні функції нейронів, топологія штучних нейронних мереж, розглядається математична постановка задачі навчання ШНМ, алгоритм зворотного поширення помилки для навчання багатошарового персептрона.

Сьомий розділ доповнює тему штучних нейронних мереж розглядом мереж глибокого навчання, приділяючи увагу згортковим і рекурсивним нейронним мережам..

У 8 розділі розглядається задача оптимізації на прикладі генетичних алгоритмів, мурашиного алгоритму, алгоритму рою часток, алгоритму бджолиної колонії і алгоритму кожана.

Дев'ятий і десятий розділи розкривають тему застосування ШІ в задачах кібербезпеки і телекомунікаціях.

1 ШТУЧНИЙ ІНТЕЛЕКТ – ЩО ЦЕ?

У терміні Artificial Intelligence (Штучний Інтелект) присутній термін інтелект (*intelligence*), який походить від латинського *intellectus*, що означає розум, розумові здібності людини. Тому спочатку кілька слів про інтелект. Більш розвинуто інтелектом можна назвати здатність мозку розв'язувати задачі, шляхом набуття інформації від зовнішнього середовища, перевірки її на достовірність і тлумачення як знань, що упорядковуються, накопичуються та цілеспрямованого застосовуються до практичних задач після процесу навчання.

Однак єдиного чіткого визначення поняття інтелекту не існує. Інтелект по різному визначався протягом історії. Зараз фахівці з психології виділяють 12 типів інтелекту людини. Зі своїм розвитком ШІ стає одним з типів інтелекту, який теж треба досліджувати.

З часу появи термін ШІ набував і набуває нових змістів. На початку значення цього терміну було сильно зв'язане з принципами комп'ютера і відповідними принципами обробки даних і деякої "подібності" обробки інформації інтелектом людини.

1.1. Означення терміну штучний інтелект

В 1956 році в Дартмутському коледжі (США) інформатик та когнітивіст Джон Маккарті (John McCarthy), організовує двомісячний семінар для дослідників, зацікавлених у нейронних мережах та вивченні інтелекту, де приймається угода про нову назву для цієї галузі дослідження: Artificial Intelligence (Штучний інтелект).

John McCarthy
1927 – 2011

Сам Маккарті в інтерв'ю визначає штучний інтелект таким чином: "Це наука і розробка інтелектуальних машин і систем, особливо інтелектуальних комп'ютерних програм, спрямованих на те, щоб зрозуміти людський інтелект. При цьому використовуються методи не обов'язково біологічно правдоподібні. Інтелект – це обчислювальна частина здатності досягати цілей в світі. Різні види і ступені інтелекту зустрічаються у людей, багатьох тварин і деяких машин".

Означення іншого учасника семінару Марвіна Мінськи (Marvin Minsky): "… штучний інтелект є дисципліна, що вивчає можливість створення програм для вирішення задач, які при розв'язанні їх людиною потребують певних інтелектуальних зусиль".

Згадка про комп'ютер, комп'ютерну програму, комп'ютерні науки присутня у багатьох інших визначеннях дослідників та розробників ШІ. Наприклад, Джордж Люгер (George Luger) у своїй відомій книзі «Штучний інтелект: структури та стратегії вирішення складних проблем» говорить, що: "Штучний інтелект можна визначити як галузь комп'ютерної науки, що займається автоматизацією розумної поведінки". Часто проводять аналогію: мозок є біологічним (фізичним) засобом, що за рахунок ідеальних процесів в

нейронах мозку породжує інтелект, комп’ютер є технічним (фізичним) засобом, що створює штучний інтелект за допомогою програми.

І це не дивно, тому що поки що єдиним засобом реалізації на практиці досягнень штучного інтелекту залишається комп’ютер. Це може бути мікропроцесор, персональний комп’ютер або суперкомп’ютер. Таким чином, якщо розробник заявляє, що його система є інтелектуальною, є простий критерій для перевірки – працююча програма, яка демонструє заявлені інтелектуальні дії.

Часто ШІ визначають шляхом обговорення його основних галузей дослідження та застосування:

1. Використання комп’ютерів для міркувань, розпізнавання зразків (образів), навчання чи іншої форми розумової діяльності.
2. Орієнтація на проблеми, які не розв’язуються алгоритмічним рішенням. Це лежить в основі застосування евристичного пошуку як метода вирішення проблем ШІ.
3. Проблеми з використанням неточної, відсутньої або погано визначеної інформації та використання репрезентативних формалізмів, які дозволяють програмісту компенсувати ці проблеми.
4. Міркування про суттєві якісні особливості ситуації.
5. Спроба вирішити питання семантичного значення, а також синтаксичної форми.
6. Відповіді, які не є ані точними, ані оптимальними, але в якомусь сенсі „достатніми”. Це результат суттєвої залежності від евристичних методів вирішення проблем у ситуаціях, коли оптимальні або точні результати є занадто дорогими або неможливими.
7. Використання великих обсягів специфічних для домену знань при вирішенні проблем.
8. Використання знань на метарівні для здійснення більш досконалого контролю стратегій вирішення проблем. Незважаючи на те, що це дуже складна проблема, яка вирішується у відносно небагатьох сучасних системах, вона постає як важливий напрямок досліджень.

В доповіді «Artificial Intelligence and National Security» конгресу Сполучених Штатів Америки системи штучного інтелекту визначаються наступним чином:

1. Будь-яка штучна система, яка виконує завдання в мінливих і непередбачуваних обставинах без значного людського нагляду або яка може вчитися на власному досвіді і підвищувати продуктивність при роботі з наборами даних.
2. Штучна система, розроблена в комп’ютерному програмному забезпеченні, фізичному обладнанні або іншому контексті, яка вирішує завдання, що вимагають людського сприйняття, пізнання, планування, навчання, спілкування або фізичних дій.
3. Штучна система, створена для того, щоб думати або діяти як людина, включаючи когнітивні архітектури та нейронні мережі.
4. Сукупність прийомів, включаючи машинне навчання, призначене для моделювання задачі пізнання.

5. Штучна система, покликана діяти раціонально, включаючи інтелектуального програмного агента або втіленого робота, який досягає цілей за допомогою сприйняття, планування, міркувань, навчання, спілкування, прийняття рішень та дії.

В практичних реалізаціях ідеї ШІ мова зазвичай йде про системи штучного інтелекту або інтелектуальні системи. В цьому напрямі найбільш цитованим визначенням штучної інтелектуальної системи є наступні [4]:

- Системи, які мислять, як люди
- Системи, які діють, як люди
- Системи, які мислять раціонально
- Системи, які діють раціонально

Більш детально ці визначення подані в табл. 1.

Таблиця 1 – Деякі означення штучного інтелекту, згруповані за чотирма категоріями

Мислити як людина	Мислити раціонально
"Нове захоплююче зусилля змусити комп'ютери думати... машини з розумом у повному і буквальному сенсі." (Haugeland, 1985) "[Автоматизація] видів діяльності, які ми асоціюємо з людським мисленням, такі види діяльності, як прийняття рішень, вирішення проблем, навчання .. ." (Hellman, 1978)	"Вивчення розумових здібностей шляхом використання обчислювальних моделей." (Charniak and McDermott, 1985) "Вивчення обчислень, які дозволяють сприймати, міркувати і діяти." (Winston, 1992)
Діяти як людина	**Діяти раціонально**
"Мистецтво створення машин, які виконують функції, які вимагають розуму, коли їх виконують люди." (Kurzweil, 1990) "Дослідження того, як змусити комп'ютери робити речі, в яких на даний момент люди краще." (Rich and Knight, 1991)	"Обчислювальний інтелект є дослідження дизайну інтелектуальних агентів." (Poole *et at*, 1998) "ШІ ... займається розумною поведінкою артефактів." (Nilsson, 1998)

Визначення зверху стосуються процесів мислення та міркувань, тоді як унизу – поведінки. Визначення зліва вимірюють успіх у термінах відповідності людській діяльності, тоді як визначення праворуч вимірюють ідеальний показник продуктивності, який називається раціональністю. Система є раціональною, якщо вона робить «правильні речі», враховуючи те, що вона знає.

Історично склалося так, що всі чотири підходи до ШІ використовувалися різними людьми з різними методами. Підхід, орієнтований на людину, має бути частково емпіричною наукою, що включає спостереження та гіпотези щодо

людської поведінки. Раціоналістський підхід передбачає поєднання математики та інженерії.

Центральна наукова мета ШІ – зрозуміти принципи, які роблять розумну поведінку можливою в природних або штучних системах.

Штучний інтелект (ШІ) є загальноприйнятою назвою для галузі, але термін «штучний інтелект» є джерелом великої плутанини, оскільки штучний інтелект можна інтерпретувати як протилежність справжньому.

1.2. Тест Тюрінга

Ідею запропонував Алан Тюрінг у статті «Computing machinery and intelligence», опублікованій в 1950 році у філософському журналі Mind. Ця ідея про те, що інтелект визначається зовнішньою поведінкою, була мотивацією для тесту на інтелект, розробленого Тюрінгом [1950], який став відомий як тест Тюрінга. Тест Тюрінга складається з імітаційної гри, де дослідник (експерт) може задати співрозмовнику (людині і інтелектуальній системі) будь-яке запитання через текстовий інтерфейс. Якщо експерт не може відрізнити співрозмовника від людини, співрозмовник (інтелектуальна система) повинен бути розумним.

Alan Mathison Turing
1912-1954

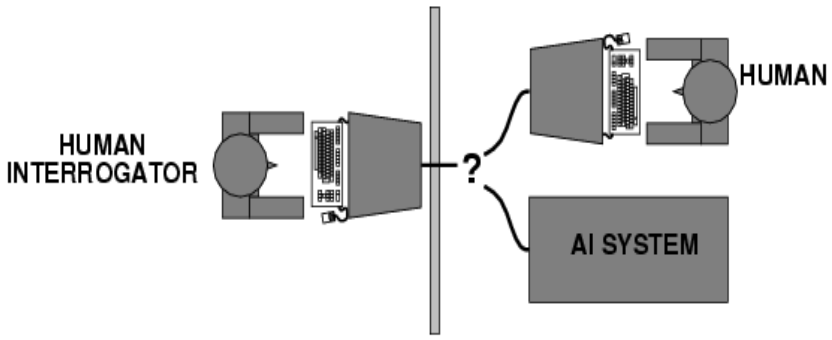


Рисунок 1.1. – Схема проведення тесту Тюрінга

Приклад діалогу експерта і інтелектуальної системи.

Експерт: Чи не знаходите Ви, що в першому рядку Вашого сонета: «Порівняю тебе я з літнім днем» вираз «з весняним днем» звучав би краще?

Співрозмовник: Воно порушувало б розмір вірша.

Експерт: А якщо сказати «із зимовим днем»? З розміром тут усе гаразд.

Співрозмовник: Це так, але ніхто не захоче, щоб його порівнювали із зимовим днем.

Експерт: А хіба містер Піквік не нагадує Вам Різдво?

Співрозмовник: В певному сенсі, так.

Експерт: Але Різдво – зимовий день, і я не думаю, щоб містер Піквік мав щось проти цього порівняння.

Співрозмовник: Я не думаю, що ви серйозно. Під зимовим днем розуміється типовий зимовий день, а не такий особливий, як Різдво.

Було багато дискусій щодо тесту Тюрінга. На жаль, хоча це може бути тестом на те, як розпізнати інтелект, він не дає шлях до нього; щороку намагатися підробити це не здається корисним шляхом дослідження.

1.3. Напрями штучного інтелекту

Символьний і конекціоністський ШІ

Зараз відбувається конкуренція між двома великими напрямами ШІ: символьним і конекціоністським. Ці напрями визначаються наступним чином.

Top-Down AI (низхідний) або семіотичний, символьний. Створення експертних систем, баз знань та систем логічного виведення, що моделюють та імітують високорівневі психічні процеси: мислення, міркування, мови, емоцій, творчості і т.п.;

Bottom-Up AI (висхідний) або біологічний, конекціоністський. Вивчення нейронних мереж і еволюційних обчислень, які моделюють інтелектуальну поведінку на основі біологічних елементів, а також створення відповідних обчислювальних систем, таких як нейрокомп'ютери.

Зараз перевагу набуває другий, особливо після появи глибокого навчання.

Більшість досягнень першого напряму заснована на Physical Symbol System Hypothesis (гіпотеза про Фізичну Символьну Систему) Ньеюла і Саймона (Newell & Simon), висунуту в 1976 році.

Гіпотеза стверджує що: фізична символьна система має необхідні і достатні засоби для проведення базових інтелектуальних дій, в широкому розумінні.

Іншими словами, без символьних обчислень неможливо виконати осмислені дії, а можливості виконувати символьні обчислення цілком достатньо для того, щоб бути спроможним виконувати осмислені дії.

Таким чином, якщо ми вважаємо, що тварина, або людина, або машина діють осмисллено, це означає, що вони якимось чином виконують символьні обчислення (ваша кішка до певної міри – обчислювальна машина). І навпаки, оскільки комп'ютер спроможний до подібних обчислень, то на його основі може бути створений штучний інтелект.

Основні положення гіпотези

- ❖ система, що реалізована фізично і займається маніпулюванням символами;
- ❖ сутність потенційно інтелектуальна тоді і тільки тоді, коли вона представлена фізичною системою символів;
- ❖ символи повинні позначати;
- ❖ символи повинні бути атомарними;
- ❖ символи можуть об'єднуватися для формування виразів.

Приклад. Поняття Число, як поняття, є ідеальним об'єктом. Цей ідеальний об'єкт ми подаєм у виді послідовності цифр (символів), наприклад, записуючи авторучкою на папері. Таким чином, чорнила на папері є фізичні символи, якими позначений ідеальний об'єкт. Нехай ми маємо два числа, написані одне під

одним, і маємо правила, що визначають операцію додавання (таблицю додавання для двох цифр). Необхідно знайти суму чисел. Діючи згідно з таблицею, під кожним стовпчиком цифр ми записуємо результат додавання з врахуванням переносу в старший розряд. По завершенню маємо символьний запис нового числа (ідеального об’єкта), що є сумою двох чисел.

Серед множини створених систем на основі цієї гіпотези відмітимо General Problem Solver (*GPS*, буквально *Загальний розв’язувач проблем*) – програму для моделювання міркувань людини. Програма створена в 1957 році Гербертом Саймоном (Herbert Simon), Дж. Шоу (J.C. Shaw), і Алленом Ньюелом (Allen Newell), призначена для роботи як універсальна машина для розв’язування проблем (задач). За допомогою GPS можна розв’язати, в принципі, будь-яку формалізовану задачу в символьному поданні. Наприклад: доведення теорем, геометричні задачі й гру в шахи. Вона була заснована на теоретичній роботі Саймона і Ньюелла з логічних машин. GPS була першою комп’ютерною програмою, яка розділила свої знання про задачі (правила, які подавались на вхід програми) від своєї стратегії розв’язування задач (загальний розв’язувальний пошуковий двигун).

Гіпотеза вразлива до критики, але так склалося, що більша частина досліджень штучного інтелекту пішла саме шляхом створення символьних систем.

Сильний та слабкий штучний інтелект

J
o
h
n
S... ий філософ, ввів поняття **Strong Artificial Intelligence** (слабкий штучний інтелект). За його визначенням **Сильний Штучний Інтелект** (СШІ) – це програма, яка «… буде не просто моделлю розуму, вона в буквальному розумом, ... розумінні, в якому людський розум – розум». Іншими словами, СШІ – це програма, яка виконує будь-які інтелектуальні завдання, які може виконувати людина, відповідає рівню інтелекту людини і навіть перевершує його і це є основною метою досліджень в галузі штучного інтелекту. Крім цього СШІ – це штучний інтелект з такими людськими рисами, як свідомість, чутливість, розум, мудрість, інтуїція і самосвідомість.

John R. Searle
1932

Сильний ШІ також називають **Artificial General Intelligence (AGI)** – загальний штучний інтелект або здатність виконувати загальні інтелектуальні дії. Сильний ШІ все ще залишається повністю теоретичним, без практичних прикладів використання. Але це не означає, що розробники ШІ не досліджують (з обережністю) ще один напрям – **Artificial Super Intelligence (ASI)** – штучний суперінтелект, який є штучним інтелектом, що перевершує інтелект або здібності людини.

Слабкий Штучний Інтелект – це система штучного інтелекту для вирішення прикладних інтелектуальних задач. Також відомий як **Artificial Narrow Intelligence (ANI)** – прикладний ШІ або вузький ШІ.

Гіпотеза слабкого ШІ: філософська позиція, згідно з якою машини можуть демонструвати інтелект, але не обов'язково мати розум, розумові стани або свідомість. Термін «Слабкий Штучний Інтелект» розглядають лише як інструмент, який дозволяє розв'язувати задачі, які не потребують повного спектру людських пізнавальних здібностей. Серед інструментів СШІ виділяються наступні:

- Моделювання мислення (Логіка, Нечітка логіка).
- Подання знань.
- Експертні системи.
- Розпізнавання образів.
- Машинне навчання, включаючи штучні нейронні мережі.
- Опрацювання природної мови (NLP).
- Еволюційні обчислення.
- Інтелектуальні агенти.
- Робототехнічні і Smart системи.

Найбільш популярні сьогодні напрями ШІ, що активно розвиваються наведені на рис. 1.2.

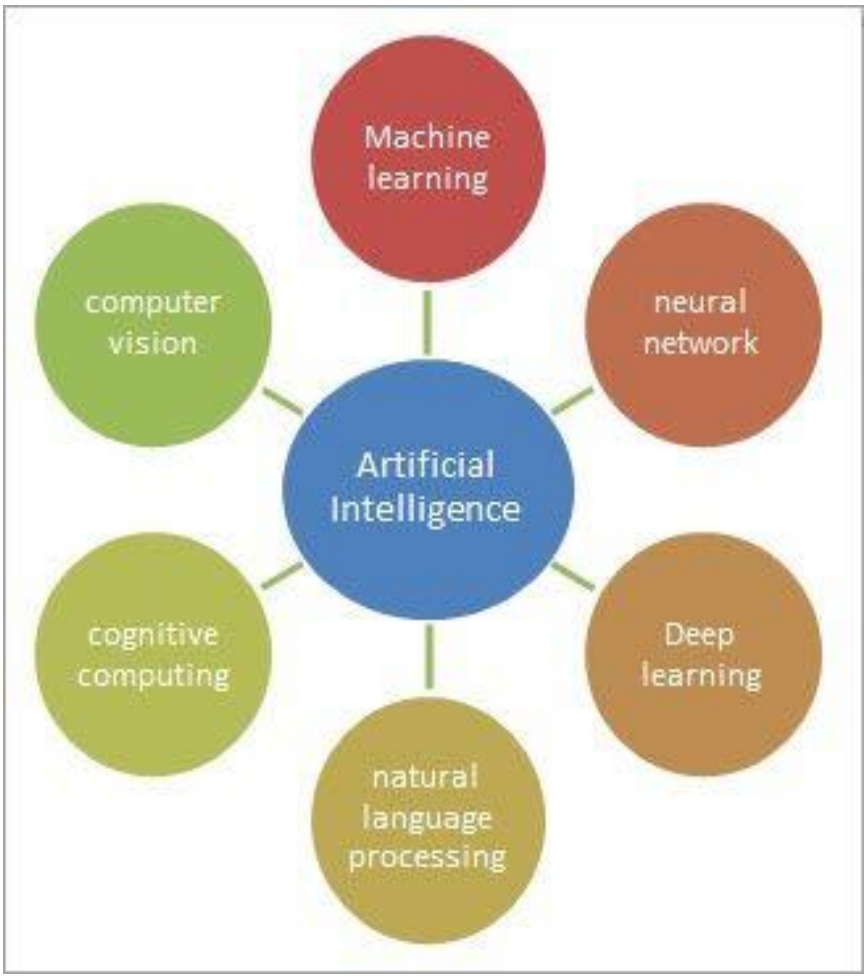


Рисунок 1.2 – Технологічні напрями штучного інтелекту

1.4. Наукові підґрунтя штучного інтелекту

ШІ є результатом досягнень наукової думки людства і не міг виникнути раніше, ніж з’явились відповідні досягнення в цілій низки наук: філософії, математики, економічних науках, нейронауках, психології, комп'ютерній інженерії, кібернетики та теорії управління, мовознавства (лінгвістики). Досягнення в цих науках з великою мірою є основою алгоритмів і методів ШІ. В табл. 1.1 наведені основні напрями згаданих вище наук, що використовуються розробниками ШІ та які надихають їх створювати все нові і нові алгоритми.

Таблиця 1.1 – Наукові підґрунтя штучного інтелекту

Математика	Формальні моделі і доведення теорем, теорія ймовірностей, математичний аналіз, дослідження операцій, чисельні методи, методи оптимізації, матричні обчислення.
Теорія управління та кібернетика	Теорія систем, оптимальне керування, теорія автоматів, прийняття рішень в умовах невизначеності.
Нейронауки	Структура і функції нейрона, фізика і хімія нейронної активності, зв’язок нейронної активності з ментальними станами.
Комп’ютерні науки	Теорія інформації і кодування, структури даних, теорія алгоритмів, мови програмування, операційні системи.
Лінгвістика	Моделі мови, граматики, семантика та прагматика слів, речень, текстів.
Психологія	Пристосування, навчання, сприйняття, ментальні стани.
Філософія	Логіка, міркування, проблема мозок-розум, розум як фізична система, суть категорій філософії: сприйняття, навчання, мова, раціональність, поведінка, сенс.

Слід зауважити, що фахівці з ШІ в своїх алгоритмах часто використовують терміни, взяті з зазначених наук. Але, як правило, за ними розуміються дуже приблизні моделі понять наук, з яких вони запозичені. Про це треба пам’ятати, щоб не переоцінювати і недооцінювати результати досягнень ШІ.

2 МОДЕЛІ ПОДАННЯ ЗНАНЬ В ІНТЕЛЕКТУАЛЬНИХ СИСТЕМАХ

В предметних галузях, де застосовується ШІ, головною практичною задачею є проаналізувати зазвичай велику кількість даних про об'єкти галузі, витягти з них знання і на основі цього аналізу і отриманих знань зробити висновки і виконати дії для покращення стану цих об'єктів. Тому ще з самого виникнення ШІ моделювання знань, як однієї з ознак інтелекту людини, стало важливим напрямом ШІ.

2.1. Визначення поняття даних і знання

Означення 1. ***Дані*** – це інформація, отримана в результаті спостережень або вимірів окремих властивостей (атрибутів), що характеризують об'єкти, процеси та явища предметної галузі.

Дані – це конкретні факти, такі як температура повітря, висота будинку, прізвище співробітника, адреса сайту й ін.

Означення 2. Знання – це зв'язки й закономірності предметної галузі (моделі, закони), отримані в результаті практичної діяльності й професійного досвіду, що дозволяє фахівцям ставити й розв'язувати задачі в даній галузі.

Знання – це добре структуровані дані, або метадані (дані про дані).

Відмінність знань від даних:

- внутрішня інтерпретація;
- структурованість;
- зв'язність;
- семантична метрика;
- активність.

Внутрішня інтерпретація. При збереженні знань у пам'яті СШІ, поряд із традиційними елементами даних, зберігаються й інформаційні структури; що дозволяють інтерпретувати контекст знань (атрибути: прізвище, рік народження, спеціальність, стаж). За атрибутами можна здійснювати пошук потрібної інформації.

Структурованість. Знання складаються з окремих інформаційних одиниць, між якими можна встановити класифікаційні стосунки: рід – вид, клас – елемент, тип – підтип, частина – ціле і т.п.

Зв'язність. Між інформаційними одиницями передбачаються зв'язки різного типу: причина – наслідок, одночасно, бути поруч і ін. Дані зв'язки визначають семантику і прагматику предметної галузі.

Семантична метрика. На множині інформаційних одиниць, збережених у пам'яті, вводяться деякі шкали, що дозволяють оцінити їхню семантичну близькість або силу асоціативного зв'язку. Це дозволяє знаходити в інформаційній базі знання, близькі до нового знання або заданого.

Активність. За допомогою даної властивості підкреслюється принципова відмінність знань від даних. Виконання тих чи інших дій у СШІ ініціюється станом бази знань; При цьому передбачається, що поява нових фактів і зв'язків може активізувати систему.

Центральним питанням побудови систем, заснованих на знаннях, є вибір форми подання знань. Подання знань – це спосіб формального опису знань про

предметну галузь у виді, що забезпечує їхню програмну реалізацію на комп'ютері. Відповідні формалізми, що забезпечують зазначене подання, називають **моделями подання знань.**

Поняття знання і типи знань присутнє у всіх наукових дисциплінах і різняться своїми визначеннями з точки зори предметної галузі, що досліджуються цими дисциплінами. В ШІ знання класифікуються за кількома критеріями (рис. 2.1).

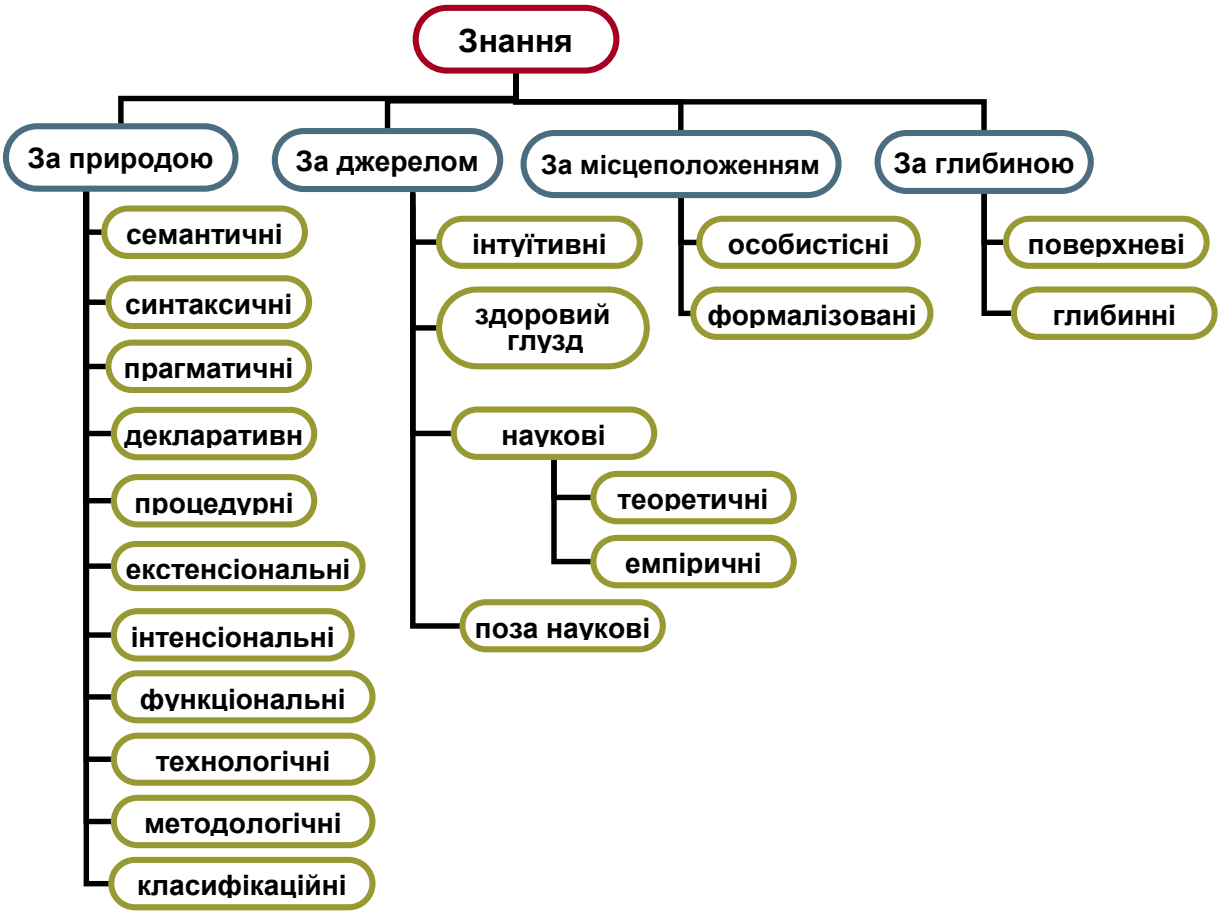


Рисунок 2.1 – Загальна класифікація знань

Синтаксичні знання характеризують синтаксичну структуру потоку інформації, яка не залежить від змісту і змісту понять, які використовуються при цьому.

Семантичні знання розглядаються як структура, що утворює поточний контекст. Вони містить інформацію, безпосередньо пов'язану з поточними значеннями і змістом описуваних понять і визначають стан зв'язків даних в інформаційній базі.

Прагматичні знання зумовлюють найбільш ймовірні зв'язки, що описують дані з точки зору розв'язуваної задачі (узагальнений або «об'єктивний» контекст), наприклад, з урахуванням діючих в даній задачі специфічних критеріїв і угод.

Декларативні знання містять у собі уявлення про структуру понять. (таблиці, списки, абстрактні типи даних). Ці знання наближені до даних, фактів.

Наприклад: вищий навчальний заклад є сукупність факультетів, а кожен факультет в свою чергу є сукупність кафедр.

Процедурні знання, мають активну природу. Вони визначають уявлення про засоби і шляхи отримання нових знань, перевірки знань. Це – знання, «розчинені» в алгоритмах. Для їхньої зміни потрібно змінювати текст програм. Призначення – управління даними.

Інтенсіональні знання – це знання про предметну галузь, що відображають факти, закономірності, властивості і характеристики, справедливі для будь-яких ситуацій, які можуть виникнути в цій предметній галузі. Інтенсіонал поняття – це визначення його через співвіднесення з поняттям більш високого рівня абстракції із зазначенням специфічних властивостей. Інтенсіоналом формують знання про об'єкти.

Екстенсіональні знання – це знання про предметну галузь, що відображають факти, закономірності, властивості і характеристики, типові для конкретних ситуацій або класів однотипних ситуацій, які можуть виникнути в цій галузі. Екстенсіонал поняття – це визначення поняття через перерахування його конкретних прикладів, тобто понять більш низького рівня абстракції. Екстенсіонал об'єднує дані.

Функціональні знання – це знання про виконувані функції окремих предметів і про їх застосування в реальній дійсності.

Технологічні знання – спеціалізовані знання, що забезпечують підтримку технологічних параметрів виробництва; виробничий досвід і навички, які використовуються при вирішенні повсякденних виробничих питань. Це може бути знання послідовності операцій або знання технологічного ланцюжка дозволяють досягати поставлених цілей відповідно до прийнятої технології.

Методологічні знання – знання про методи перетворення дійсності, наукові знання про побудову ефективної діяльності. До методологічних знань відносять знання цілей, форм і напрямків розвитку теорії, методів і способів ефективного перетворення практики.

Класифікаційні знання застосовуються головним чином в науці, є узагальненими, системними знаннями. Приклад – система елементів Д.І. Менделєєва.

Інтуїтивні знання – це вид знань, специфіка яких обумовлена способом їхнього придбання. Це знання, що не потребує доведення і сприймається як достовірне. За способом отримання інтуїція – це визнання об'єктивного зв'язку речей, що не спирається на доказ (інтуїція, від лат. Intueri – споглядати, – є розсуд внутрішнім зором).

Здоровий глузд – знання, що дозволяють приймати правильні рішення і робити правильні припущення, ґрунтуючись на логічному мисленні і накопиченому досвіді. У цьому значенні термін часто акцентує увагу на здатності людського розуму протистояти забобонам, помилкам, містифікаціям.

Наукові знання в будь-якому випадку повинні бути обґрунтованими на емпіричній або теоретичній доказовій основі.

Теоретичні знання – абстракції, аналогії, схеми, що відображають структуру і природу процесів, що протікають в предметної галузі. Ці знання

пояснюють явища і можуть використовуватися для прогнозування поведінки об’єктів. Теоретичний рівень наукового знання припускає встановлення законів, що дають можливість ідеалізованого сприйняття, описи і пояснення емпіричних ситуацій, тобто пізнання сутності явищ. Теоретичні закони мають більш строгий, формальний характер, в порівнянні з емпіричними. Терміни опису теоретичного знання відносяться до ідеалізованих, абстрактних об’єктам. Подібні об’єкти неможливо експериментально перевірити.

Емпіричні знання отримують в результаті застосування емпіричних методів пізнання – спостереження, вимірювання, експерименту. Це знання про видимі взаємозв’язки між окремими подіями і фактами в предметної галузі. Вони, як правило, констатують якісні та кількісні характеристики об’єктів і явищ. Емпіричні закони часто носять імовірнісний характер і не є строгими.

Позанаукові знання можуть бути різними. Паранормальні знання – знання несумісні з наявним гносеологічним стандартом. Широкий клас паранаукового (пара від грец. – близько, при) знання включає в себе вчення або роздуми про феномени, пояснення яких не є переконливим з точки зору критеріїв науковості.

Лженаукові знання – свідомо експлуатують домисли і забобони. Як симптоми лженауки виділяють малограмотний пафос, принципову нетерпимість до спростовуючих доводів, а також претензійність. Лженаукові знання співіснують з науковими знаннями.

Особистісні (неявні, приховані) знання – це знання людей, отримані з практики і досвіду.

Формалізовані (явні) знання – знання містяться в документах, в Інтернеті, в базах знань, в експертних системах. Формалізовані знання об’єктивуються знаковими засобами мови, охоплюють ті знання, про які ми знаємо, що їх можна записати, повідомити іншим.

Поверхневі – описують сукупності причинно-наслідкових відносин між окремими поняттями предметної галузі.

Глибинні – відносять абстракції, аналогії, зразки, які відображають глибину розуміння всіх процесів, що відбуваються в предметній галузі. Введення в базу глибинних знань дозволяє зробити систему більш гнучкою та адаптивною, так як глибинні знання є результатом узагальнення проектувальником або експертом первинних примітивних понять.

2.2. Класифікація моделей подання знань

Сукупність знань потрібних для прийняття рішень називають предметною галуззю (ПГ) або **знаннями про предметну галузь**. У будь-якій ПГ є свої поняття і зв’язки між ними, своя термінологія, свої об’єкти, свої закони, що зв’язують між собою об’єкти даних ПГ, свої процеси і події. Крім того, кожна ПГ має свої методи вирішення задач. Вирішуючи задачі такого виду на комп’ютері використовують інформаційні системи, ядром яких є база знань, яка містить основні характеристики ПГ.

База знань – основа будь-якої інтелектуальної системи, де знання описані деякою мовою подання знань, наближеної до природної. Сьогодні знання придбали чисто декларативну форму. Знаннями вважаються речення

(твердження, вислови), записані на мовах подання знань, наближених до природної мови і зрозумілих неспеціалістам.

Бази знань базуються на **моделях подання знань**. При поданні знань в пам'яті інтелектуальної системи традиційні мови, засновані на чисельному поданні даних, є неефективними. На практиці найбільш часто використовується класифікація моделей подання знань, наведена на рис. 2.2, де моделі подання знань діляться на дві категорії: детерміновані і м'які.

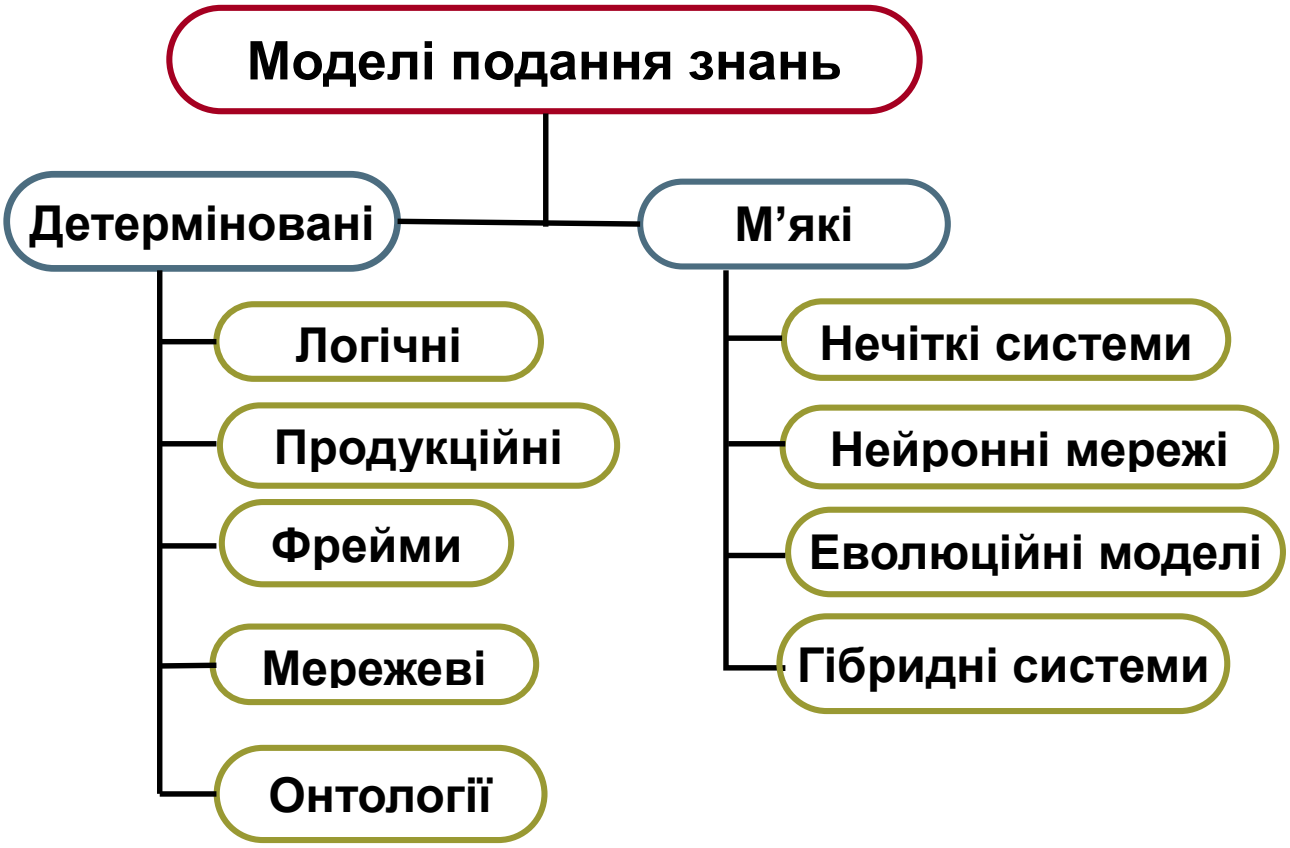


Рисунок 2.2 – Моделі подання знань в ШІ.

Детерміновані означає, що знання описуються чітко визначеною формальною моделлю (логічною, математичною) або структурою. Для цього використовуються спеціальні мови подання знань, засновані на символьному поданні даних і наближені до традиційної природної мови. М'які означає, що знання описуються формальними моделями, які оперують з нечіткими, розмитими, неповними, спотвореними даними.

2.3. Логічні моделі подання знань

Логічні моделі є прикладом формальної символьної системи, яка вислови природною мовою подає у виді логічних виразів (формул) і за допомогою логічний операцій (зв'язок) здійснює їх перетворення. Строго математично –

Логічна модель – це формальна система, що задається четвіркою:

$$S = < T, P, A, R >,$$

де:

T – множина базових елементів, з яких формуються логічні вирази;

P – множина синтаксичних правил, що визначають синтаксично правильні вирази з термінальних елементів *T* логічної моделі *S*;

A – множина аксіом моделі *S*, що відповідають синтаксично правильним виразам, які в рамках цієї моделі *S* апріорно вважаються істинними;

R – множина правил виведення, що дозволяють одержувати з одних синтаксично правильних виразів інші.

Найпростішою логічною моделлю є обчислення висловлювань, яке є одним з початкових розділів математичної логіки, що служить основою для побудови більш складних формалізмів. Логіка висловлювань реалізується за допомогою обчислення **логіки предикатів першого порядку**.

Під обчисленням предикатів розуміється формальна мова опису відносин в деяких предметних галузях. Основна перевага обчислення предикатів – добре зрозумілий механізм математичного доведення, який може бути безпосередньо запрограмований. Предикатом називають висловлювання (вираз), що приймає лише два значення: «істина» або «хибність». Для позначення предикатів застосовуються логічні зв'язки між висловлюваннями:

‘ ¬ ’ – **не**, ‘∧’ – **або**, ‘∧’ – **і**, ‘ ⊃ ’ – **якщо**, а також квантор існування ∃ та квантор загальності ∀.

Таким чином, логіка предикатів оперує логічними зв'язками між висловлюваннями, наприклад, вона вирішує питання: чи можна на основі висловлювання А отримати висловлювання В і т.д.

Допустимим виразом в обчисленні предикатів називаються правильно побудована формула, що складаються з атомарних формул. Атомарні формули складаються з предикатів і термів, що поділяються круглими, квадратними та фігурними дужками.

Теоретичною основою логічних моделей є **методи автоматичного доведення теорем**. Центральним серед сучасних методів автоматичного доведення теорем вважається **метод резолюцій** Робінсона (1965р.) Застосування правила резолюцій виявилося більш ефективним, ніж класичне виведення у математичній логіці на основі modus ponens та ін.

Позитивною характеристикою логічних моделей є однозначність теоретичного обґрунтування і можливість реалізації системи формально точних висловлювань і висновків; недолік – формальний процедурний стиль мислення.

2.4. Продукційні моделі

У ***продукційних моделях*** знання задаються за допомогою правил “if-then” або "*ЯКЩО-ТО*" (явище-реакція). Вони є дуже широко використовуваними, оскільки для них характерні простота модифікації баз знань. З одного боку, їх правила виведення близькі до логічних моделей, а з іншого – вони відображають знання у порівнянні з класичними логічними моделями, більш наочно.

Продукція визначається як четвірка:

$$S = < (i)\ Q;\ P;\ A \Rightarrow B;\ N >,$$

де:

(i) – ім’я продукції;

Q – предметна галузь (сфера застосування);

P – умова застосування продукції;

$A \Rightarrow B$ – ядро продукції (інтерпретується як «якщо маєш *A*, зроби *B*»);

N – післяумова продукції (процедури, які необхідно виконати після реалізації ядра.

Інформаційні системи, побудовані на основі продукційних моделей, називаються ***продукційними системами***. У продукційних системах (рис. 2.3) можна виділити:

базу фактичних знань (робочу пам’ять),

базу знань (набір продукцій, машину виведення)

інтерфейс користувача.

Машина виведення визначає, в якій послідовності слід застосовувати продукції, щоб отримати потрібний результат. Особливо це стосується випадків, коли у деякій ситуації можна застосувати не одну, а декілька продукцій (конфліктний набір). Для керування застосовуються два загальні принципи: *обмеження конфліктного набору та застосування алгоритмів розв’язання конфлікту.*

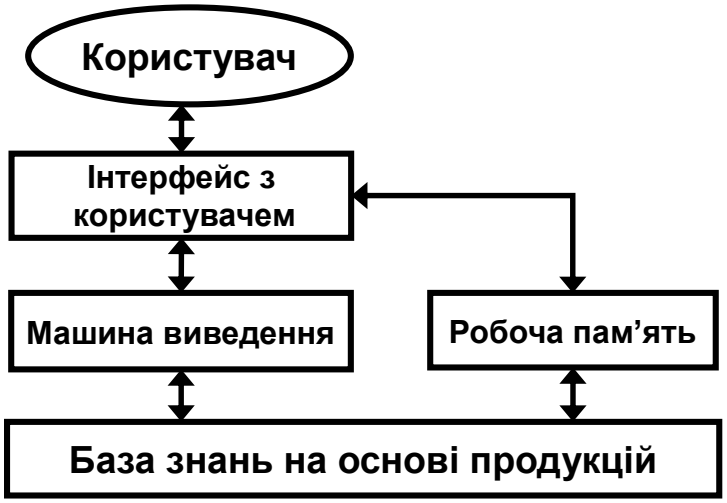


Рисунок 2.3 – Структура продукційної системи

Існують дві основні стратегії логічного виведення у продукційних системах. Стратегія, при якій для отримання висновку проводиться робота по вибору попередньо записаного вмісту робочої пам’яті, застосуванню правил і доповнень даних в пам’яті, називається ***прямим виведенням***. Спосіб, в якому на основі фактів, які потребують підтвердження, щоб виступати в ролі висновку, досліджується можливість застосування правила, яке дає можливість підтвердження, називається ***зворотнім виведенням***.

Пряме виведення передбачає аналіз:

- наслідків з фактів,
- наслідків з наслідків і т. д.

Процес продовжується, поки не буде встановлено істинність або хибність запиту користувача.

Приклад: є продукції A ⇒ B та B ⇒ C; користувач повідомив про істинність факту A і спитав про істинність C. Ланцюжок прямого виведення матиме вигляд: з A виводиться B, з B виводиться C, тому C – істинне.

Зворотне виведення.

При зворотному виведенні логічний блок починає роботу із запиту користувача. Розглядається одне з правил, на основі яких можна вивести це твердження, після чого перевіряється істинність лівої частини цього правила.

Процес повторюється до тих пір, доки він не дійде до фактів, які вважаються істинними.

У нашому прикладі ланцюжок зворотного виведення матиме вигляд: «C виводиться з B, B виводиться з A, A істинне, отже, і C істинне».

Режими керування виконанням активних продукцій:

- *режим негайного виконання;*
- *режим формування конфліктного набору.*

У режимі негайного виконання, перша знайдена продукція виконується невідкладно. У режимі формування конфліктного набору знайдені активні продукції не виконуються відразу, а включаються до конфліктного набору – списку продукцій, готових до виконання. Лише після завершення формування конфліктного набору відбувається вибір зі списку готових продукцій якоїсь однієї за одним з принципів або з стратегій.

Принцип «стосу книг». Найкориснішою є продукція, яка використовується найчастіше. Керуватися принципом «стосу книг» доцільно, якщо продукції відносно незалежні одна від одної, наприклад, коли кожна з них являє собою правило «ситуація A ⇒ дія B».

Принцип «найдовшої умови». Вибирається продукція, для якої стала справедливою найдовша умова виконання.

Приклад: Правило 1 Якщо A – птах, то A літає.

Правило 2 Якщо A – птах і A – пінгвін, то A не літає.

Якщо відомо, що A – пінгвін, то за принципом найдовшої умови слід вибрати (2). Принцип доцільно застосовувати до ситуацій, зв’язаних відношенням «часткове – загальне» та аналізі винятків. Правило (2) є винятком з правила (1).

Принцип метапродукцій. Вводяться правила (метапродукції) використання продукцій.

Принцип вибору за пріоритетом. Кожній продукції надається її пріоритет, який відображає її важливість. Ці пріоритети можуть бути різними для різних ситуацій, а також статичними і динамічними.

Переваги продукційних систем.

- Модульний стиль – простий у додаванні, оновленні та видаленні;
- природні для багатьох проблемних доменів;
- можуть бути представлені невизначені знання.

Недоліки продукційних систем.

- Можуть бути важко зрозумілими;
- можуть мати непередбачувану поведінку;
- додаткові зусилля для структурних знань;
- суперечливість.

2.5. Фрейми

Фреймові моделі базуються на теорії фреймів, розробленій у 1975 р. М. Мінським. Гіпотеза Мінського – знання групуються в фрейми (модулі). Фрейм можна інтуїтивно уявити як каркас, на якому розміщені знання, що описують різноманітні об'єкти, ситуації та поняття і, коли людина потрапляє в нову ситуацію, вона співставляє цю ситуацію з фреймами, які зберігаються в пам'яті. М. Мінський визначав ***фрейм як мінімальний опис деякої сутності, такий, що подальше скорочення цього опису призводить до втрати цієї сутності.***

У найпростішому випадку під фреймом розуміють структуру:

$$\mathbf{<n, (ns_1, vs_1, ps_1), (ns_2, vs_2, ps_2), \dots , (ns_k, vs_k, ps_k)>},$$

де: n – ім'я фрейму, ns_1 – ім'я слота, vs_1 – значення слота, ps_1 – ім'я приєднаної процедури.

Зв'язки між фреймами задаються значеннями спеціального слота з ім'ям «Зв'язок» або АКО (A-Kind-Of). Структура фрейма наведена на рис. 2.4 .

Ім'я фрейму				
Ім'я слоту	Значення слоту	Приєднана процедура	Покажчик успадкування	Покажчик атрибутів
Слот 1				
…….				
Слот n				

Рисунок 2.4 – Структура фрейму

Ім'я слота. Повинно бути унікальним в межах фрейму. Зазвичай ім'я слота є ідентифікатором з певною семантикою. Іменем слота може виступати довільний текст. Наприклад, <Ім'я слота> = Головний герой поеми І.П. Котляревського «Енеїда», «Значення слота> = Еней. Імена системних слотів зазвичай зарезервовані, в різних системах вони можуть мати різні значення. Приклади імен системних слотів: **IS-A, HASPART, RELATIONS** і ін. Системні слоти служать для редагування бази знань і управління виведенням у фреймовій системі.

Значення слоту. Це дані таких типів: **frame** – покажчик на фрейм; **real** – дійсне число; **integer** – ціле число; **boolean** – логічний тип; **text** – фрагмент тексту; **list** – список; **table** – таблиця; **expression** – вираз; **lisp** – зв'язана процедура і т.д

Демони. Демоном називається процедура, що автоматично запускається при виконанні деякої умови. Типи демонів пов'язані з умовою запуску процедури:

IF-NEEDED запускається, якщо в момент звернення до слоту його значення не було встановлено.

IF-ADDED запускається при спробі зміни значення слота.

IF-REMOVED запускається при спробі видалення значення слота. Можливі також інші типи демонів. Демон є різновидом приєднаної процедури.

Приєднана процедура. Як значення слота може використовуватися процедура, яка називається службовою в мові Lisp або методом в мовах об'єктно-орієнтованого програмування. Приєднана процедура запускається за повідомленням, переданим з іншого фрейму.

Процедури є засіб управління виведенням у фреймових системах, причому з їх допомогою можна реалізувати будь-який механізм виведення. Подання таких знань і заповнення ними інтелектуальних систем – дуже нелегка справа, яка вимагає додаткових витрат праці та часу розробників. Тому проектування фреймових систем виконується, як правило, фахівцями, що мають високий рівень кваліфікації в галузі штучного інтелекту.

Покажчики успадкування вказують на інформацію про атрибути слотів з фрейма верхнього рівня, яку успадковують слоти з аналогічними назвами в даному фреймі. Покажчики успадкування характерні для фреймових систем ієрархічного типу, заснованих на відносинах типу «абстрактне – конкретне». У конкретних системах показчики успадкування можуть бути організовані різними способами і мати різні позначення:

U (Unique) – значення слота не успадковується;

S (Same) – значення слота успадковується;

R (Range) – значення слота повинні знаходитися в межах інтервалу значень, зазначених в однойменному слоті батьківського фрейма;

O (Override) – при відсутності значення в поточному слоті воно успадковується з фрейму верхнього рівня, однак в разі визначення значення поточного слота воно може бути унікальним. Цей тип вказівника виконує одночасно функції вказівників **U** і **S**.

Види фреймів:

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

Розрізняють:

- фрейми-зразки, або прототипи;
- фрейми-екземпляри для відображення реальних фактичних ситуацій на основі даних, що надходять.

Розглянемо, наприклад, поняття **"Учень",** яке описується відповідними фреймами (рис. 2.5). Кожний учень, наприклад, може бути охарактеризований:

прізвищем та ім’ям; віком та річчю, яку носить. Ці характеристики у фреймовій моделі представляються **слотами**.

Для фреймових моделей характерна ***ієрархія понять***. У нашому прикладі фрейм "**Учень**" успадковує слоти від більш загального фрейму "**Дитина**", який успадковує слоти від фрейму "**Людина**". Якщо ці фрейми описані і заповнені, то відповідні слоти в описі фрейму "**Учень**" можна не задавати. **Якщо ж якісь слоти уже заповнені конкретними значеннями, то ці значення можуть успадковуватися фреймами-нащадками.**

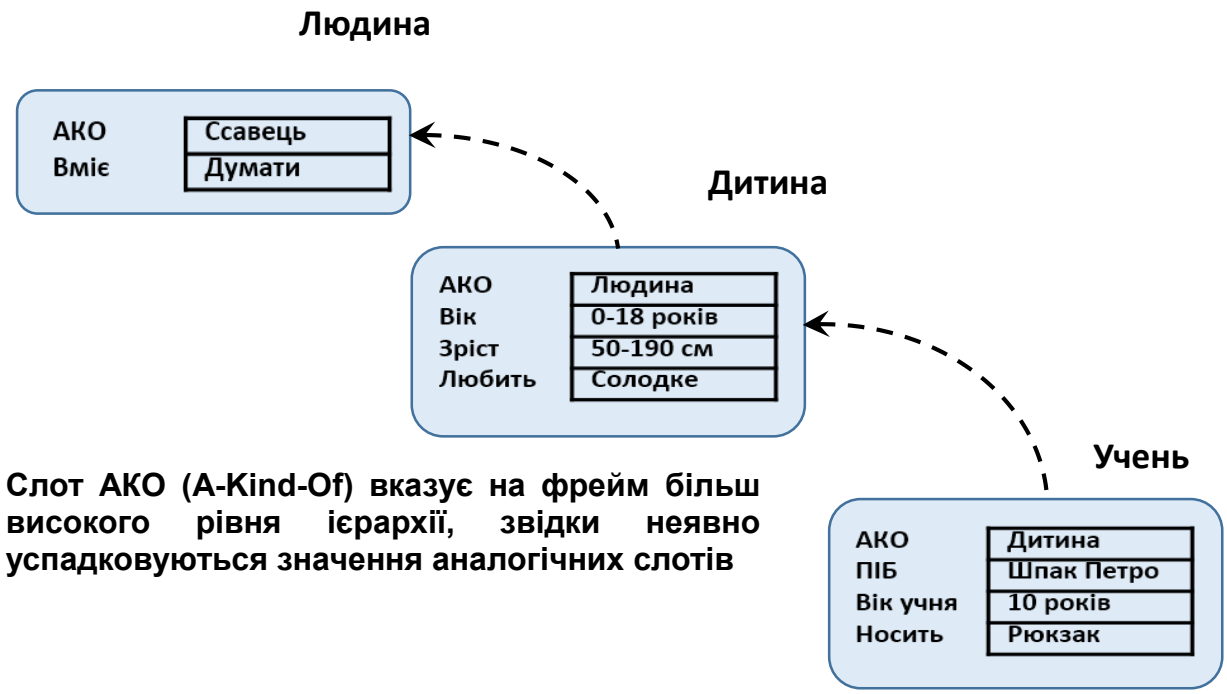


Рисунок 2.5 – Приклад фреймової структури.

2.6. Семантичні мережі

Семантична мережа – це орієнтований граф, вершини якого є поняттями, а дуги – відносинами між ними.

Формальна модель семантичної мережі

$$< I; (C1, C2, ..., Cn); R >,$$

де:

I – множина інформаційних одиниць;

C1, *C2*, ..., *Cn* – типи видносин між інформаційними одиницями;

R – відображення, що задає зв’язки між інформаційними одиницями.

Типи відносин:

- елемент класу (троянда – це квітка);
- атрибутивні зв’язки / мати властивість (пам’ять має властивість – розмір);
- значення властивості (колір має значення жовтий);
- зв’язки типу «частина-ціле» (авто включає кермо);
- функціональні зв’язки (визначені зазвичай дієсловами «виготовляє», «впливає» і ін.);

- кількісні (більше, менше, дорівнює і ін.);
- просторові (далеко від, близько від, за, під, над і ін.);
- часові (раніше, пізніше, протягом і ін.);
- логічні зв'язки (і, або, не) і ін.

Види мереж:

- **Класифікаційні** – відносини ієрархії між інформаційними одиницями;
- **Функціональні** – процедури обчислень одних інформаційних одиниць через інші;
- **Сценарії** – відносини типу «причина-наслідок», «дія», «засіб дії» та ін.

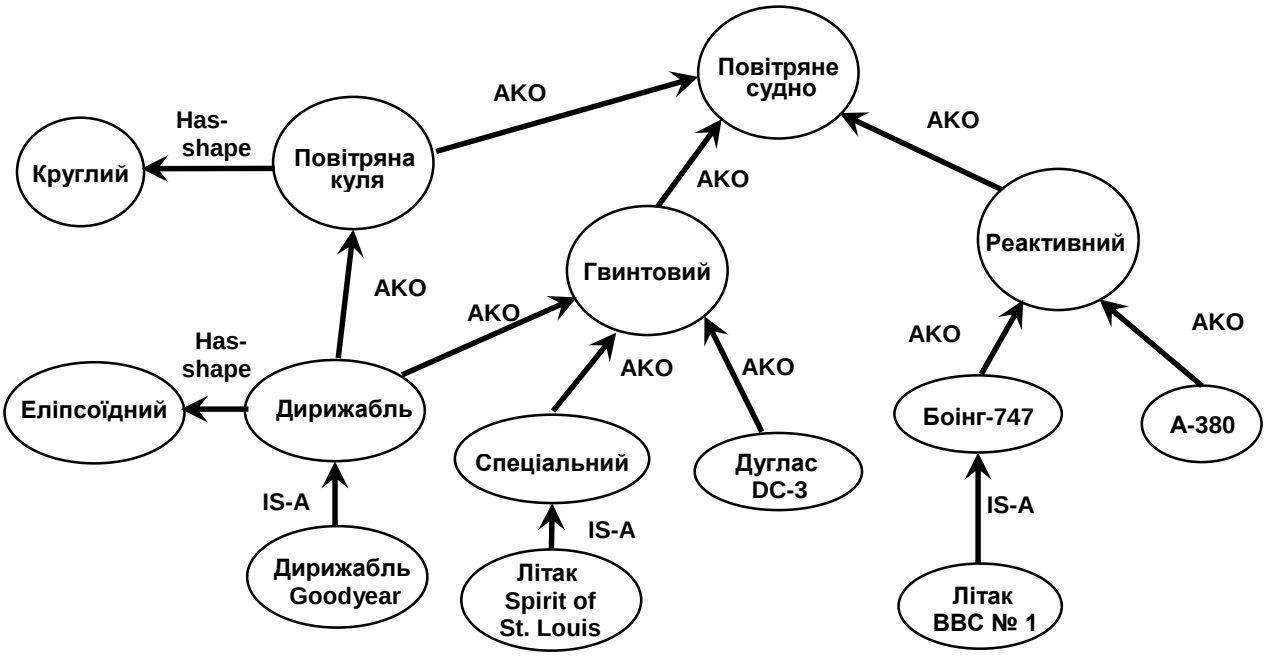


Рисунок 2.6 – Семантична мережа з відносинами IS-A та A-KIND-OF (AKO)

Широко застосовуються трійки ***"об'єкт-атрибут-значення"*** (object-attribute-value – OAV). Приклад трійки наведено в табл. 2.1

Таблиця 2.1 Трійки

Об'єкт	Атрибут	Значення

Виведення в семантичних мережах може виконуватися на основі алгоритмів співставлення, шляхом пошуку підграфів з визначеними властивостями на мережі.

До переваг семантичних мереж відносять: гнучкість; легкість розуміння; підтримка спадщини; «природний» спосіб подання знання; схожість структури мережі до семантичної структури фраз природної мови. Недоліками є: складність роботи з винятками і процедурними знаннями; немає стандартів для визначення вузлів та відносин.

2.7. Онтології.

Термін "онтологія" має два значення.

• **Означення 1. Онтологія** – розділ філософії, що вивчає поняття: існування, буття, становлення та реальність. Він включає питання про те, як сутності групуються в основні категорії і які з цих об'єктів існують на найбільш фундаментальному рівні.

• **Означення 2. Онтологія** – це артефакт, структура, яка описує значення елементів певної системи.

Для ШІ основним є друге визначення. Неформально онтологія є деяким описом погляду на світ стосовно конкретної предметної галузі. Цей опис складається з термінів та правил використання цих термінів у межах конкретної галузі.

На формальному рівні онтологію можна представити як упорядковану трійцю скінченних множин:

$$\boldsymbol{O = < C, R, F >},$$

де:

C – концепти (поняття, терміни) предметної галузі, яку описує онтологія,

R – зв'язки (відносини) між концептами предметної галузі,

F – функції (аксіоми, процедури), що інтерпретують поняття та відносини, встановлюють семантичні обмеження між ними.

Іншими словами, онтологія – це система, що складається з набору понять та набору тверджень про ці поняття, на основі яких можна описувати класи, відносини, функції та індивіди.

Означення Тома Грубера: онтологія – це точна специфікація концептуалізації. Концептуалізація – це структура реальності, що розглядається незалежно від словника предметної галузі та конкретної ситуації. Наприклад, якщо ми розглядаємо просту предметну галузь, що описує кубики на столі, то концептуалізацією є набір можливих положень кубиків, а не конкретне розташування в поточний момент часу. Пізнішою модифікацією визначення Грубера є таке означення: онтологія – це формальна специфікація узгодженої концептуалізації. Під узгодженою концептуалізацією мається на увазі, що дана концептуалізація не є приватною думкою, а є спільною для деякої групи людей.

Онтологія – це формальна теорія, що обмежує можливі концептуалізації світу.

Деякі означення відображають способи, якими автори будують та використовують онтології, наприклад, онтологія – це ієрархічно структурована множина термінів, що описують предметну галузь, яка може бути використана як вхідна структура для бази знань.

Узагальнюючи, визначимо, що онтологію предметної галузі є:

1) ієрархічна структура скінченної множини понять, що описують задану предметну галузь;

2) структура є онтографом, вершинами якого є поняття, а дугами – семантичні відносини між ними;

3) поняття і відносини інтерпретуються відповідно до загальнозначущих функцій інтерпретації, взятих з електронних джерел знань заданої ПГ;

4) визначення понять і відносин виконується на основі аксіом і обмежень їх області дії;

5) формально онтограф описується однією з мов опису онтологій;

6) функції інтерпретації та аксіоми описані в деякій формальній теорії.

Побудова онтології часто не є сама по собі кінцевою метою, зазвичай онтології далі використовуються іншими програмами для вирішення практичних цілей.

Причини для створення онтології :

- Для спільного використання людьми чи програмними агентами загального розуміння структури інформації.
- Для можливості повторного використання знань у предметній галузі.
- Щоб зробити припущення в предметної області явними.
- Для відокремлення знань у предметній галузі від оперативних знань.
- Для аналізу знань у предметній галузі.

Класифікація онтологій в задачах ШІ

Онтології розробниками класифікуються на два типи:

- за метою створення (рис. 2.7);
- за змістом (рис. 2.8);

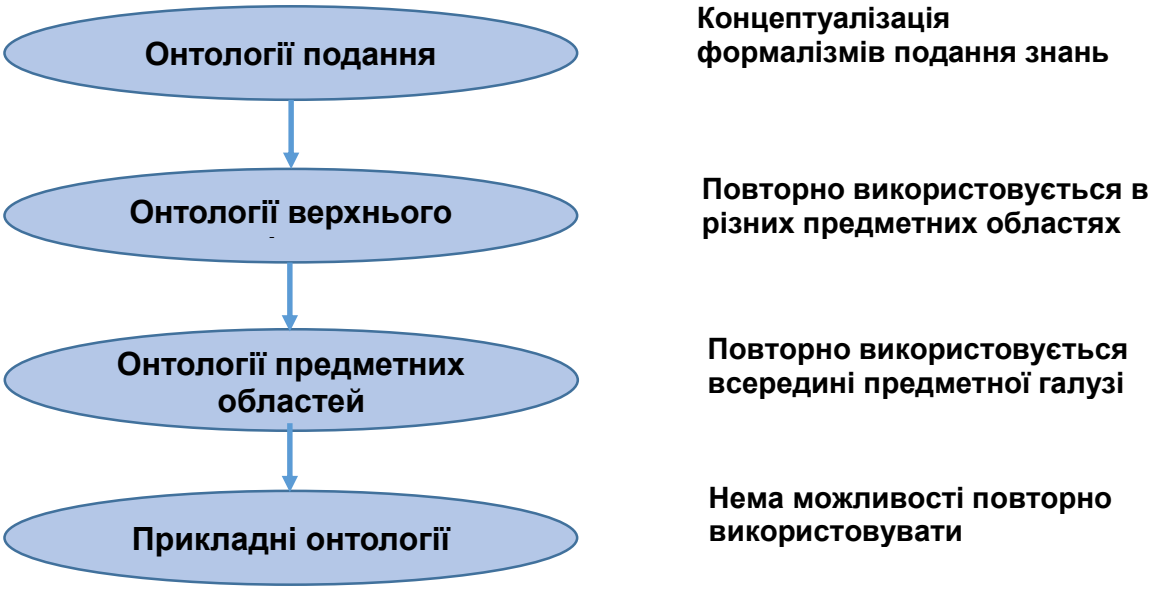


Рисунок 2.7 – Класифікація онтологій за метою створення

Онтологія подання.

Мета онтології – описати галузь подання знань, створити мову для специфікації інших онтологій нижчих рівнів. У цьому описі визначаються такі поняття, як "клас", "відносини", "обмеження значення властивості", "домен", "діапазон" і т.п.

Онтологія верхнього рівня

Призначення – створення єдиної "правильної" онтології, що фіксує знання, загальні для кількох предметних галузей, та у багаторазовому використанні даної онтології. Існує кілька великих онтологій верхнього рівня: Cyc, DOLCE, SUMO, онтологія Джона Сови (J. Sowa) та інші. Але загалом спроби створити онтологію верхнього рівня на всі випадки життя поки що не призвели до очікуваних результатів. Багато онтологій верхнього рівня схожі одна на одну. Вони містять однакові концепти: сутність, явище, процес, об'єкт, роль, простір, час, матерія, подія, дія тощо.

На рис. 2.9 Подано приклад онтології верхнього рівня.

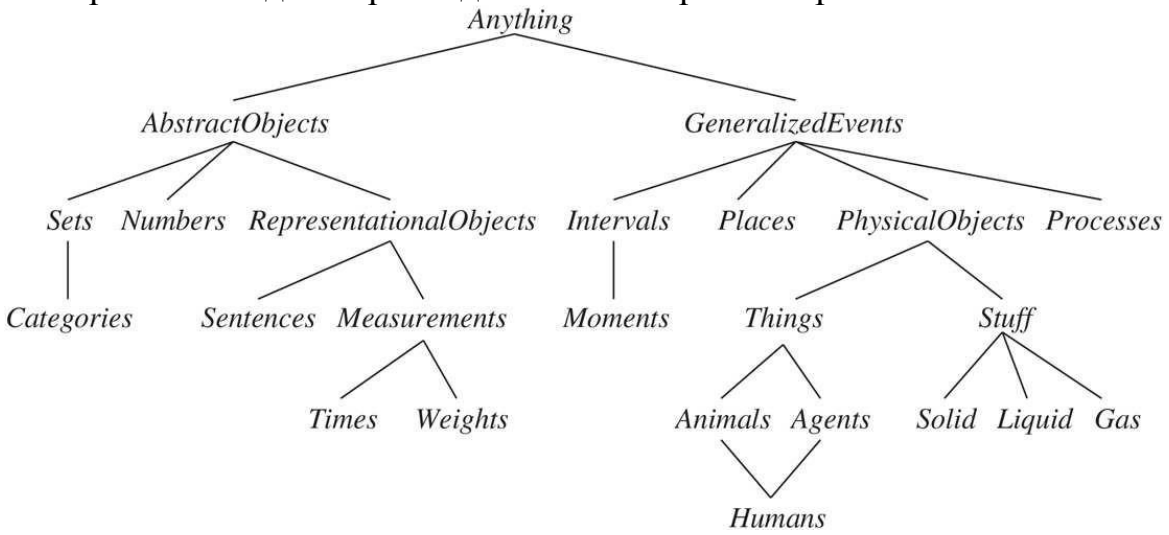


Рисунок 2.9 – Top-Level Ontology (Онтологія верхнього рівня)

Онтологія предметної галузі

Інша назва – онтологія домену. Галузь інтересу обмежена однією ПГ, наприклад, авіація, медицина, культура, дистанційне навчання, Інтернет-технології. Онтологія ПГ узагальнює поняття, які у деяких задачах домену абстрагуються від самих задач (так, онтологія автомобілів незалежна від конкретних марок машин). Різновидом онтологій є словники та тезауруси, розміщені в глобальній мережі і до яких можна звертатись із додатка з метою використання для конкретної задачі ШІ. Наприклад, у галузі медицини створені великі стандартні, структуровані словники, такі як SNOMED CT (Systematized Nomenclature of Medicine – Clinical Terms – систематизована номенклатура медицини – клінічна термінологія).

Прикладна онтологія

Призначення онтології – описати концептуальну модель конкретної задачі або програми. Прикладні онтології описують концепти, які залежать як від онтології задач, і від онтології предметної галузі. Прикладом може бути онтологія для будівельних матеріалів, обчислювальної техніки, текстів. Такі онтології містять специфічну інформацію. Приклад – онтологія Plinius. Метою проекту Plinus є напівавтоматичне вилучення знань із текстів природною мовою, зокрема, літератури про механічні властивості керамічних матеріалів. Так як тексти охоплюють широкий діапазон понять, потрібно багато інтегрованих

онтології для охоплення таких понять, як керамічні матеріали та їх властивості, способи їх обробки, різні дефекти матеріалів і т.д.

Класифікація онтологій за змістом

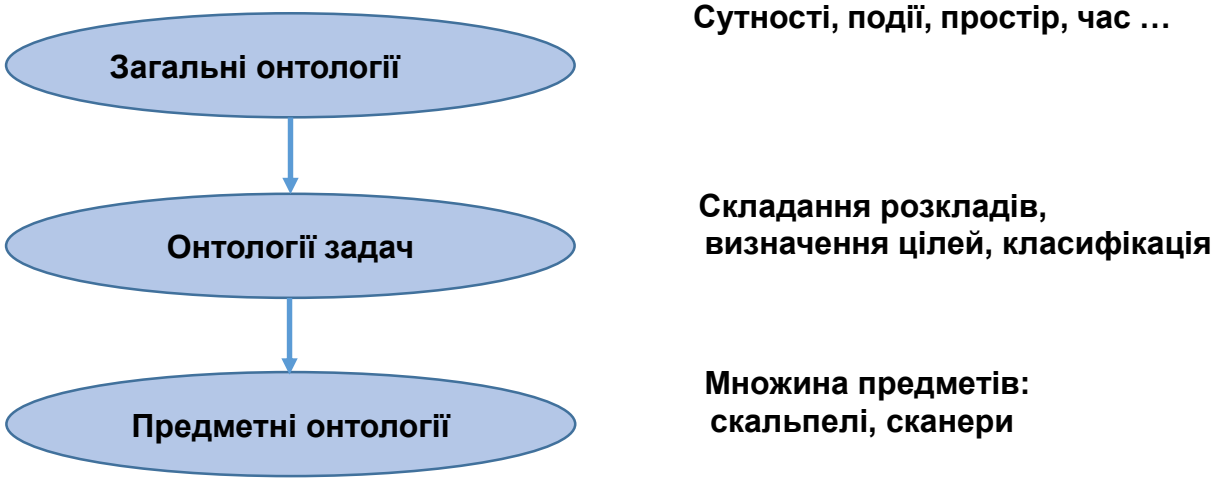


Рисунок 2.8 – Класифікація онтологій за змістом

Загальні онтології описують найбільш загальні концепти (простір, час, матерія, об'єкт, подія, дія, тощо), які незалежні від конкретної проблеми чи галузі. До цієї категорії потрапляють і онтології подання, і онтології верхнього рівня.

Онтологія, орієнтована на задачу – це онтологія, використовувана конкретної прикладної програмою і містить терміни, що використовуються при розробці ПЗ, що виконує конкретну задачу. Вона відображає специфіку програми, але може також містити деякі загальні терміни. Задачі, для яких створюються онтології, можуть бути найрізноманітнішими: складання розкладу, визначення цілей, діагностика, продаж, розробка ПЗ, побудова класифікації. У цьому онтологія завдання використовує спеціалізацію термінів, які у онтологіях верхнього рівня (загальних онтологіях).

Предметна онтологія (або онтологія предметів) визначає реальні предмети у будь-якій діяльності (виробництві). Наприклад, це може бути онтологія всіх частин і компонентів літаків певної марки (Boeing) і відомості про їх постачальників, характеристики, спосіб з’єднання один з одним і т.п.

Кожна ланка вказує на те, що нижнє поняття є спеціалізацією верхнього. Спеціалізації не обов’язково розрізнені – людина є і твариною, і агентом.

Галузі використання онтологій:

- машинний переклад;
- системи питання-відповідь;
- інформаційний пошук;
- системи видобування знань;
- системи ведення діалогу між комп’ютером і людиною;
- системи розуміння мови (автоматичне реферування тексту і рубрикація і ін.)

Semantic WEB

Ідея Семантичної Мережі (Semantic Web) вперше була проголошена в 2001 Тімом Бернерсом-Лі (творцем World Wide Web). Суть її полягає в автоматизації "інтелектуальних" задач обробки змісту (в семантичному сенсі) тих чи інших ресурсів, що є в Мережі. Обробкою та обміном інформації повинні займатися спеціальні інтелектуальні агенти (програми, розміщені в Мережі). Але для того, щоб взаємодіяти між собою, агенти повинні мати загальне (розділене всіма) формальне подання змісту для будь-якого ресурсу. Саме з метою подання загальної, явної і формальної специфікації змісту в Semantic Web використовуються онтології.

Незважаючи на окремі успіхи, досі не можна сказати, що ідея Semantic Web реалізована на практиці.

Робота над засобами опису семантики в Мережі розпочалася задовго до публікації 2001 року. В 1997 World Wide Web Consortium (W3C) визначив специфікацію RDF (Resource Description Framework). RDF надає просту, але потужну мову опису ресурсів, засновану на триплетах (triple-based) "Суб'єкт-Предикат-Об'єкт" та специфікації URI. У 1999 році RDF набуває статусу рекомендації. Концептуально RDF дає мінімальний рівень для подання знань у Мережі.

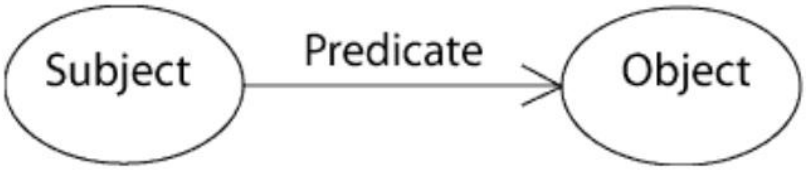


Рисунок 2.10 – Триплет Суб'єкт-Предикат-Об'єкт

Крім RDF було розроблено мову опису структурованих словників для RDF – RDF Schema (RDFS). Він надає мінімальний набір засобів специфікації онтологій. RDFS отримав статус рекомендації W3C у 2004 році. Однак перешкодою для Semantic Web стало те, що документів, написаних мовою RDF/RDFS, було мало. У період з 2001 по 2004 роки йшла інтенсивна робота щодо створення програмних засобів для обробки та автоматичної генерації RDF-документів.

Вираз RDF має структуру простого речення, з єдиною відмінністю – всі слова є URI. Кожне висловлювання RDF являє собою трійку (S, P, O), де S – суб'єкт, P – предикат і O – об'єкт. Вирази RDF дозволяють стверджувати будь-яку інформацію про елементи Semantic Web і робити висновки.

Потім стала розвиватися мова RDF Schema (RDFS) – мова опису словників RDF термінів. RDF Schema визначає класи, властивості та інші ресурси. Таким чином, RDFS став семантичним розширенням RDF. RDF і RDFS дозволяють працювати з метаданими, забезпечувати комп'ютер семантичною інформацією і обробляти цю інформацію автоматично. Однак, використовуючи RDF, «хто завгодно (тобто будь-який користувач RDF) може сказати що завгодно (тобто фіксувати довільне твердження) про що завгодно (тобто про будь-який ресурс Мережі)». Крім того, «RDF не забороняє робити безглуздих тверджень або тверджень які не узгоджуються з іншими. Отже, немає ніякої гарантії цілісності

і несуперечності RDF-описів », тому при використанні інформації користувачі повинні самі перевіряти цю цілісність.

У 2004 році статус рекомендації отримала мова OWL (Web Ontology Language). Вона має 3 діалекти (3 множини структурних одиниць), що використовуються в залежності від необхідної виразної потужності (рис. 2.11).

OWL фактично є надбудовою над RDF/RDFS і підтримує ефективне представлення онтологій у термінах класів та властивостей, забезпечення простих логічних перевірок цілісності онтології та зв'язування онтологій між собою (імпорт зовнішніх визначень). Багато формалізмів опису знань можуть бути відображені на формалізм OWL (рис. 2.11). Два з її діалектів – OWL Lite та OWL DL – відповідають двом дескриптивним логікам, що мають різну виразну силу. Велика кількість створюваних нині онтологій кодуються на OWL; вже існуючі онтології транслюються до неї.

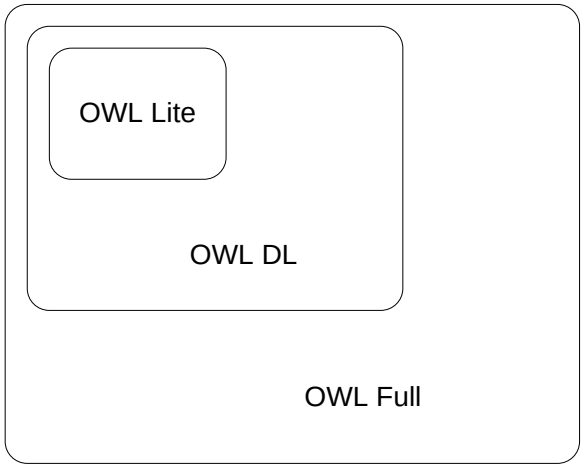


Рисунок 2.11 – Ієрархія видів OWL

OWL Lite підтримує тих користувачів, які мають потребу передусім в класифікаційній ієрархії і простих обмеженнях. Для розробників має бути простіше у своїх продуктах забезпечити підтримку OWL Lite, ніж його виразніших побратимів, так, зокрема, OWL Lite дозволяє швидку міграцію тезаурусів та інших таксономій.

OWL DL призначена для користувачів, які хочуть максимальної виразності без втрати повноти обчислень (всі висновки гарантовано будуть обчислюваними). OWL DL охоплює усі мовні конструкції OWL з обмеженнями. OWL DL через його відповідність дескриптивній логікі, дисципліні. OWL DL спроектована, щоб підтримати певний сегмент бізнесу, що займається дескриптивною логікою, і мати бажані обчислювальні властивості для систем штучного інтелекту.

OWL Full призначається для користувачів, які хочуть мати максимальну виразність і синтаксичну свободу RDF без обчислювальних гарантій. OWL Full уможливлює такі онтології, які розширюють склад зумовленого (RDF або OWL) словника.

Розроблена мова SPARQL – мова запитів до RDF-сховищ. У січні 2008 року вона набула статусу офіційної за рекомендаціями Консорціуму W3C.

Синтаксично вона дуже нагадує SQL. На рис. 2.12 представлена діаграма, звана іноді стеком (або навіть "шаровим пирогом") Semantic Web.

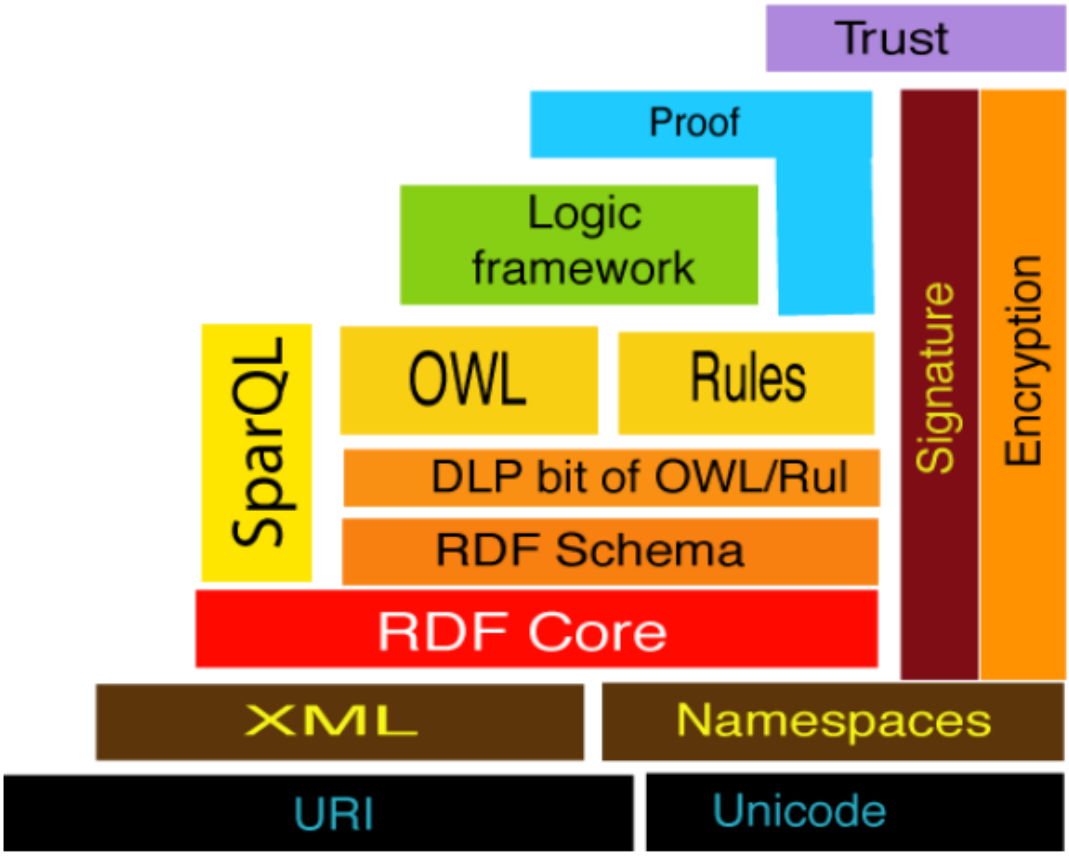


Рисунок 2.12 – Структура Semantic WEB

Інструментальні засоби створення онтологій

Protege

Одним з популярних редакторів онтологій є вільно поширюваний редактор Protege, розроблений в Стенфордському університеті (США). Спочатку єдиною моделлю знань (формалізмом), підтримуваної Protege, була фреймова модель, однак згодом була додана підтримка мови OWL2. Версія редактора Protege-OWL, на даний момент стала основною.

Редактор Protege-Frames дозволяє користувачам будувати і заповнювати онтології, засновані на фреймах, відповідно до протоколу OKBC3, а форматом подання є мова KIF. Основними елементами онтології в Protege-Frames є класи (classes), екземпляри (instances), слоти (slots), що представляють властивості класів і екземплярів, і фасети (facets), що задають додаткову інформацію про слоти.

Protege має відкриту, легко розширювану архітектуру і підтримує модулі розширення функціональності (plug-ins). Зокрема, модуль розширення CLIPSTab дозволяє здійснювати інтеграцію з мовою розробки експертних систем CLIPS (C Language Integrated Production System). Нарешті, для Protege розроблена концепція спільної роботи декількох користувачів (Collaborative Protege), втілена в онлайн-версії даного редактора.

DOE (Differential Ontology Editor) – простий редактор, який дозволяє користувачеві створювати онтології. Процес специфікації онтології складається із трьох етапів.

- На першому етапі користувач будує таксономію понять та відносин, явно окреслюючи позицію кожного елемента (поняття) в ієрархії. Потім користувач вказує, у чому специфіка поняття щодо його "батька", і в чому це поняття подібне або відмінне від його "братів". Користувач може також додати синоніми та енциклопедичне визначення кількома мовами для всіх понять.
- На другому етапі дві таксономії розглядаються з різних точок зору. Користувач може розширити їх новими об'єктами або додати обмеження щодо відносин.
- На третьому етапі онтологія може бути перекладена мовою подання знань.

OntoEdit – інструментальний засіб, який забезпечує перегляд, перевірку та модифікацію онтології. Воно підтримує мови представлення онтології OIL і RDFS, а також внутрішню мову представлення знань OXML, засновану на XML. Як і Protege, це автономний Java-додаток, але його коди закриті. Версія OntoEdit Free, що вільно розповсюджується, обмежена 50 концептами, 50 відносинами і 50 примірниками.

OilEd – автономний графічний редактор онтологій, розроблений у рамках проекту On-To-Knowledge. Він вільно розповсюджується за загальнодоступною ліцензією GPL. Інструмент використовує для представлення онтологій мову OIL. В OilEd відсутня підтримка екземплярів класів.

WebOnto є Java-аплетом і розроблений для перегляду, створення та редагування онтологій. Для моделювання онтологій він використовує мову OCML (Operational Conceptual Modeling Language). Користувач може створювати різні структури, у тому числі класи з численним наслідуванням. Інструмент має низку корисних особливостей: перегляд відносин, класів та правил, можлива спільна робота над онтологією кількох користувачів.

ODE, WebODE

ODE (Ontological Design Environment) взаємодіє з користувачами на концептуальному рівні, забезпечує їх набором таблиць для заповнення (концептів, атрибутів, відносин) та автоматично генерує код мовами LOOM, Ontolingua та F-Logic. Цей інструмент отримав свій розвиток у редакторі онтологій WebODE, який інтегрує всі сервіси ODE у єдину архітектуру, зберігаючи свої онтології у реляційній БД.

3. СИСТЕМИ НЕЧІТКОЇ ЛОГІКИ

3.1. Виникнення нечіткої логіки

Lotfi Zadeh
1921-2017

Датою появи нечіткої логіки вважається 1965 рік, коли в журналі "Information and Control" була надрукована стаття Lotfi Zadeh "Fuzzy Sets". Концепція нечіткої множини зародилася у Заде «… як незадоволеність математичними методами класичної теорії систем, що змушувала домагатися штучної точності, недоречної в багатьох системах реального світу, особливо в так званих гуманістичних системах, що включають людей». Наприклад, якщо стандартом точного визначення сприйняття людиною холоду буде «температура нижче +15°С», то +14,99°С буде розцінюватися як холод, а +15°С – як комфортно. В реальному житті сприйняття температури є розмитим. Холодом може вважатись і 16 °С, а комфортним – 14 °С (рис 3.1), тому прикметник «fuzzy» (нечіткий, розмитий) введено з метою відокремлення від традиційної чіткої математики й Аристотелевої логіки, що оперують з чіткими поняттями: «належить-не належить», «істина-хибність». Нечітка логіка, як і продукційні системи, оперує з правилами «Якщо → То», але факти в умові правила і дії в наслідку подаються іншим способом.

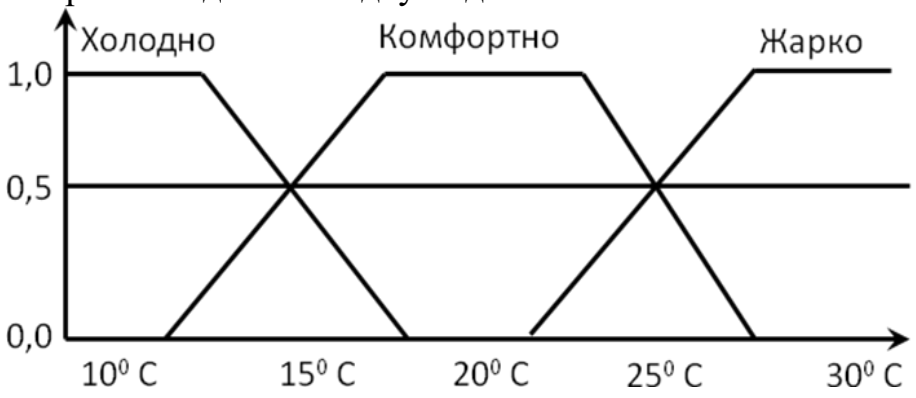


Рисунок 3.1 – Нечітке визначення температури, що відчувається людиною

Етапи розвитку нечіткої логіки.

1975 р. Мамдані і Ассіліан (Mamdani and Assilian) побудували перший нечіткий контролер для управління простим паровим двигуном.

1982 р. Холмблад і Остергад (Holmblad and Osregaad) розробили перший промисловий нечіткий контролер, який був впроваджений в управління процесом випалу цементу на заводі в Данії.

1992 р. Ванг (L.Wang) показав, що нечітка система може апроксимувати будь-яку неперервну функцію з довільною точністю, якщо використовує набір з n ($n\to\infty$) правил виду «Якщо → То», гаусові функції належності, композиції у вигляді добутку, імплікації у формі Ларсена та центроїдний метод приведення до чіткості.

1993 р. Коско (Kosko) теорема про нечітку апроксимацію Fuzzy Approximation Theorem (FAT): будь-яка математична система може бути апроксимована системою нечіткою логіки.

Отже, за допомогою природномовних висловлювань «Якщо А то В», з подальшою їх формалізацією засобами теорії нечітких множин, можна скільки завгодно точно відобразити довільний зв'язок «входи-вихід» без використання апарату диференціального й інтегрального числення, традиційно застосовуваного в управлінні та ідентифікації.

Для створення дійсно інтелектуальних систем, здатних адекватно взаємодіяти з людиною, необхідний був новий математичний апарат, що переводить невиразні і неоднозначні життєві твердження в мову чітких і формальних математичних формул.

Апарат теорії нечітких множин, продемонструвавши ряд багатообіцяючих можливостей застосування – від систем керування літальними апаратами до прогнозування підсумків виборів, виявився разом з тим надмірно складним для втілення, враховуючи наявний на той час рівень технології, і на багато років нечітка логіка зайняла своє місце в ряді інших спеціальних наукових дисциплін – десь посередині між експертними системами і нейронними мережами.

Своє друге народження теорія нечіткої логіки пережила на початку вісімдесятих років минулого століття, коли відразу кілька груп дослідників (в основному в США і Японії) всерйоз зайнялися створенням електронних систем різного застосування, що використовують нечіткі керуючі алгоритми.

Третій період почався з кінця 80-х років і триває дотепер. Цей період характеризується бумом практичного застосування теорії нечіткої логіки в різних сферах науки і техніки. До 90-го року минулого століття з'явилося близько 40 патентів, що відносились до нечіткої логіки (30 японських). Сорок вісім японських компаній утворили спільну лабораторію LIFE (Laboratory for International Fuzzy Engineering), японський уряд фінансував 5річну програму з нечіткої логіки, що включає 19 різних проектів – від систем оцінки глобального забруднення атмосфери і передбачення землетрусів до АСУ заводських цехів і складів. Результатом виконання цієї програми з'явилася поява цілого ряду нових масових мікрочіпів, заснованих на нечіткій логіці. Сьогодні їх можна знайти в пральних машинах і відеокамерах, цехах заводів і моторних відсіків автомобілів, у системах керування складськими роботами і бойовими вертольотами.

У США розвиток нечіткої логіки йде шляхом створення систем, що потрібні великому бізнесу і військовим. Нечітка логіка застосовується при аналізі нових ринків, біржовій грі, оцінці політичних рейтингів, виборі оптимальної цінової стратегії та ін. З використанням теорії нечітких множин вирішуються питання узгодження суперечливих критеріїв прийняття рішень, створення логічних регуляторів систем. Нечіткі множини дають змогу застосовувати лінгвістичний опис складних процесів, встановлювати нечіткі відносини між поняттями, прогнозувати поведінку системи, формувати множину альтернативних дій, виконувати формальний опис нечітких правил прийняття рішень.

Перспектива застосування нечіткої логіки полягає у розробленні гібридних методів штучного інтелекту, до яких можна віднести нечіткі штучні нейронні мережі, адаптивне поповнення баз нечітких правил, підтримка нечітких запитів до баз даних, побудова нечітких когнітивних карт, нечіткі графи, нечіткі мережі Петрі, нечіткі дерева прийняття рішень, нечітка кластеризація та ін.

3.2. Означення понять нечіткої логіки

Розглянемо основні поняття нечітких систем, залучаючи тільки найосновніші математичні поняття.

Універсум (універсальна множина) – звичайна множина, що містить в рамках деякого контексту всі можливі елементи або всі множини, що розглядаються, які є її підмножинами, наприклад, множина усіх цілих чисел, множина всіх гладких функцій і т.д. Позначимо універсум через X. Окремий елемент X будемо позначати через x.

Означення 1. **Нечітка множина (fuzzy set)** – це сукупність елементів довільної природи, відносно яких не можна з повною визначеністю стверджувати, належить той чи інший елемент сукупності даній множині, чи ні.

Формальне означення:

Нечіткою множиною A на універсальній множині X називається сукупність пар $\{\mu_A(x), x\}$, де $\mu_A(x)$ ступінь належності елемента $x \in X$ нечіткій множині A.

Ступінь належності μ_A називається функцією належності (характеристичною функцією), значення якої вказують, чи є x елементом множини А. Як правило, μ_A визначається в діапазоні [0, 1]. Для універсуму функція належності дорівнює одиниці для всіх елементів множини.

Означення 2. Нечітка змінна це трійка:

$$<\alpha, X, A>,$$

де:

α – найменування нечіткої змінної;

X – область визначення нечіткої змінної;

A – нечітка множина на X, $A=\{x, \mu_A(x)\}$

Якщо універсальна множина є скінченною $X=\{x_1, x_2, \ldots, x_k\}$, тоді нечітка множина A записується так:

$$A = \sum_{i=1}^{k} \frac{\mu_A(x_i)}{x_i} \qquad A = \left(\frac{\mu_A(x_1)}{x_1}, \frac{\mu_A(x_2)}{x_2}, \ldots, \frac{\mu_A(x_k)}{x_k} \right)$$

Знак $\sum$ в даному випадку не означає операцію підсумування.

У разі неперервної множини X використовують таке позначення:

$$A = \int_{x \in X} \frac{\mu_A(x)}{x}$$

Знак $\int$ в даному випадку не означає операцію інтегрування.

На рис. 3.2 представлені неперервні і дискретні нечіткі множини для поняття «Приблизно нуль».

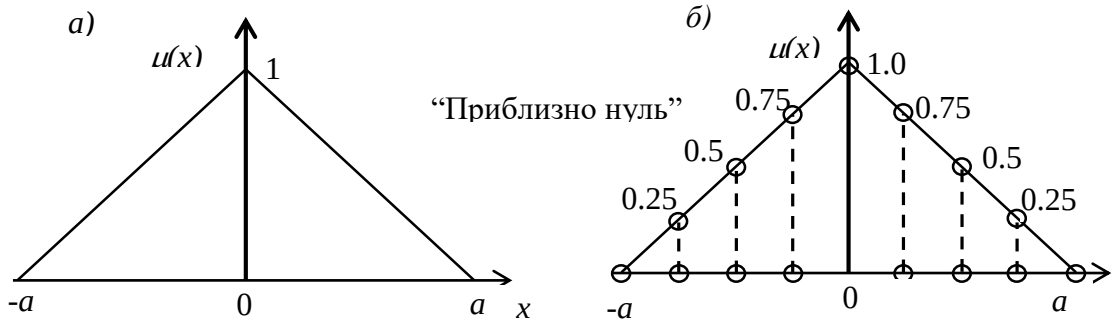


Рисунок 3.2 – Графічне подання нечіткої множини.
а) неперервна функція належності б) дискретна функція належності

Для зменшення суб'єктивності при визначенні функції належності використовуються різні способи обробки думок експертів. В табл. 1, 2 і на рис. 3.3 наведений приклад побудови функцій належності (термів) для поняття "Зріст людини чоловічої статі".

Таблиця 1 – Результати опитування думок експертів

	Терм	[160, 165)	[165, 170)	[170, 175)	[175, 180)	[180, 185)	[185, 190)	[190, 195)	[195, 200)
Експерт 1	Низький	1	1	1	0	0	0	0	0
	Середній	0	0	1	1	1	0	0	0
	Високий	0	0	0	0	0	1	1	1
Експерт 2	Низький	1	1	1	0	0	0	0	0
	Середній	0	0	1	1	0	0	0	0
	Високий	0	0	0	0	1	1	1	1
Експерт 3	Низький	1	0	0	0	0	0	0	0
	Середній	0	1	1	1	1	1	0	0
	Високий	0	0	0	0	0	1	1	1
Експерт 4	Низький	1	1	1	0	0	0	0	0
	Середній	0	0	0	1	1	1	0	0
	Високий	0	0	0	0	0	0	1	1
Експерт 5	Низький	1	1	0	0	0	0	0	0
	Середній	0	1	1	1	0	0	0	0
	Високий	0	0	0	1	1	1	1	1

Таблиця 2 – Усереднені результати опитування думок експертів

Терм	[160, 165)	[165, 170)	[170, 175)	[175, 180)	[180, 185)	[185, 190)	[190, 195)	[195, 200)
Низький (Low)	5	4	3	0	0	0	0	0
	1	0.8	0.6	0	0	0	0	0
Середній (Middle)	0	2	4	5	3	2	0	0
	0	0.4	0.8	1	0.6	0.4	0	0
Високий (High)	0	0	0	1	2	4	5	5
	0	0	0	0.2	0.4	0.8	1	1

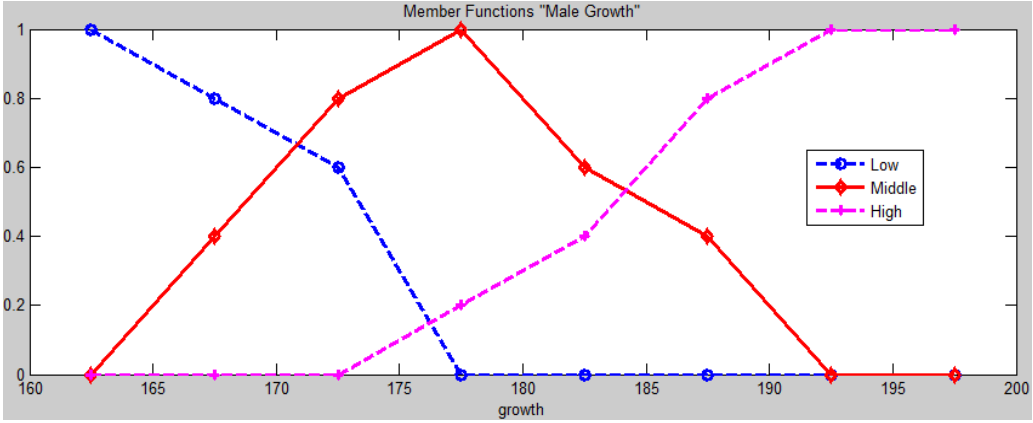


Рисунок 3.3 – Функції належності для поняття “Зріст людини чоловічої статі”

Формальне визначення нечіткої множини не накладає ніяких обмежень на вибір конкретної функції належності для її подання. Однак на практиці зручно використовувати ті з них, які допускають аналітичне подання у вигляді деякої простої математичної функції. Це спрощує чисельні розрахунки і скорочує обчислювальні ресурси при реалізації. Необхідність типізації окремих функцій належності також обумовлена наявністю їх реалізацій в інструментальних засобах. Визначення функції належності для практичної задачі є не однозначною. Функцію належності визначає розробник, тому вона несе на собі суб’єктивний відтінок (у кожного розробника своє уявлення про задачу).

Параметри функції належності

Висота нечіткої множини – максимальне значення функції належності:

$$h_A = \max_{x \in X} \mu_A(x) \quad \text{або} \quad h_A = \sup_{x \in X} \mu_A(x)$$

Нечітку множину називають **нормальною**, якщо її висота дорівнює 1:

$$h_A = \sup_{x \in X} \mu_A(x) = 1$$

Нечітку множину називають **субнормальною**, якщо:

$$h_A \neq 1$$

Нечітку множину називають **унімодальною**, якщо $\mu_A(x)=1$ лише для одного елемента з A.

Нечітка множина є **порожньою**, якщо:

$$\forall x \in X \qquad \mu_A(x) = 0$$

Не порожню субнормальну множину можна нормалізувати за формулою:

$$\mu_A(x) = \frac{\mu_A(x)}{\sup_{x \in X} \mu_A(x)}$$

Параметри функції належності наведені на рис. 3.4.

Носієм нечіткої множини A є підмножина зі значеннями $\mu_A(x) > 0$:

$$A_a = \{x | \mu_A(x) > a\} \quad \forall x \in X$$

Елементи $x \in X$, для яких $\mu_A(x)$ дорівнює постійному значенню a (***a*-рівень**) називаються точками зрізу множини A або ***a*-зрізом**.

Ядром нечіткої множини A є підмножина зі значеннями $\mu_A(x) = 1$:

$$A_n = \{x|\mu_A(x)>a\} \quad \forall x \in X$$

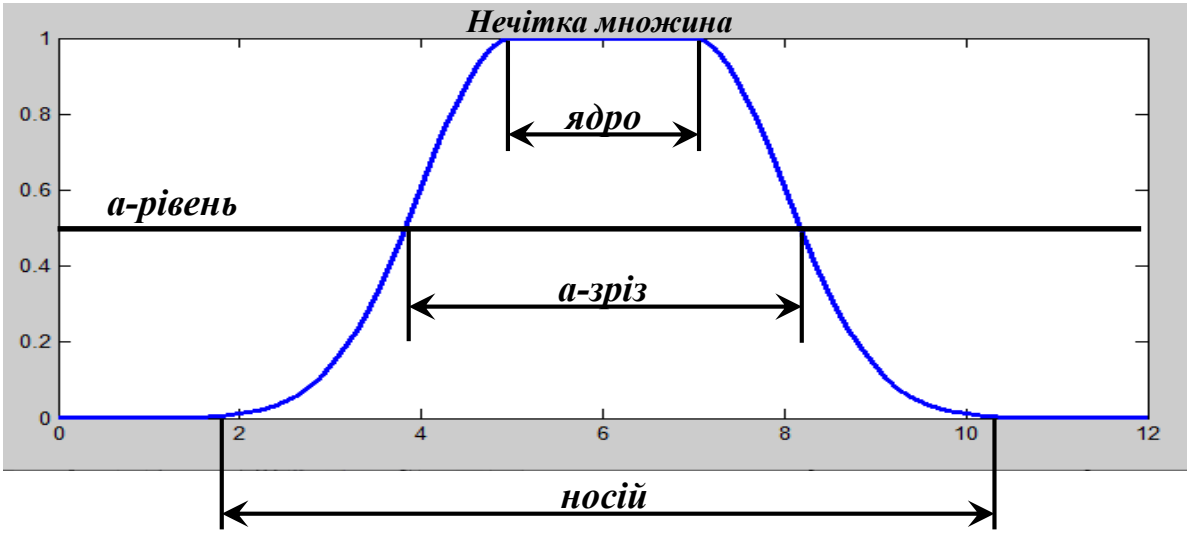


Рисунок 3.3 – Параметри функції належності.

Означення 3. **Лінгвістична змінна** це п’ятірка:

$$<\beta, T, X, G, M>$$

де:

β – найменування лінгвістичної змінної;

T – базова терм-множина лінгвістичної змінної або множина її значень (термів), кожен з яких є найменуванням окремої нечіткої змінної α;

X – галузь визначення нечітких змінних, межі, в яких змінюється лінгвістична змінна β;

G – синтаксичні правила, що описують процес генерації нових термів із множини T;

M – семантичні правила, що задають функції належності термам, створеним правилами із G .

Лінгвістичну змінну можна визначити як змінну, значеннями якої є не числа, а слова або висловлювання природною (або формальною) мовою.

Наприклад, лінгвістична змінна «вік» може приймати такі значення: «дуже молодий», «молодий», «середнього віку», «старий», «дуже старий» та ін. Зрозуміло, що змінна «вік» буде звичайною змінною, якщо її значення – точні числа; лінгвістичної вона стає, будучи використаною в нечітких міркуваннях людини.

Приклад лінгвістичної змінної для деякої предметної галузі:

β – швидкість автомобіля;

T = {«низька», «середня», «висока»};

X = [0, 100];

G – правила (процедура) створення нових термів за допомогою логічних зв’язок: «І», «АБО», «НЕ» і модифікаторів типу «дуже», «злегка» і ін. Наприклад, «мала низька», «дуже висока».

M – правила (процедура) визначення на X=[0, 100] нечітких змінних α1=«низька», α2=«середня», α3=«висока», а також термів із $G(T)$ у відповідності

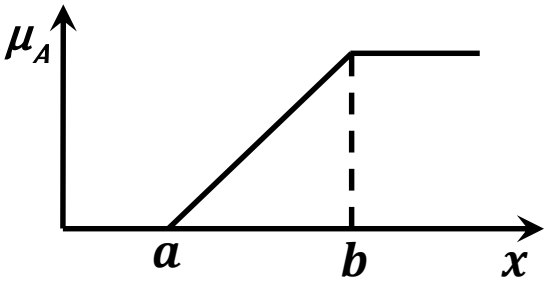
до правил трансляції нечітких зв'язок і модифікаторів «І», «АБО», «НЕ», «дуже», «злегка».

3.3. Типи функцій належності

Кусково-лінійні функції належності

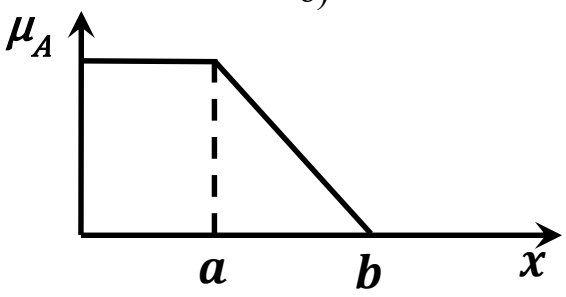
Виходячи з назви, вони складаються з відрізків прямих ліній, утворюючи безперервну або кусочно-безперервну функцію. На рис. 3.4 а), 3.4.б) представлені монотонні лінійні функції належності.

а)



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

б)



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

Рисунок 3.4 – Монотонні лінійні функції належності

$$\mu_A(x)=\begin{cases}0, & x<a\\ \dfrac{x-a}{b-a}, & a\le x\le b,\\ \dfrac{c-x}{c-b}, & b\le x\le c,\\ 0, & x>c\end{cases}$$

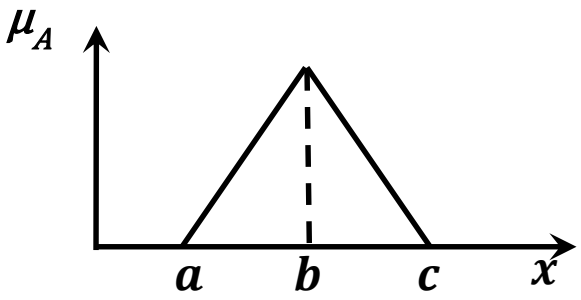


Рисунок 3.5 – Трикутна функція належності

$$\mu_A(x)=\begin{cases}0, & x<a\\ \dfrac{x-a}{b-a}, & a\le x\le b,\\ 1 & b<x<c,\\ \dfrac{d-x}{d-c}, & b\le x\le c,\\ 0, & x>d\end{cases}$$

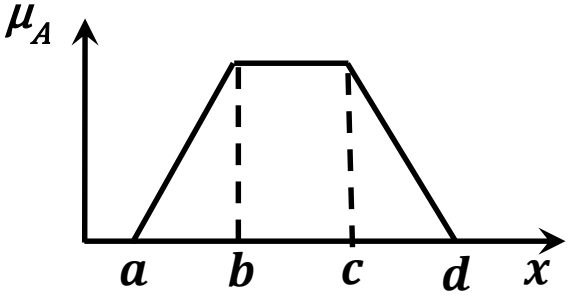


Рисунок 3.6 – Трапецієподібна функція належності

$$\mu_A(x) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$

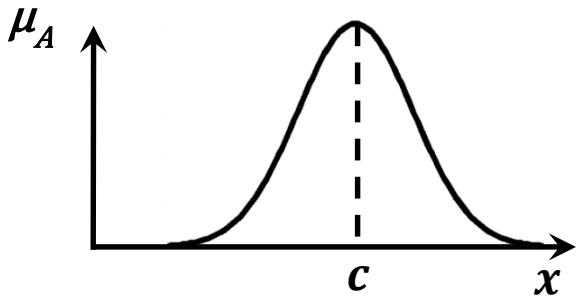


Рисунок 3.6 – Дзвіноподібна функція належності (Функція Гауса)

$$\mu_A(x) = \begin{cases} 1, & x = a, \\ 0, & x \neq a \end{cases}$$

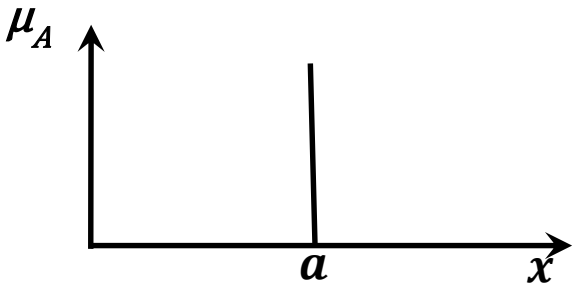


Рисунок 3.7 – Функція належності типу синглтон

Функції типу S

Використовуються для подання таких нечітких множин, які характеризуються невизначеністю типу: «велика кількість», «велике значення», «значна величина», «високий рівень доходів і цін», «висока норма прибутку», «висока якість послуг », « високий сервіс обслуговування ». Особливість нечіткого моделювання при цьому полягає в поданні відповідних нечітких множин за допомогою монотонно зростаючих функцій належності.

$$s(x;a,b) = \begin{cases} 0, & x < a, \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x-a}{b-a}\pi\right), & a < x \le b, \\ 1, & x > b, \end{cases}$$

Одним з прикладів таких функцій є монотонна лінійна рис. 3.4 а).

Функція типу Z

Використовуються для нечітких, які характеризуються невизначеністю типу: «малу кількість», «невелике значення», «незначна величина», «низька собівартість продукції», «низький рівень цін або доходів», «низька процентна ставка». Загальним для всіх таких ситуацій є слабка ступінь про явища того чи іншого якісного або кількісного ознаки. Особливість нечіткого моделювання при цьому полягає в поданні відповідних нечітких множин за допомогою монотонно спадаючих функцій належності.

$$z(x;a,b)=\begin{cases}1, & x<a,\\ \frac{1}{2}+\frac{1}{2}\cos\left(\frac{x-a}{b-a}\pi\right), & a<x\le b,\\ 0, & x>b,\end{cases}$$

Одним з прикладів таких функцій є монотонна лінійна рис. 3.4 б).

Функція типу π

До даного типу функцій належності можна віднести цілий клас кривих, які за своєю формою нагадують дзвін, згладжену трапецію або літеру «**π**». Одним з способів завдання функції є використання функцій типу **S** і **Z**

$$\pi(x;a,b,c)=\begin{cases}s\left(x;c-b,c-\frac{b}{2},c\right), & x\le c,\\ 1-s\left(x;c,c+\frac{b}{2},c+b\right), & x>c.\end{cases}$$

3.3. Операції над нечіткими множинами

Операції над нечіткими множинами мають три особливості: по-перше, результат операції над нечіткими множинами є узагальненнями операцій над класичними множинами. Так як для нечутких множин можливі різні варіанти узагальнення, це призводить до різних варіантів операцій над ними. По-друге, виконання операцій над нечіткими множинами можливо, коли нечіткі множини визначені на одному і тому ж універсумі. По-третє, операції над нечіткими множинами зводяться до операцій над їх функціями належності. З урахуванням цих особливостей розглянемо основні операції над нечіткими множинами. Слід зауважити, що не всі аксіоми теорії множин виконуються для операцій над нечіткими множинами.

Домінування *(Вміщення)*

Нехай A і B – нечіткі множини на універсальній множині X.

Говорять, що A міститься в B, якщо $\forall x \in X$: $\mu_A(x) < \mu_B(x)$.Позначення: A $\subset$ B.

Інколи використовують термін «домінування», тобто у випадку, якщо A $\subset$ B, говорять, що B домінує A.

Рівність

A і B рівні, якщо $\forall x \in X$: $\mu_A(x) = \mu_B(x)$

Позначення: A = B.

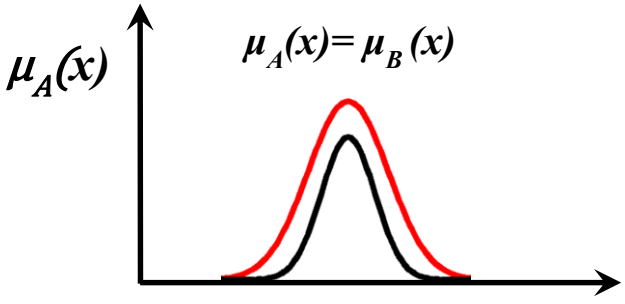


Рисунок 3.7 – Операція Домінування (Вміщення)

Доповнення

Нехай $\mu = [0, 1]$, A і B – нечіткі множини, задані X доповнюють один одного, якщо

$\forall x \in X: \mu_A(x) = 1 - \mu_B(x)$

Доповнення нечіткої множини A позначається символом $\bar{A}$.

Операція доповнення відповідає логічному запереченню.

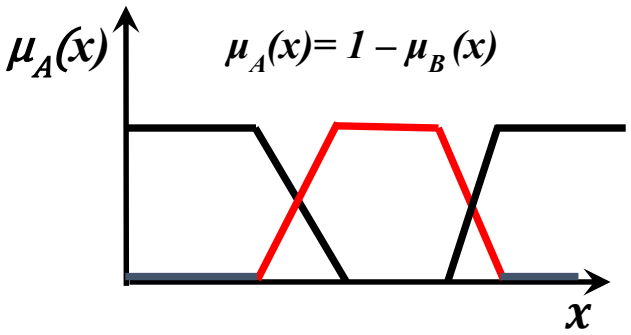


Рисунок 3.8 – Операція Доповнення

Перетин

Перетин A і B позначається A ∩ B визначається

$\mu_{A \cap B}(x) = min\ \mu_A(x), \mu_B(x))$

Перетин відповідає логічній зв'язці «і» – найменша нечітка підмножина, яка міститься одночасно в A і B.

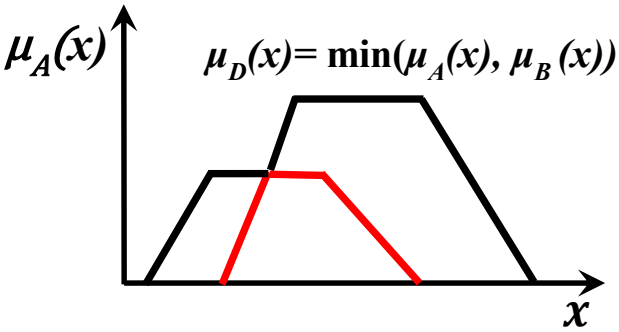


Рисунок 3.9 – Операція Перетину

Об'єднання

Об'єднання нечітких множин A і B (A + B)

Об'єднання відповідає логічній зв'язці «або».

A U B – найбільша нечітка підмножина, яка включає як A, так і B, з функцією належності:

$\mu A \cup B(x) = max\ (\mu A(x), \mu\ B(x))$.

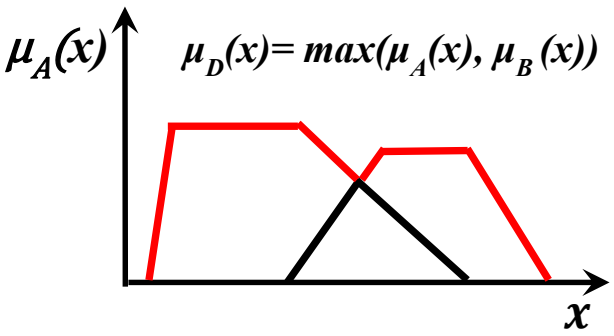


Рисунок 3.10 – Операція Об’єднання

Концентрація

$CON(A)=A^2$

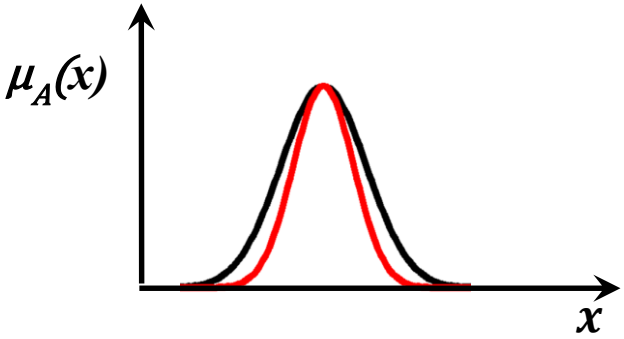


Рисунок 3.11 – Операція Концентрації

Розтягування (розмивання)

$DIL(A)=\sqrt{A}$

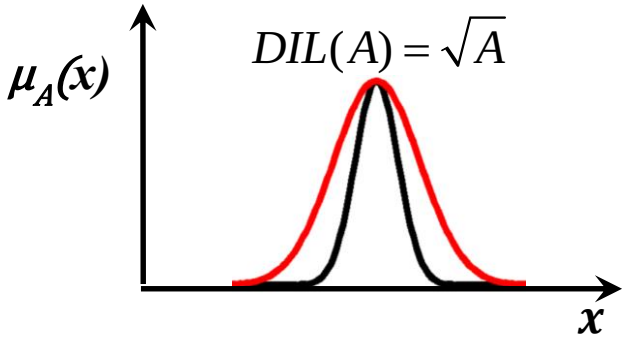


Рисунок 3.11 – Операція Розтягування

Поняття нечіткого числа. Операції над нечіткими числами

Нечітким числом називається задана на множині дійсних чисел нечітка множина, що має нормальну і опуклу функцію належності, тобто таку, що [6]:

a) існує значення елемента, в якому функція належності дорівнює одиниці;

b) при зміщенні від свого максимуму вліво чи вправо функція належності не зростає.

Нечітке число A називається ***нечітким нулем***, якщо

$$\mu_A(0) = \sup_x(\mu_A(x)).$$

Підмножина $S_A \subseteq R$ називається ***носієм нечіткого числа*** A, якщо

$$S_A = \{x | \mu_A(x) > 0\}.$$

Відповідно до принципу узагальнення Заде введено поняття арифметичних операцій для нечітких чисел. Для довільних нечітких чисел A,B,C і для будь-яких чисел x y z R, , □ операції додавання, віднімання, множення та ділення реалізуються наступним чином (символ * позначає кожну з цих операцій):

$$c = A \tilde{*} B \Leftrightarrow \mu_C(z) = \sup_x(\min \mu_A(x), \mu_B(x))).$$

Функція *sup* для скінчених нечітких множин та безперервних функцій належності обчислюється як максимум мінімумів для всіх можливих пар значень x та y, які утворюють значення $z=x*y$, для якого визначається значення функції належності.

Розглянемо приклад виконання операції над нечіткими числами. Нехай дані нечіткі числа $A = \{\langle 1|0,4\rangle, \langle 3|0,9\rangle\}$ та $B = \{\langle 0|0,3\rangle, \langle 2|0,6\rangle\}$. Необхідно знайти нечітке число $C=A+B$. Універсум нечіткої множини C складається з усіх можливих варіантів сум елементів універсумів нечітких множин A та B. Необхідно знайти їх та обчислити відповідні значення функції належності по формулі.

1) $z = 1+0=1$.

$$\mu_C(1) = \max_{z=1}(\min(\mu_A(1), \mu_B(0))) = \max_{z=1}(\min(0,4; 0,3)).$$

2) $z = 1+2=3+0=3$.

$$\mu_C(1) = \max_{z=3}(\min(\mu_A(1), \mu_B(2)), \min(\mu_A(3), \mu_B(0))) = \max_{z=1}(\min(0,4; 0,3)) =$$

$$= \max_{z=3}(\min(0,4; 0,6)), \min(0,9; 0,6)) = 0,4.$$

3) $z = 3+2=5$.

$$\mu_C(5) = \max_{z=5}(\min(\mu_A(3), \mu_B(2))) = \max_{z=1}(\min(0,9; 0,6)) = 0,6.$$

Таким чином, нечітка множина C = $\{\langle 1|0,3\rangle, \langle 3|0,4\rangle, \langle 5|0,6\rangle\}$.

Основні види нечітких чисел та операції над ними

При вирішенні задач моделювання нечітких систем можна використовувати нечіткі числа (L-R)-типу, які мають більш просту реалізацію операцій.

Нечіткі числа (L-R)-типу є спеціальним видом нечітких чисел, які задаються за певними правилами для зниження обсягу обчислень при виконанні операцій над ними.

Функції належності для нечітких чисел (L-R)-типу задаються з використанням незростаючих на множині невід'ємних чисел функцій дійсної змінної $L\ x(\ )$ і $R\ x(\ )$, що задовольняють властивостям:

а) $L\ x(-=) L\ x(\ )$, $R\ x(-=) R\ x(\ )$;

б) $L(0) = R(0)$.

Нехай $L\ y(\ )$та $R\ y(\ )-$ функції (L-R)-типу. Унімодальне нечітке число A з модою a (тобто $\mu_A(a) = 1$) можна задати наступним чином:

$$\mu_{A_{LR}}(x) = \begin{cases} L\left(\dfrac{a-x}{a}\right), & якщо\ x \le a, \\ R\left(\dfrac{a-x}{\beta}\right), & якщо\ x > a, \end{cases}$$

де a – мода; $\alpha>0$, $\beta>0$ – лівий і правий коефіцієнти нечіткості.

Таким чином, при заданих $L\ (y)$ та $R\ (y)$ унімодальне нечітке число A задається трійкою $(a;\alpha,\beta)$.

Унімодальні нечіткі числа (L-R)-типу називають трикутними числами.

Трикутні числа формалізують висловлювання типу "приблизно дорівнює a". Ясно, що $a \pm \delta \approx a$, причому при зниженні δ до нуля ступінь впевненості в оцінці величини зростає до одиниці. Трикутні нечіткі числа є одним типів нечітких чисел, що використовується найбільш часто.

На практиці часто використовується альтернативне визначення нечіткого трикутного числа.

Трикутним нечітким числом A називається нечітке число, представлене трійкою дійсних чисел $\langle a\ b\ c\rangle$, де $(a \le b \le c)$, через які його функція належності визначається наступним чином:

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

Друге число b трійки $\langle a,\ b,\ c\rangle$, зазвичай називають модою або чітким значенням нечіткого трикутного числа. Числа a і c визначають так званий ступінь розмитості числа.

При завданні функції належності трикутного нечіткого числа не завжди потрібно використовувати лише лінійні функції. Часто в різних практичних застосуваннях використовуються дві функції, з яких одна монотонно зростає на інтервалі $[a,\ b]$, а друга монотонно убуває на інтервалі $[b,\ c]$. Таким чином, без зниження рівня узагальненості, кожне трикутне нечітке число може бути представлено впорядкованою трійкою дійсних чисел.

Якщо $A = \langle a_A, b_A, c_A\rangle$ і $B = \langle a_B, b_B, c_B\rangle$ є трикутними нечіткими числами, то, згідно з принципом узагальнення Заде, нечітке число $C = A^* B$ також є трикутним і характеризується трійкою $\langle a_C, b_C, c_C\rangle$, де:

$$a_C = \min\{a_A^* a_B,\ a_A^* c_B,\ c_A^* a_B,\ c_A^* c_B\},$$

$$c_C = \max\{a_A^* a_B,\ a_A^* c_B,\ c_A^* a_B,\ c_A^* c_B\},$$

$$b_C = b_A^* b_B$$

Під * розуміються операції додавання, віднімання, множення та ділення.

Однак в деяких випадках при оцінках параметрів використовуються не конкретні значення, а певні інтервали. Наприклад, при оцінці тривалості виконання задачі по проекту може бути отримана оцінка «потрібно від 15 до 20 годин робочого часу». Для формалізації подібних значень використовують так звані трапецієподібні нечіткі числа, які також називають нечіткими інтервалами.

Трапецієподібним нечітким числом (***нечітким інтервалом***) $A = \langle a,b,c,d \rangle$ називається нечітке число із трапецієподібною функцією належності:

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

Арифметичні операції над двома трапецієподібними нечіткими числами $A = \langle a_A,b_A,c_A,d_A \rangle$ та $B = \langle a_B,b_B,c_B,d_B \rangle$ виконуються за правилами:

1) додавання

$$\langle a_A,b_A,c_A,d_A \rangle + \langle a_B,b_B,c_B,d_B \rangle = \langle a_A + a_B, b_A + b_B, c_A + c_B, d_A + d_B \rangle;$$

2) віднімання

$$\langle a_A, b_A, c_A, d_A \rangle + \langle a_B, b_B, c_B, d_B \rangle = \begin{pmatrix} a_A - b_B + c_B - d_B \\ b_A - b_B \\ c_A - c_B \\ d_A + b_B - a_B - c_B \end{pmatrix};$$

3) множення

$$\langle a_A, b_A, c_A, d_A \rangle \times \langle a_B, b_B, c_B, d_B \rangle = \begin{pmatrix} b_A \times a_B - b_A \times b_B + a_A \times b_B \\ b_A \times b_B \\ c_A \times c_B \\ c_A \times d_B - c_A \times c_B + d_A \times c_B \end{pmatrix};$$

4) ділення

$$\langle a_A, b_A, c_A, d_A \rangle / \langle a_B, b_B, c_B, d_B \rangle = \begin{pmatrix} (b_A \times c_B - b_A \times d_B + a_A \times c_B) / c_B^2 \\ b_A / c_B \\ c_A / b_B \\ (c_A \times b_B - c_A \times a_B + d_A \times b_B) / b_B^2 \end{pmatrix};$$

Існують також інші підходи до завдання операцій над трикутними та трапецієвидними нечіткими числами, з якими можна ознайомитися у спеціальній літературі.

3.3. Нечітка система виведення

Нечітке логічне виведення

Математична постановка задачі

Знання про ПГ можна формалізувати у виді системи нечітких логічних тверджень (висловлювань) «Якщо А ТО В» у термінах лінгвістичних змінних та нечітких множин.

Нечіткі вхідні значення системи перетворюються на вихідні на основі правил нечіткої логіки. Нехай система прийняття рішень здійснює перетворення значень *n* вхідних лінгвістичних змінних $\tilde{X} = \{\tilde{X}_i \mid i = 1...n\}$ у вихідну лінгвістичну змінну $\tilde{Y} = R(\tilde{X})$ згідно з базою правил $R = \{R_k \mid k = 1...K\}$. Правила R акумулюють знання розробників чіткої системи у виді нечіткої імплікації $R=A \rightarrow B$, яку можна розглядати як нечітку множину на декартовому добутку носіїв вхідних та вихідних нечітких множин. Процес отримання нечіткого результату B' з нечітких вхідних множин A' на основі знань $A \rightarrow B$ можна зобразити у такому виді:

$$B' = A' \bullet R = A' \bullet (A \rightarrow B),$$

де: $\bullet$ – композиційне правило нечіткого виведення.

Запропоновано достатньо багато композиційних правил, що можна вважати недоліком нечіткого виведення, тому що важко визначити, яке правило є кращим для конкретної задачі. Найбільш відомими є:

– «Правило Мамдані» (Правило типу minimum)

$$\mu_{A \rightarrow B}(x,y) = \mu_R(x,y) = \mu_A(x) \wedge \mu_B(y) = \min [\mu_A(x), \mu_B(y)]$$

– «Правило Ларсена» (Правило типу «добуток»)

$$\mu_{A \rightarrow B}(x,y) = \mu_R(x,y) = \mu_A(x) \cdot \mu_B(y)$$

– *«Правило Лукашевича»*

$$\mu_{A \rightarrow B}(x,y) = \mu_R(x,y) \min [1, 1 - \mu_A(x) + \mu_B(y)]$$

– *«Правило Заде»* (Правило типу max-min)

$$\mu_{A \rightarrow B}(x,y) = \mu_R(x,y) = \max\{\min [\mu_A(x) + \mu_B(y)], [1 - \mu_A(x)]\}$$

Загальний алгоритм нечіткого виведення

У відповідності до математичної постановки функціональність нечіткої системи прийняття рішень визначається такими кроками:

1) fuzzyfication (фазифікація) – перетворення чітких вхідних змінних на нечіткі, тобто визначення ступеня відповідності значень входів кожній із нечітких множин;

2) обчислення правил на основі використання нечітких операторів та застосування нечіткої імплікації для отримання вихідних значень правил;

3) агрегування нечітких виходів правил у загальне нечітке вихідне значення;

4) defuzzification (дефазифікація) – перетворення агрегованого вихідного значення на чітке значення.

Фазифікація входів

Фазифікація полягає у перетворенні чітких значень вхідних змінних:

$\overline{x} = (x_1, x_2, ...x_n)$ на нечіткі множини:

$A' = (A'_1, A'_2, .., A'_n)$

При фазифікації чіткого входу визначають ступені його відповідності кожному лінгвістичному терму A_i^k з функціями належності $\mu_{A_i^k}(x), \quad i = 1, ..., n$. Ці ступені є значеннями функцій належності $\mu_{A_i^k}(x)$ для поточного значення змінної x_i.

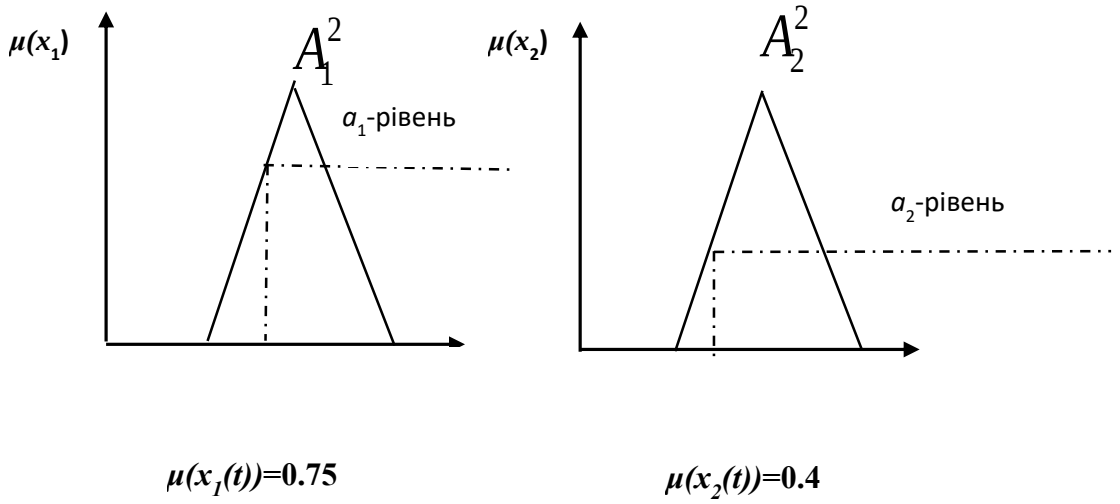


Рисунок 3.12 – Приклад фазифікації для двох змінних

Нечітке логічне виведення

Основою проведення операції нечіткого логічного виведення є база правил, що містить нечіткі висловлювання у вигляді **Якщо А То В** і функції належності відповідних лінгвістичних термів. Для повноти бази нечітких правил повинні виконуватися такі умови:

1) для будь-якого терму вхідної змінної існує хоча б одне правило, в якому цей терм використовується у лівій частині правила;

2) існує хоча б одне правило для кожного лінгвістичного терму вихідної змінної.

Інакше має місце неповна база нечітких правил.

Загалом до правила можуть входити всі можливі комбінації лінгвістичних термів для усіх вхідних змінних, об'єднаних логічними операціями. Слід зазначити, що за допомогою перетворень нечітких множин будь-яке правило, що містить у лівій частині як кон'юнкції, так і диз'юнкції, можна перетворити на систему правил, у лівій частині яких будуть або тільки кон'юнкції, або тільки диз'юнкції. Для визначення нечіткої кон'юнкції можна використати знаходження мінімуму, а для нечіткої диз'юнкції – знаходження максимуму двох функцій належності. Не зменшуючи загальності, розглядатимемо правила,

побудовані на основі кон'юнкції. Розрізняють дві моделі логічного виведення: Мамдані (Mamdani) та Сугено (Sugeno).

Модель Мамдані оперує лише лінгвістичними змінними та нечіткими множинами і перетворює нечіткі входи на нечіткі виходи. Наприклад, для моделі Мамдані правила мають вигляд:

$$R_k : if\ x_1\ is\ A_1^k and ... and\ x_n\ is\ A_n^k\ then\ y\ is\ B^k$$

де $A_n^k \in \tilde{X}_i$ – нечіткі множини для вхідних та $B^k \in \tilde{Y}$ – нечіткі множини для вихідної лінгвістичних змінних, які використовуються в k-му правилі (k=1…K). Операція *and* інтерпретується як $min\ (\mu_A(x), \mu_B(y))$.

Модель Сугено оперує з чіткими величинами, лінгвістичними змінними та нечіткими множинами і перетворює чіткі входи у чіткі виходи. Правила моделі Сугено можуть мати вигляд:

$$R_k : if\ x_1\ is\ A_1^k and ... and\ x_n\ is\ A_n^k\ then\ y = f_k(x_1, ..., x_n)$$

де f_k –лінійні функції входів.

Для кожного правила 1...N визначається рівень його впевненості a_k (*а-зріз*) за правилом нечіткої імплікації щодо входів. В результаті одержимо нові нечіткі змінні, або відповідні ступені впевненості в значенні виходів при застосуванні відповідного правила до заданих входів. Так, на основі визначення нечіткої імплікації, наприклад за правилом Мамдані, як мінімуму лівої й правої частин правила, маємо:

$$B_k^{'} = \min(a_k, B_k), \quad k = 1..N,$$

де B_k – зрізи вихідних нечітких множин на рівні a_k.

Агрегування виходів

Завершальним кроком нечіткого логічного виведення є агрегування виходів правил. Один з основних способів агрегації є нечітка диз'юнкція вихідних множин, або, інакше, знаходження максимуму отриманих функцій належності. Як результат, отримаємо значення агрегованого виходу:

$$B^{'} = \max_k(B_k^{'}), \quad k = 1..N,$$

При нечіткому логічному виведенні паралельно опрацьовують велику кількість правил з подальшим їх агрегуванням у завершальне рішення. Правила можуть будуватися на основі досвіду та знань експертів, створенням моделі дій оператора, методом навчання. При проектуванні пристроїв з нечіткою логікою важливо забезпечити можливості їх пристосування до змін навколишнього середовища методом навчання бази правил за експериментальними даними.

Навчання полягає в адаптивному підборі параметрів нечітких множин та автоматичному генеруванні правил нечіткого логічного виведення. Для цього використовуються алгоритми оптимізації та інтелектуального опрацювання даних – градієнтний, генетичний, штучних нейронних мереж, байєсових мереж та ін.

Дефазифікація виходів

Після визначення індивідуальних виходів правил здійснюється *дефазифікація* агрегованого виходу. У загальному випадку етап дефазифікації є

необов’язковим і використовується за необхідності перетворення виведених нечітких лінгвістичних змінних до точного значення.

COG (Center Of Gravity, centroid) – "центр тяжіння". Фізичним аналогом цієї формули є знаходження центру тяжіння плоскої фігури, обмеженої осями координат та графіком функції належності нечіткої множини.

$$y_{cog}(B) = \frac{\int_{Min}^{Max} \mu_B(y) \cdot y dy}{\int_{Min}^{Max} \mu_B(y) dy}$$

COA (Centre of Area, Bisector of Area, bisector) – центр площі.

Перпендикуляр, відновлений в цій точці, ділить площу під кривою функції належності на дві рівні частини.

$$y_{coa}(B) = \frac{\int_{Min}^{U} \mu_B(y) \cdot y dy}{\int_{U}^{Max} \mu_B(y) dy}$$

SOM (Smallest of Maximum)

$$y_{SOM} = \min(y_m)$$

де y_m – модальне значення результуючої функції належності. Іншими словами, як чіткої вихідної змінної береться найменша (найбільша ліва) мода.

LOM (Largest Of Maximums)

$$y_{LOM} = \max(y_m)$$

де y_m – модальне значення результуючої функції належності. Іншими словами, як чіткої вихідної змінної береться найбільша (сама права) мода.

MOM (Mean Of Maximums) – "центр максимумів". При використанні методу центру максимумів потрібно знайти середнє арифметичне елементів універсальної множини, що мають максимальні ступені належності.

$$y_{MOM} = \frac{\int_{G} u du}{\int_{G} du}$$

де G – множина всіх елементів з інтервалу, що мають максимальну ступінь належності нечіткій множині.

Складові системи нечіткого виведення подані на рис. 3.13

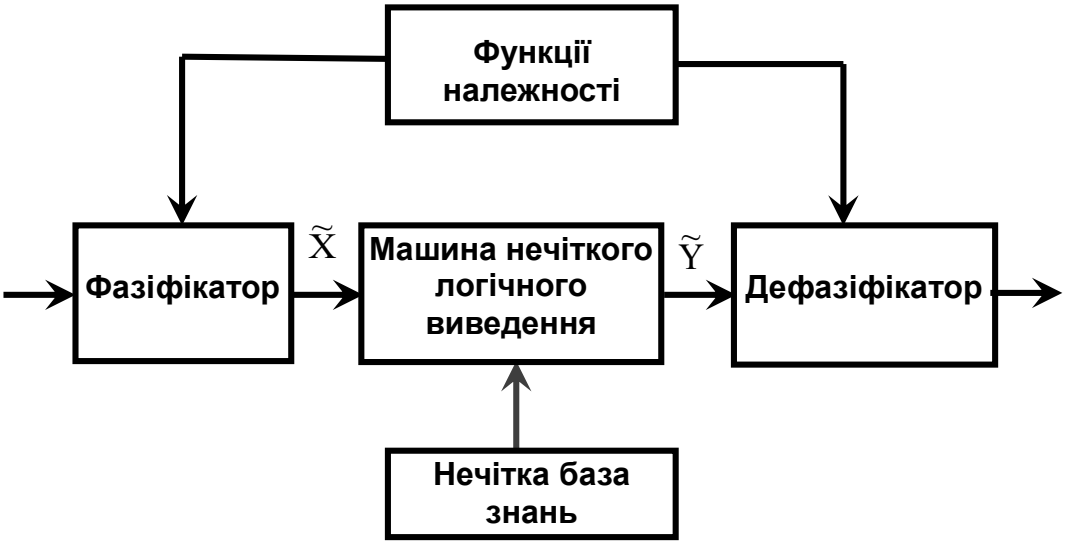


Рисунок 3.13 – Структура системи нечіткого логічного

Алгоритм Мамдані

Ebrahim Mamdani
1942

В Англії у 1973 в University of London професор Mamdani і студент S. Assilian на засадах нечіткої логіки намагались стабілізувати швидкість невеликого парового двигуна, розробленого студентом.

Функціональна схема процесу нечіткого виведення у спрощеному вигляді представлена на рис. 3.13. На цій схемі виконання першого етапу нечіткого виведення – фазифікації – здійснює фазифікатор. За процедуру безпосередньо нечіткого виведення відповідальна машина нечіткого логічного виведення, яка виробляє другий етап процесу виведення на підставі нечіткої бази знань (набору правил), а також етап композиції. Дефаззифікатор виконує останній етап нечіткого виведення – дефаззифікацію.

Алгоритм Мамдані:

1. Введення нечіткості (fuzzification): для заданих (чітких) значень аргументів $x_1=x_1(t)$, $x_2=x_2(t)$ знаходяться степені істинності для передумов правил 1 і 2.

2. Знаходиться рівні зрізів для передумов кожного з правил (з використання правила мінімуму):

$$a_1 = \min[\mu_{A_1^1}(x_1(t)), \mu_{A_2^1}(x_2(t))]$$

$$a_2 = \min[\mu_{A_1^2}(x_1(t)), \mu_{A_2^2}(x_2(t))]$$

3. Знаходиться *a*-зрізи вихідних функцій належності

$$B^1[y(t)] = a_1 * B^1$$

$$B^2[y(t)] = a_2 * B^2$$

4. З використанням операції *max* проводиться агрегація (об’єднання) знайдених зрізаних функцій, що призводить до отримання підсумкової нечіткої множини для змінної виходу з функцією належності:

$$B = a_1 * B^1 \vee a_2 * B^2$$

5. Приведення до чіткості (знаходження *y(t))* проводиться, наприклад, методом центроїда (як центр ваги фігури під кривою функції належності).

Приклад нечіткого виведення за алгоритмом Мамдані наведений на рис.3.14.

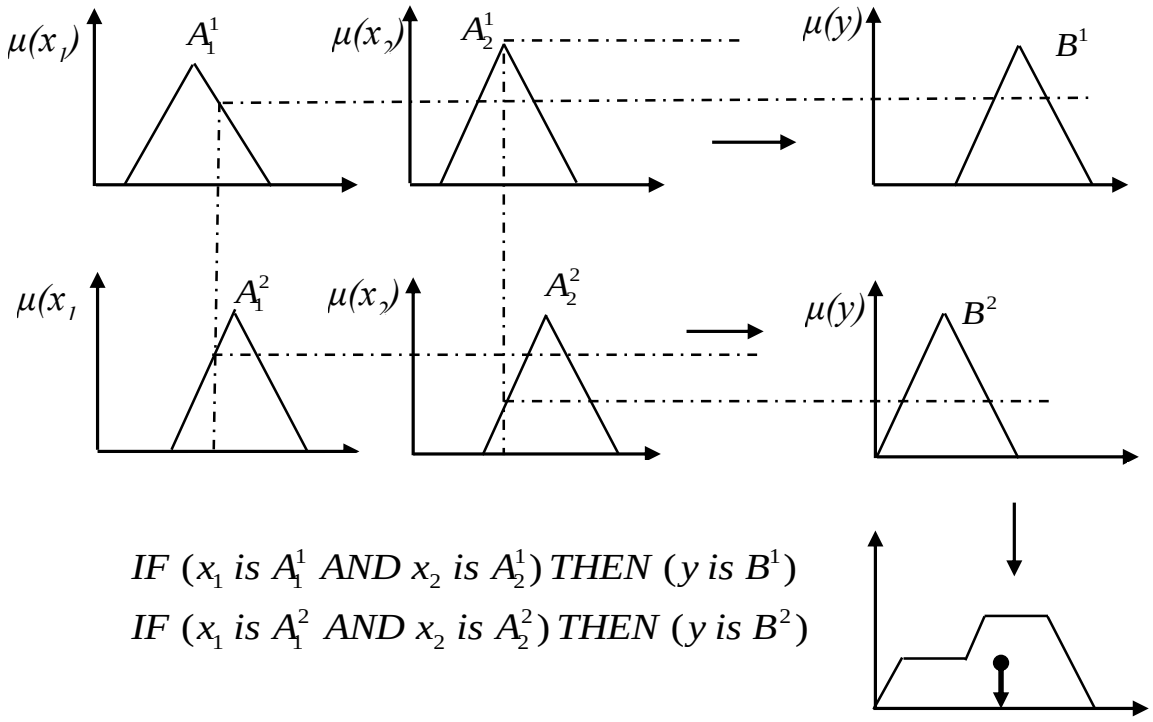


Рисунок 3.14 – послідовність дій алгоритму Мамдані

На рисунку розглянутий наступний приклад.

Припустимо, що базу знань утворюють два нечітких правила:

Правило 1: $IF\ (x_1\ is\ A_1^1\ AND\ x_2\ is\ A_2^1)\ THEN\ (y\ is\ B^1)$

Правило 2: $IF\ (x_1\ is\ A_1^2\ AND\ x_2\ is\ A_2^2)\ THEN\ (y\ is\ B^2)$

де: x_1, x_2 – імена вхідних змінних;

y – ім’я вихідної змінної;

$A_1^1, A_1^2, A_2^1, A_2^2$ – функція належності вхідних змінних;

B^1, B^2……. – функція належності вихідних змінних.

Переваги нечітких систем:

- функціонування в умовах невизначеності;
- оперування якісними та кількісними даними;
- використання експертних знань в управлінні;
- побудова моделей наближених міркувань людини;
- стійкість при дії на систему всіляких збурень.

Недоліками нечітких систем:

- відсутність стандартної методики конструювання нечітких систем;
- неможливість математичного аналізу нечітких систем існуючими методами;

.

4. ЕКСПЕРТНІ СИСТЕМИ

4.1. Означення поняття експертна система

Ідея побудови експертних систем (ЕС) сформувалася в ході досліджень в галузі штучного інтелекту, як переведення знань провідних експертів різних галузей до моделей подання знань і створення програмних систем для застосування цих знань в практичній діяльності. Наведемо декілька визначень фахівців, що розробляють ЕС.

Експертні системи – це програмні комплекси, що акумулюють досвід спеціалістів у деякій предметній галузі з метою його (досвіду) тиражування для консультацій менш кваліфікованих користувачів.

Експертні системи – комп'ютерні програми для виконання видів діяльності, що під силу людині-експерту. Вони імітують образ дій людини-експерта, істотно відрізняються від точних, добре аргументованих алгоритмів і не схожі на математичні процедури традиційних розробок.

Експертна система – програмний засіб, що використовує експертні знання для забезпечення високоефективного рішення неформалізованих задач у вузькій предметній галузі.

Експертні системи розпадаються на два великі класи з точки зору задач, які вони вирішують. Системи першого класу призначаються для підвищення культури роботи і рівня знань фахівців в різних галузях діяльності (лікарів, геологів, інженерів і т.д.). Системи другого класу можна назвати консультуючими, або діагностуючими.

Завдання експертних систем, які, по суті, являють собою комбінацію машинного і людського знання – зберігати і поповнювати досвід фахівців в галузях, які важко формалізувати, таких, як медицина, біологія, хімія, історія. Експертні системи повинні зіграти роль висококваліфікованих помічників, здатних дати корисну пораду користувачу, який перебуває в скрутному становищі. Ним може виявитися молодий лікар з недостатнім досвідом, перед яким виникла необхідність провести складну і нетривіальну операцію.

Експертна система зберігає масу відомостей, отриманих з різних джерел (книг, журнальних публікацій, усних повідомлень фахівців і т.д.). Вона може використовувати ці відомості для консультації і при необхідності пояснити фахівцеві, як вона прийшла до висновків, що повідомляються йому. Експертні системи здатні поповнювати свої знання в ході взаємодії з експертом. Експерт, знання якого вводяться в систему, може не бути знайомим з деталями програми і комп'ютером, на якому реалізована експертна система. Тому з'являється необхідність зручного та зрозумілого інтерфейсу користувача для заповнення бази знань, щоб уникнути залучення додаткових інженерів для роботи з базою знань.

4.2 Задачі експертних систем

Неформалізовані та слабоформалізовані задачі

На відміну від традиційних програм, призначених для вирішення математично строго визначених задач з точними алгоритмами, за допомогою

експертних систем вирішуються задачі, що відносяться до класу неформалізованих або слабоформалізованих задач. Алгоритмічних рішень таких задач або не існує в силу неповноти, невизначеності, неточності, розпливчастості розглянутих ситуацій і знань в них або ж такі рішення неприйнятні на практиці в силу складності дозволених алгоритмів. Тому експертні системи використовують логічне виведення і евристичний пошук рішення.

Неформалізовані задачі мають наступні особливості:

• помилковість, неоднозначність, неповнота і суперечливість вхідних даних;

- помилковість, неоднозначність, неповнота і суперечливість знань про проблемну галузь і розв’язувану задачу;
- велика розмірність простору рішення, тобто перебір при пошуку рішення досить великий;
- дані і знання змінюються у часі.

Крім цього, для неформалізованих задач загальним є:

- задачі не можуть бути задані в числовій формі;
- мету задачі не можна виразити в термінах точно визначеного критерію або цільової функції;
- не існує алгоритмічного розв’язку задачі;
- якщо алгоритмічний розв’язок є, то ним не можна скористатись через обмеженість ресурсів (час, пам’ять).

Критерії вибору задач для розв’язання експертною системою

Критерій – це свого роду правило, за допомогою якого можна що-небудь оцінити. До основних критеріїв вибору задач, що реалізуються методами і засобами експертних систем, можна віднести такі:

1. Дані і знання повинні бути надійними, достовірними, не змінюватись в часі, тобто бути стабільними в процесі вирішення задачі: не повинні коректуватися і не повинні містити помилок і суперечностей.

2. Простір або галузь можливих рішень відносно невелика. Простір пошуку повинен бути невеликим, оскільки необхідно зосередитися на вузькій предметній галузі, для якої характерний невеликий об’єм знань, у тому числі і основаних на здоровому глузді.

3. В процесі вирішення задачі повинні використовуватися формальні міркування, а задача повинна бути не дуже проста і не дуже важка для експерта (з розрахунку приблизно 30 хв. для експерта).

4. Задача повинна бути поставлена чітко, тобто визначені цілі (або мета), які ставляться перед експертною системою в процесі консультації, необхідний набір евристик, які використовуються в процесі вирішення задачі людиною.

5. Повинен бути, принаймні, один експерт, що вміє чітко виражати свої думки: явно формулювати свої знання і пояснювати методи застосування цих знань для вирішення задачі (експлікувати знання).

Також можна виділити додаткові (неявні) критерії вибору задач, що реалізуються методами і засобами ЕС.

1. Існування формалізованого апарату даної предметної галузі. Однією з найпривабливіших властивостей систем ШІ є здатність автоматизувати процес міркувань фахівців предметної галузі навіть в тих додатках, де відсутня для цього формалізована теоретична база. Це галузі, в яких застосовуються емпіричні асоціації для отримання шуканого результату.

2. Відбір такої предметної галузі, для якої підходять вербальні міркування. Галузі, в якій знання відносяться або до галузі чуттів, або не можуть бути висловлені і носять характер практичного досвіду, мало придатні для використання в системах штучного інтелекту.

В цілому, ЕС не рекомендується застосовувати для таких типів задач:

- математичних, які розв’язуються звичним шляхом формальних перетворень або процедурного аналізу;
- задач розпізнавання, які в загальному випадку розв'язуються чисельними методами (хоча в деяких ситуаціях це твердження спірне);
- задач, знання про методи вирішення яких відсутні.

Істотно утруднене створення ЕС для предметної галузі, в якій має місце одна або декілька перерахованих нижче особливостей:

- думки експертів часто не збігаються;
- використовуються складні стратегії міркувань;
- знання включають просторові і тимчасові співвідношення;
- є дуже велике число об’єктів розгляду і дуже багато залежить від понять, що опираються на "здоровий глузд";
- експертам потрібен значний час для вирішення задачі.

Відповідно до вищевказаного, тепер можна виділити клас задач, для яких розробка ЕС є найбільш придатною:

1. Задачі прогнозу і класифікації, які припускають велику кількість можливих відповідей (декілька десятків).
2. Багатопараметричні задачі, для яких важко, а іноді і неможливо визначити відповідні строгі аналітичні залежності.
3. Задачі, для вирішення яких потрібна тривала професійна підготовка.
4. Задачі обробки недостовірної інформації.

4.3. Класифікація експертних систем

У загальному випадку, всі системи, основані на знаннях, можна розділити на:

- системи, які вирішують задачі аналізу,
- системи, які вирішують задачі синтезу.

Основна відмінність задач аналізу від задач синтезу – в задачах аналізу множина рішень може бути перерахованою і включеною в систему, а в задачах синтезу множина рішень потенційно будується з рішень компонентів або підпроблем. Задачі аналізу – інтерпретація даних, діагностика, задачі синтезу – проектування, планування. Комбіновані задачі – навчання, моніторинг, прогнозування.

Загальна класифікація задач експертних систем

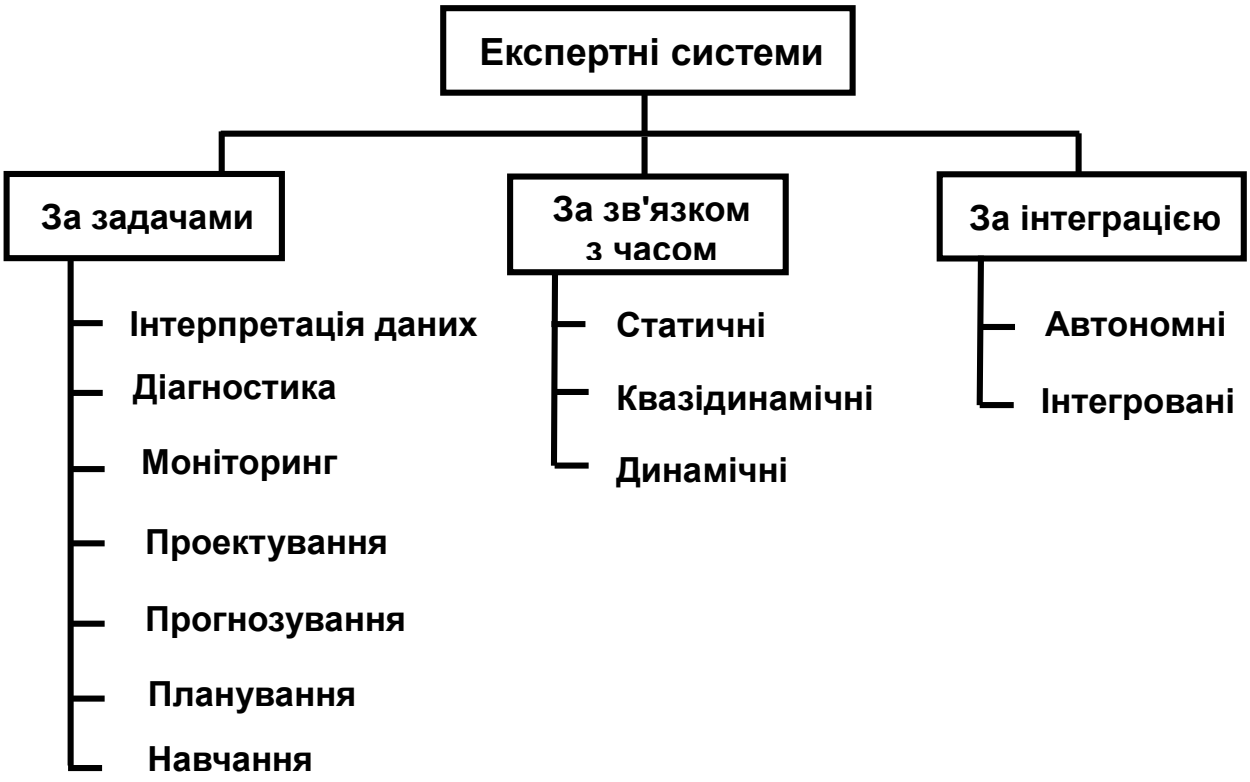


Рисунок 4.1 – Класифікація експертних систем

Інтерпретація даних. Під інтерпретацією розуміється визначення значення даних, результати якого повинні бути узгодженими і коректними. Звичайно передбачається багатоваріантний аналіз даних. Це одна з традиційних задач для ЕС.

Діагностика. Під діагностикою розуміється виявлення несправності в деякій системі. Несправність – це відхилення від норми. Таке трактування дозволяє з єдиних теоретичних позицій розглядати і несправність устаткування в технічних системах, і захворювання живих організмів і всілякі природні аномалії. Важливою специфікою є необхідність розуміння функціональної структури системи діагностування.

Моніторинг. Основна задача моніторингу – безперервна інтерпретація даних в реальному масштабі часу і сигналізація про вихід тих чи інших параметрів за допустимі межі. Головні проблеми – пропуск тривожної ситуації і інверсна задача помилкового спрацьовування. Складність цих проблем – в розмитості симптомів „тривожних” сигналів.

Проектування. Підготовка специфікацій на створення об'єктів з наперед визначеними властивостями. Під специфікацією розуміється весь набір необхідних документів: креслення, записка пояснення і т.д. Основні проблеми тут – отримання чіткого структурного опису знань про об'єкт. Для організації ефективного проектування необхідно формувати не тільки самі проектні рішення, але і мотиви їх ухвалення. Таким чином, в задачах проектування тісно пов'язуються два основні процеси: процес виведення рішення і процес пояснення.

Прогнозування. Системи прогнозування логічно виводять вірогідні наслідки із заданих ситуацій. У системі прогнозування звичайно використовується параметрична динамічна модель, в якій значення параметрів підганяються під задану ситуацію. Наслідки, що виводяться з цієї моделі, складають основу для прогнозів з оцінками вірогідності.

Планування. Під плануванням розуміється знаходження планів дій, що відносяться до об'єктів, здатних виконувати деякі функції. У таких ЕС використовуються моделі поведінки реальних об'єктів з тим, щоб логічно вивести наслідки планованої діяльності.

Навчання. Системи навчання діагностують помилки при вивченні будь-якої дисципліни за допомогою комп'ютера і підказують правильні рішення. Вони акумулюють знання про гіпотетичного учня і його характерні помилки, потім в роботі здатні діагностувати слабкості в знаннях тих, кого навчають, і знаходити відповідні способи для їх ліквідації. Такі системи планують процес спілкування з учнем залежно від успіхів учня, з метою передачі знань.

Статичні ЕС – розробляються в предметних областях, в яких БЗ і дані, що інтерпретуються, не змінюються в часі, вони стабільні.

Динамічні ЕС – працюють в поєднанні з датчиками об'єктів в режимі реального часу з безперервною інтерпретацією даних, що надходять. Структура система наведена на рис. 4.2.

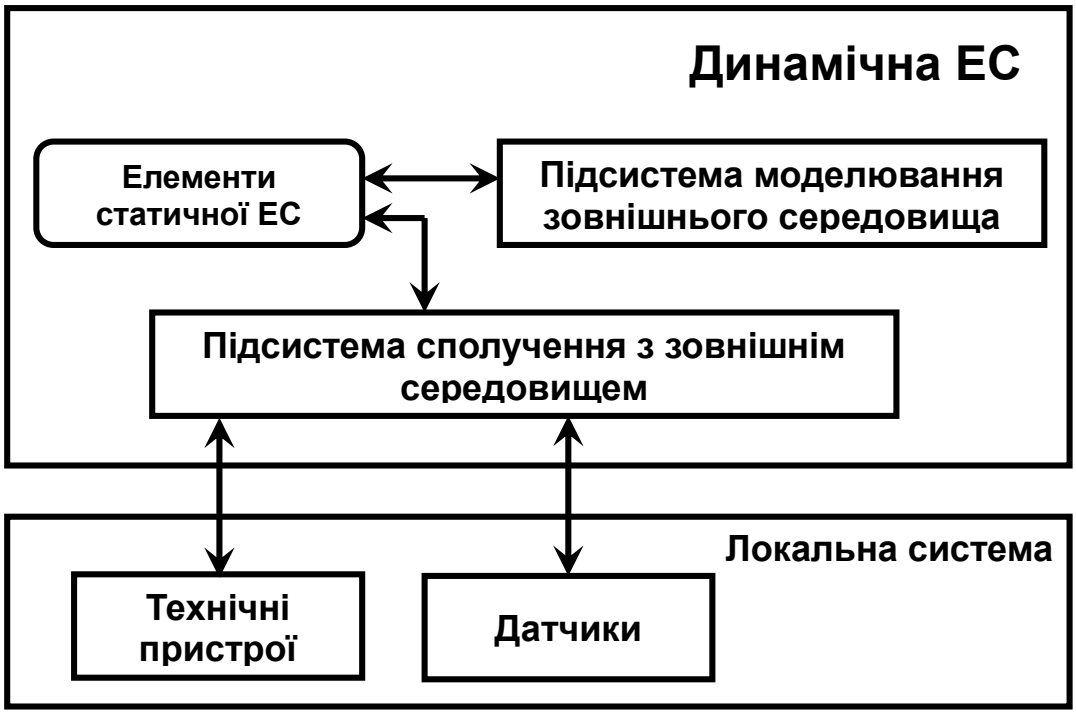


Рисунок 4.2 – Структура динамічної ЕС.

Квазідинамічні ЕС – інтерпретують ситуацію, яка змінюється з деяким фіксованим інтервалом часу.

Автономні ЕС – працюють безпосередньо в режимі консультацій з користувачем для специфічних експертних задач при рішенні яких не вимагається залучати традиційні методи обробки даних.

Гібридні ЕС – програмний комплекс, що агрегує стандартні прикладні програми (наприклад, математичну статистику, лінійне програмування, СУБД) і засоби маніпулювання знаннями. Це може бути інтелектуальна надбудова над прикладними програмами або інтегроване середовище для вирішення складної задачі з елементами експертних знань. Не дивлячись на зовнішню привабливість гібридного підходу, розробка таких систем є надзвичайно складною задачею. Компонування не просто різних програм, а різних методологій породжує цілий комплекс і теоретичних, і практичних труднощів.

В кожній галузі задачі теж класифікуються, наприклад, задачі ЕС в економічних інформаційних системах:

1. Аналіз фінансового стану підприємства.
2. Оцінка кредитної спроможності підприємства.
3. Планування фінансових ресурсів підприємства.
4. Формування портфеля інвестицій.
5. Страхування комерційних кредитів.
6. Вибір стратегії виробництва.
7. Оцінка конкурентоспроможності продукції.
8. Вибір стратегії ціноутворення.
9. Вибір постачальника продукції.
10. Підбір кадрів.

4.4. Проектування експертної системи

Приступаючи до створення ЕС необхідно спочатку вирішити, чи є створення і застосування ЕС для задачі, що стоїть перед кінцевим користувачем, **доцільним (доречним), виправданим і можливим.**

Критерії доцільності розробки ЕС:

- вирішення задачі опирається на використання операцій з символами, тобто задача не стільки пов'язана з розрахунками за формулами, скільки з логічним аналізом і перебором варіантів;
- задача не має чіткого алгоритмічного уявлення і опирається на використання евристик, які ведуть до результату, економлять набір переборів, але не дають гарантію успіху;
- задача не тривіальна;
- задача має кінцеву, прийнятну розмірність, тобто не є дуже громіздкою для вирішення на ЕОМ;
- задача викликає великий інтерес для практики і дозволяє знаходити рішення, які за допомогою класичних методів не можуть бути одержані.

Критерії виправданості розробки ЕС (хоча б одна позиція виконується):

- вирішення задачі має високу значущість або обіцяє принести високий дохід (не обов’язково в грошовому виразі);
- досвід вирішення задач в даній предметній галузі поступово втрачається;
- рівень підготовки фахівців-експертів в даній предметній галузі надзвичайно низький;

- накопичений досвід потрібен в багатьох областях (тиражування досвіду і знань)
- експертизи проводяться в середовищі, небезпечному для людини.

Критерії можливості розробки ЕС (обов'язкові всі вимоги):

- задача потребує лише інтелектуальних навиків;
- експерти можуть чітко висловити і описати використані ними методи роботи, які застосовуються у вирішенні даної задачі;
- є фахівці, чиї знання і досвід можна закласти в ЕС,
- експерти одностайні в рішенні задачі;
- задача не дуже важка;
- задача достатньо зрозуміла, якщо для вирішення задачі необхідно розробляти спеціальні методи, то ЕС не застосовується;
- задача вимагає знань і здорового глузду, які є результатом накопичення практичного досвіду, і цей досвід з якихось причин не вдається виділити і формалізувати.

Відповідно тепер можна сформулювати основні властивості експертної системи:

- наявність доступної бази знань;
- накопичення й зберігання високоякісного досвіду (інституціональна пам'ять);
- прогностичні можливості за рахунок включення відповідних моделей і алгоритмів обробки даних і знань;
- можливість навчання й тренування;
- здатність пояснення процесу прийняття рішень;
- наявність засобів редагування, що допомагають експертам модифікувати знання;
- наявність інтерфейсу з можливостями графічного виведення інформації.

У розробці ЕС беруть участь представники таких спеціальностей:

- експерт у проблемній галузі, задачі якої буде розв'язувати ЕС;
- інженер із знань – фахівець з розробки ЕС;
- програміст з розробки інструментальних засобів (ІЗ), призначених для прискорення розробки ЕС.

Експерт визначає знання (дані і правила), що характеризують проблемну галузь, забезпечує повноту і правильність введених у ЕС знань.

Інженер із знань допомагає експерту виявляти і групувати знання ЕС; здійснює вибір того ІЗ, що найбільше підходить для даної проблемної галузі, і визначає спосіб подання знань у цьому ІЗ; виділяє і програмує (традиційними засобами) стандартні функції (типові для даної проблемної галузі), що будуть використовуватися в правилах, які вводяться експертом.

Програміст розробляє ІЗ (якщо ІЗ розробляються заново), що містять всі основні компоненти ЕС, і здійснює їх поєднання з тим середовищем, у якому вони будуть використані.

Структура ЕС

Структура експертної системи наведена на рис. 4.3.

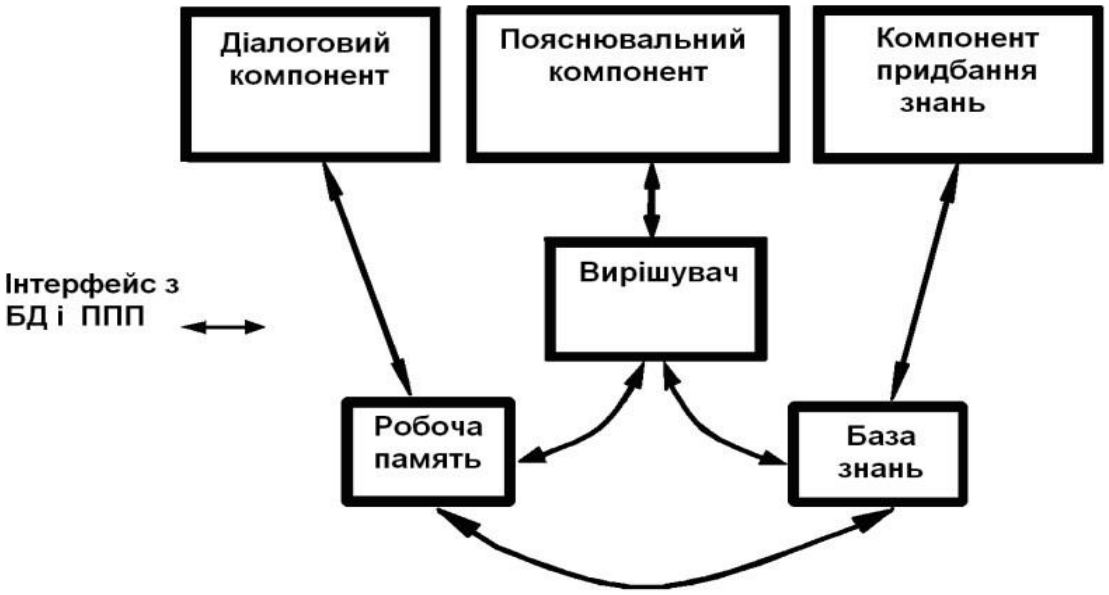


Рисунок 4.3 – Структура статичної експертної системи

База знань (БЗ) у ЕС призначена для збереження довгострокових даних, що описують розглянуту галузь (а не поточних даних), і правил, що описують доцільні перетворення даних цієї галузі.

Вирішувач використовуючи вихідні дані з робочої пам'яті і знання з БЗ, формує таку послідовність правил, що, будучи застосованими до вхідних даних, приводять до розв'язання задачі.

Компонент придбання знань автоматизує процес наповнення ЕС знаннями користувачем-експертом.

Діалоговий *компонент* орієнтований на організацію спілкування з користувачем.

Режими роботи ЕС

- придбання знань
- режим рішення задачі (режим консультації або режим використання).

У режимі *придбання знань* спілкування з ЕС здійснює експерт (за допомогою інженера знань).

Етапи розробки ЕС

Більшість розробників ЕС виділяють такі етапи.

1. ***Ідентифікація*****:**
   - визначити задачі, що підлягають рішенню
   - визначити мету розробки,
   - визначити експертів і тип користувачів.
2. ***Концептуалізація*****:**
   - проводиться змістовний аналіз предметної галузі,
   - виділяються основні поняття і їх взаємозв'язки,

- визначаються методи рішення задач.

3. ***Формалізація*:**
   - вибираються програмні засоби розробки ЕС,
   - визначаються способи представлення усіх видів знань,
   - формалізуються основні поняття.
4. ***Виконання:***
   - експертом здійснюється наповнення бази знань,
   - "витяг" знань з експерта,
   - організація знань, що забезпечує ефективну роботу ЕС,
   - представлення знань у виді, зрозумілому для ЕС.
5. ***Тестування:***
   - експерт і інженер по знанням з використанням діалогових і пояснювальних засобів перевіряють компетентність ЕС.
   - процес тестування продовжується до визнання експертом необхідного рівня компетентності системи.
6. ***Дослідна експлуатація:***
   - перевіряється придатність ЕС для кінцевих користувачів.

Проектування відбувається в парадигмі CASE засобів (рис.4.4), коли при виникненні проблем і помилок можливе повернення на попередні етапи, навіть з останнього на перший, для їх вчасного усунення, щоб досягти поставленої мети.

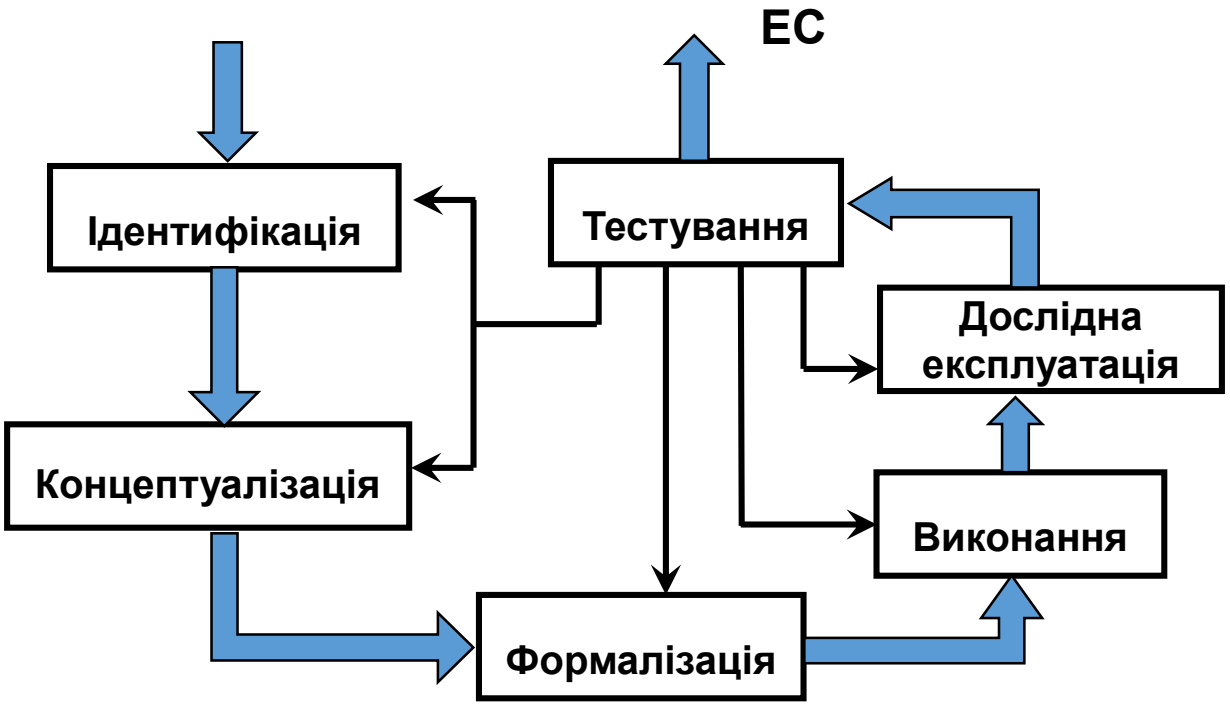


Рисунок 4.4 – Взаємодія етапів проектування ЕС.

При цьому створюється працюючий проект з мінімальними заявленими функціями і структурою, який поступово розширюється і наповнюється.

4.5. Інструментальні засоби розробки ЕС

Принцип інструментальних засобів полягає в тому, що програміст спочатку описує структуру прикладної задачі, а потім вказує що треба знайти, тобто вказує ціль. Його не цікавить яким чином ця задача буде вирішуватися, це

питання перекладається на механізм логічного виведення. Згідно цього принципу, твердження мови одночасно є і записами, які подібні до записів бази даних, і правилами, що несуть у собі способи їх обробки. За допомогою мов штучного інтелекту створюються бази знань та алгоритми їх обробки.

Мови програмування для ЕС

LISP* (*LISt Processing (обробка списків))

PROLOG (Programming in Logic)

Пролог була однією з перших логічних мов програмування, що застосовується для експертних систем, доведення теорем і різних логічних задач. Пролог – мова високого рівня, орієнтований на використання концепцій і методів математичної логіки, призначений для програмування в термінах логіки. Має вбудований механізм логічного виведення. Основною особливістю Прологу, що відрізняє його від всіх інших мов, є декларативний характер програм. Пролог-програма складається із фраз (речень). Фрази Прологу бувають трьох типів: факти, правила і питання.

***CLIPS* (C Language Integrated Production System)**

CLISP підтримує три основні способи подання знань:

- продукційні правила для подання евристичних знань, заснованих на досвіді;
- функції для подання процедурних знань;
- об’єктно-орієнтоване програмування.

JESS (Java Expert System Shell)

Це середовище для розробки експертних систем, яке є нащадком CLIPS і повністю написане під *JAVA. JESS* використовує розширену версію алгоритму *Rete* до процесу правил. *Rete* є ефективним механізмом для вирішення важких завдань типу "багато-до-багатьох".

Також має багато унікальних особливостей, включаючи зворотний ланцюжок, може безпосередньо керувати і оцінювати об’єкти *JAVA.*

Оболонки експертних систем

Оболонки експертних систем – програмний продукт, що має засоби подання знань для певних предметних областей.

Завдання користувача полягає не в безпосередньому програмуванні, а в формалізації і введенні знань з використанням наданих оболонкою можливостей.

Пусті (базові) оболонки не містять знань ні про яку Предметну Область.

Приклади:

EMYCIN – створена на базі експертної системи MYCIN,

CODEX,

DECTOOLS

HEARSAY II і ін.

Приклади експертних систем

MYCIN (Стенфордський університет, США) – одна з перших і найбільш відомих ЕС, розроблена в середині 70-х років двадцятого століття. Система

призначена для діагностики інфекційних захворювань. З її допомогою перевірялись і тестувались ідеї ЕС.

JUDITH – одна з перших юридичних ЕС, яка давала можливість юристам отримувати експертні висновки у цивільних справах. Розроблена в 1975 р в Гейдельберзькому та Дармштадтському університетах (Німеччина).

INTERNIST (США). ЕС діагностує кілька сотень захворювань з точністю, яка порівнянна з точністю діагнозу, зробленого кваліфікованим лікарем.

PROSPECTOR – експертна система, яка допомагає геологам в пошуку нових корисних копалин. На підставі інформації, введеної в ЕОМ з географічних карт, з оглядів і відповідей на питання, які задаються геологам, PROSPECTOR прогнозує місце розташування нових покладів. Використання цієї системи дозволило виявити поклади молібдену в Британській Колумбії (Канада).

DENDRAL – експертна система для визначення формули речовини по даним мас спектрографії.

Розглянемо основні складові експертної системи MYCIN.

Експертна система MYCIN

Предметна область. Первинні аналізи, взяті у пацієнта, направляють в мікробіологічну лабораторію, де з них вирощується культура бактерій. Іноді вже на ранніх стадіях можна зробити висновок про морфологічні характеристики мікроорганізмів. Але навіть якщо мікроорганізм, що викликав зараження, і ідентифікований, ще невідомо (або немає повної впевненості), до яких препаратів він чутливий.

«Антимікробний агент» – це будь-який лікарський препарат, створений для знищення бактерій і перешкоджання їх зростанню. Деякі агенти занадто токсичні для терапевтичних цілей, і не існує агента, який є ефективним засобом боротьби з будь-якими бактеріями. Вибір терапії при бактеріальному зараженні складається з чотирьох етапів:

- з'ясувати, чи має місце певний вид зараження у даного пацієнта;
- визначити, який мікроорганізм (мікроорганізми) міг викликати даний вид зараження;
- вибрати множину лікарських препаратів, придатних для застосування в даній ситуації;
- вибрати найбільш ефективний препарат або їх комбінацію.

Структура MYCIN наведена на рис.4.5.

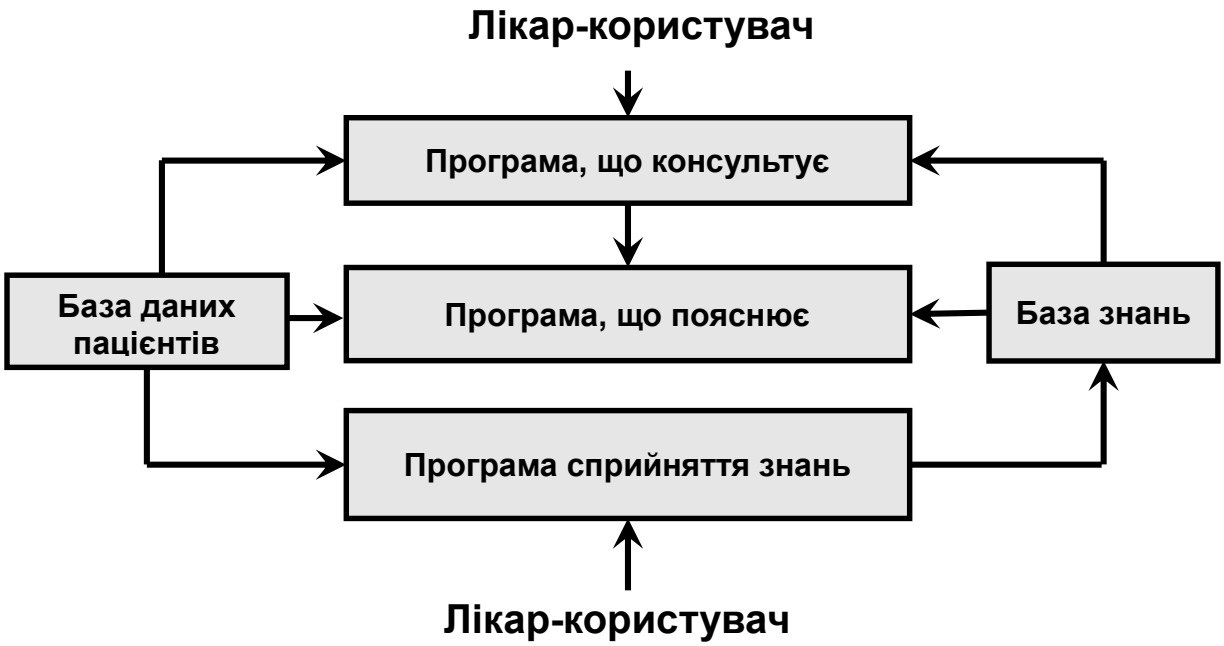


Рисунок 4.5 – Складові експертної системи MYCIN.

База знань містить фактичні знання, що стосуються предметної області, і відомості про наявні невизначеності.

Динамічна **база даних пацієнтів** містить інформацію про конкретних пацієнтів і їх захворюваннях рис 4.6.

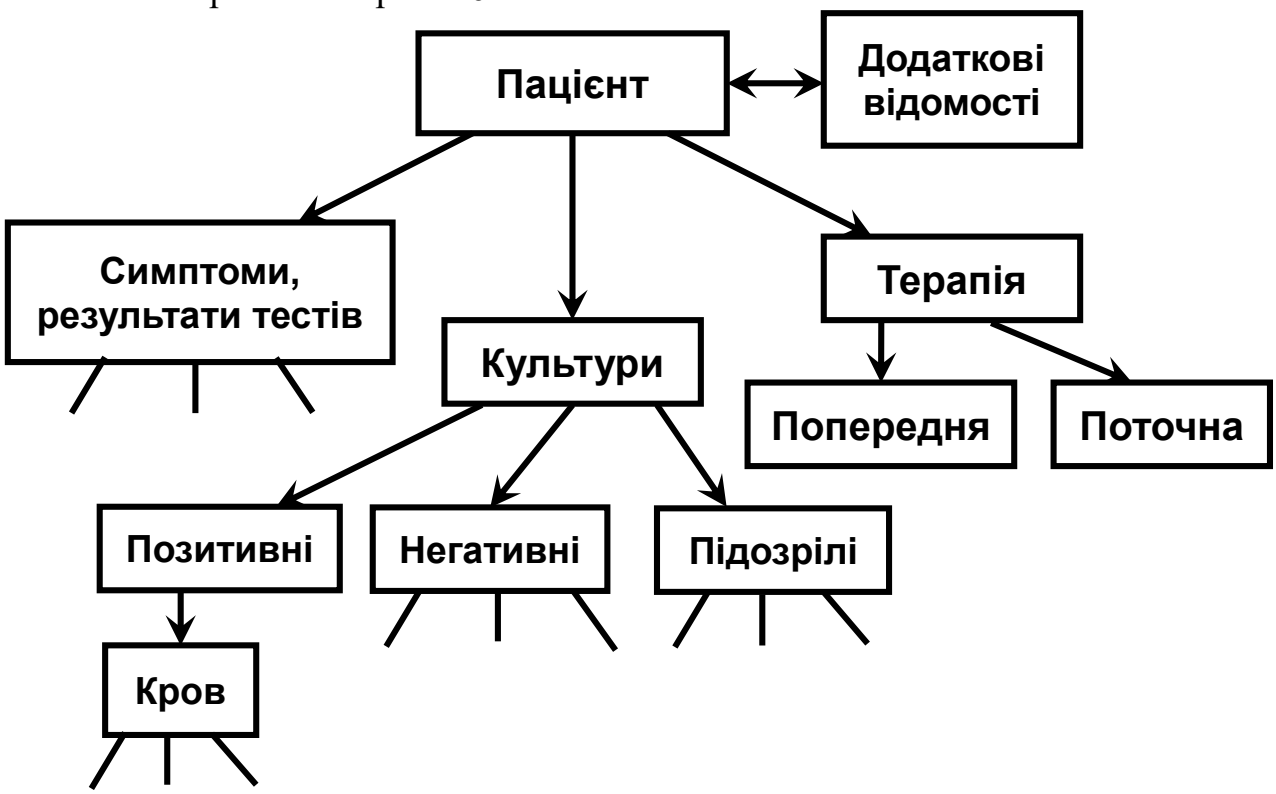


Рисунок 4.6 – Складові бази даних пацієнта експертної системи MYCIN.

Програма, що консультує, задає питання, виводить висновки системи і дає поради для конкретного випадку, використовуючи інформацію про пацієнта і статичні знання.

Програма, що пояснює, відповідає на питання користувача і дає інформацію про те, на чому ґрунтуються рекомендації або висновки, сформульовані системою. При цьому програма показує процес вироблення рекомендацій.

Програма сприйняття знань служить для оновлення знань, що зберігаються в системі, в процесі її експлуатації.

На рис. 4.7. представлені дані для лікування пацієнта.

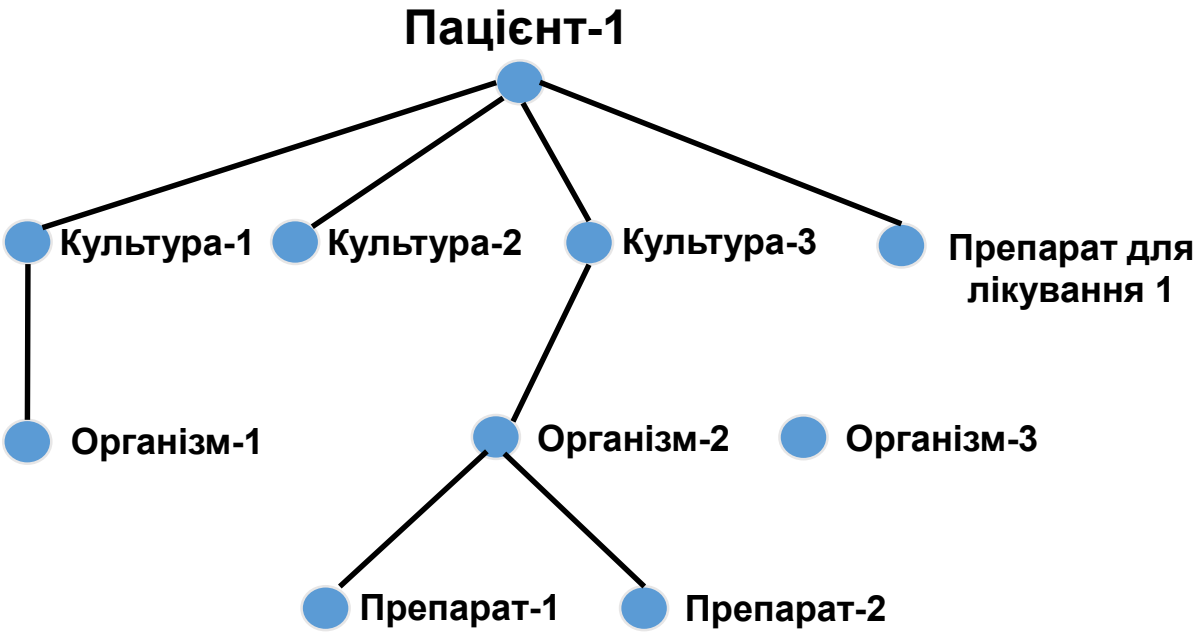


Рисунок 4.7 – Дерево даних пацієнта для його лікування.

Безпосередні дані (листя дерева) мають форму триплетів, кожен з яких складається з трьох компонентів:

- ❖ контекст,
- ❖ параметр,
- ❖ значення,
- ❖ показник визначення (ПВ) [–1, +1].

Наприклад:

Контекст: "Культура ©143 "

Параметр: "Анаеробність "

Значення: "Присутній "

ПВ: 0,48

Триплети, які не створені, автоматично мають ПВ = 0.

У це дерево включено три культури організмів (наприклад, отримані з аналізу крові пацієнта) і поточні призначення, які потрібно враховувати при аналізі, оскільки вони пов’язані з прийомом певних лікарських засобів. З культурами пов’язані мікроорганізми, присутність яких передбачається на підставі даних, отриманих в лабораторії, а з мікроорганізмами - лікарські засоби, які впливають на них.

Припустимо, що в запису, пов'язаному з вузлом ОРГАНІЗМ-1 в цій структурі, зберігаються дані:

ГРАН = (ГРАМ-НЕГ 1.0)

Морф = (ПАЛИЧКА .8) (КОКОН .2)
ПОВІТРЯ = (аероби .6),
які мають наступний зміст:

- зовсім виразно організм має грамнегативне забарвлення (1.0);
- зі ступенем впевненості 0.8 організм має форму палички, а зі ступенем впевненості 0.2 - форму колбочки;
- зі ступенем впевненості 0.6 ОРГАНІЗМ-1 є аеробним (тобто повітряне середовище сприяє його зростанню).

Виведення в MYCIN.

Приклад правила в ЕС MYCIN.

ЯКЩО організм має грамнегативне забарвлення і організм має форму палички і організм аеробний,

ТО є підстави припускати (0.8), що цей мікроорганізм відноситься до класу enterobacteriaceae.

Такого роду правила названі оргправилами (ORGRULES) і в них сконцентровані знання про такі організми, як strepococcus, pseudonomas і enterobacteriaceae.

Для виведення використовуються Три системні предикати (X - триплет):
SAME(X) = Істина, якщо | ПВ(X) | > 0,2
KNOW(X) = Істина, якщо | ПВ(X) | ≥ 0,8
DEFINITE(X) = Істина, якщо | ПВ(X) | = 1

Коефіцієнт впевненості (CF – certainty factor).

Застосовується при визначенні оцінки гіпотези захворювання, коли конкорують декілька правил для поточної умови.

CF (дія) = CF (передумова) x CPF (правило)

Мікроорганізм з прикладу ОРГАНІЗМ-1, відноситься до класу ентеробактерій зі ступенем впевненості, що дорівнює 0.6 x 0.8 = 0.48. Співмножник 0.6 – це ступінь впевненості у виконанні сукупності умов, перелічених у правилі, а 0.8 – ступінь впевненості в тому, що правило дає правильний висновок, коли всі зазначені в ньому умови гарантовано задовольняються. За співмножники і результатом цього виразу закріпився термін коефіцієнта впевненості (CF – certainty factor).

Комбінація гіпотез

В системі MYCIN може виявитися, що за судження про певний параметр відповідає не одне правило, а декілька. Застосування кожного з них (окрема гіпотеза) характеризується деяким значенням коефіцієнта впевненості.

Наприклад, з одного правила випливає, що даний мікроорганізм – це E.Coli, причому коефіцієнт впевненості цієї гіпотези дорівнює 0.8. Інше правило, беручи до уваги інші властивості аналізованого об’єкта, призводить до висновку, що цей мікроорганізм – E.Coli, але ця гіпотеза характеризується коефіцієнтом впевненості 0.5 (або, наприклад, – 0.8).

CF (X, Y) = X + Y – XY якщо X, Y> 0 (1)

CF (X, Y) = X + Y + XY якщо X, Y <0 (2)

CF (X, Y) =(X + Y)/(1 – min (|X|, |Y|)),

якщо (X > 0 і Y <0) або (X <0 і Y> 0), (3)

|X| означає абсолютне значення X

Якщо обидві гіпотези підтверджують висновок (1) або обидві його спростовують (2), то коефіцієнт впевненості їх комбінації зростає за абсолютною величиною.

Якщо ж одна гіпотеза підтверджує висновок, а інша його спростовує (3), то наявність знаменника згладжує цей ефект.

Якщо виявилося, що гіпотез кілька, то їх можна по черзі "пропускати" через ці формули.

Правила формулювання рекомендацій про курс лікування

Правила включають інформацію про чутливість організмів, відомих системі, до тих чи інших медикаментів.

ЯКЩО мікроорганізм ідентифікований як pseudomonas,

ТО рекомендується вибрати наступні медикаменти:

- COLISTIN (0.98)
- POLYMIXIN (0.96)
- GENTAMICIN (0.96)
- CARBENICILLIN (0.65)
- SULFISOXAZOLE (0.64)

Числа за назвою медикаментів, представляють оцінки ймовірності, що бактерія pseudomonas виявиться чутливою до цього препарату.

Препарат з цього переліку вибирається з урахуванням протипоказань, специфічних для кожного пацієнта. Користувач може ставити питання про альтернативні курси лікування до тих пір, поки система не вичерпає список ймовірних діагнозів.

Для визначення ефективності системи MYCIN були залучені 10 клінічних випадків і 8 експертів, які на підставі цих випадків робили висновок про діагноз. При цьому максимально можлива оцінка – 80 балів. Результати діагностування експертів і MYCIN такі:

MYCIN 52

Курс лікування, призначений в дійсності 46

Faculty-1 50

Faculty-4 44

Faculty-2 48

Resident 36

Inf dis fellow 48

Faculty-5 34

Faculty-3 46

Student 24

Неприйнятний курс лікування 0

Однакові курси лікування 1

4.7. Системи з дошкою оголошень

Corkill D. D. (1991). Blackboard systems. *Al Expert*, 6(9), p. 40-47.

Уявіть групу експертів, які сидять біля класної дошки (або великої дошки оголошень) і намагаються вирішити проблему.

Кожен експерт є фахівцем в якійсь певній області, що має відношення до проблеми.

Формулювання проблеми і вихідні дані записані на дошці.

Експерти допитливо вдивляються в те, що написано на дошці, і кожен з них думає над тим, чим він може допомогти у вирішенні проблеми.

Формулювання проблеми і вихідні дані записані на дошці.

Експерти допитливо вдивляються в те, що написано на дошці, і кожен з них думає над тим, чим він може допомогти у вирішенні проблеми.

Якщо хто-небудь з експертів відчуває, що йому є що сказати з цього приводу, він виконує відповідні обчислення і записує результат все на тій же дошці.

Цей новий результат може дозволити і іншим експертам надати певний внесок у вирішення проблеми.

Процес припиняється (а експерти розходяться по домівках), коли проблема буде вирішена.

Угоди між експертами:

всі експерти повинні говорити однією мовою, хоча при записі результатів на дошці можуть використовуватися і різні схеми позначень;

повинен існувати якийсь протокол визначення черговості "виступів", який вступає в силу в ситуації, коли відразу кілька експертів хапаються за крейду і направляються до дошки.

всі експерти повинні говорити однією мовою, хоча при записі результатів на дошці можуть використовуватися і різні схеми позначень;

повинен існувати якийсь протокол визначення черговості "виступів", який вступає в силу в ситуації, коли відразу кілька експертів хапаються за крейду і направляються до дошки.

Структурні компоненти:

Знання про предметну область розділені між незалежними джерелами знань (KS-knowledge sources), які працюють під управлінням планувальника (scheduler). Рішення формується деякій глобальній доступній структурі, що називають дошкою оголошень (blackboard). Таким чином, в цій системі всі знання "як поступити" будуть представлені не у вигляді єдиного набору правил, а у вигляді набору програм. Кожен з компонентів цього набору може мати у своєму розпорядженні власний набір правил або сумішшю правил і процедур.

Як правило, дошка оголошень розділяється на кілька рівнів опису, причому кожен рівень відповідає певній мірі деталізації.

Дані в межах окремих рівнів дошки оголошень представляють ієрархії об'єктів або графи, тобто структури складніші, ніж вектори, які

використовувалися в робочій пам'яті продукційних систем. У найсучасніших системах може бути навіть кілька дошок оголошень.

Джерела знань формують об’єкти на дошці оголошень, але це виконується за допомогою планувальника. Записи активізації джерел знань (knowledge source activation records) поміщаються в спеціальний список вибору (agenda), звідки їх витягує планувальник.

Джерела знань спілкуються між собою тільки через дошку оголошень і не можуть безпосередньо передавати дані одне одному або запускати виконання будь-яких процедур. Тут є певна аналогія з організацією роботи продукційних систем, в яких правила не можуть безпосередньо активізувати одне одного: все має відбуватися через робочу пам'ять.

ЕС HEARSAY-II

Приступаючи до розробки системи рис. 4.8, її творці прекрасно розуміли, що з кожним рівнем аналізу пов’язана окрема галузь знань – аналіз звукових сигналів, фонетика, лексичний аналіз, граматика, семантика, ораторське мистецтво.

Жодна з цих галузей окремо не здатна надати достатньо інформації для того, щоб вирішити проблему.

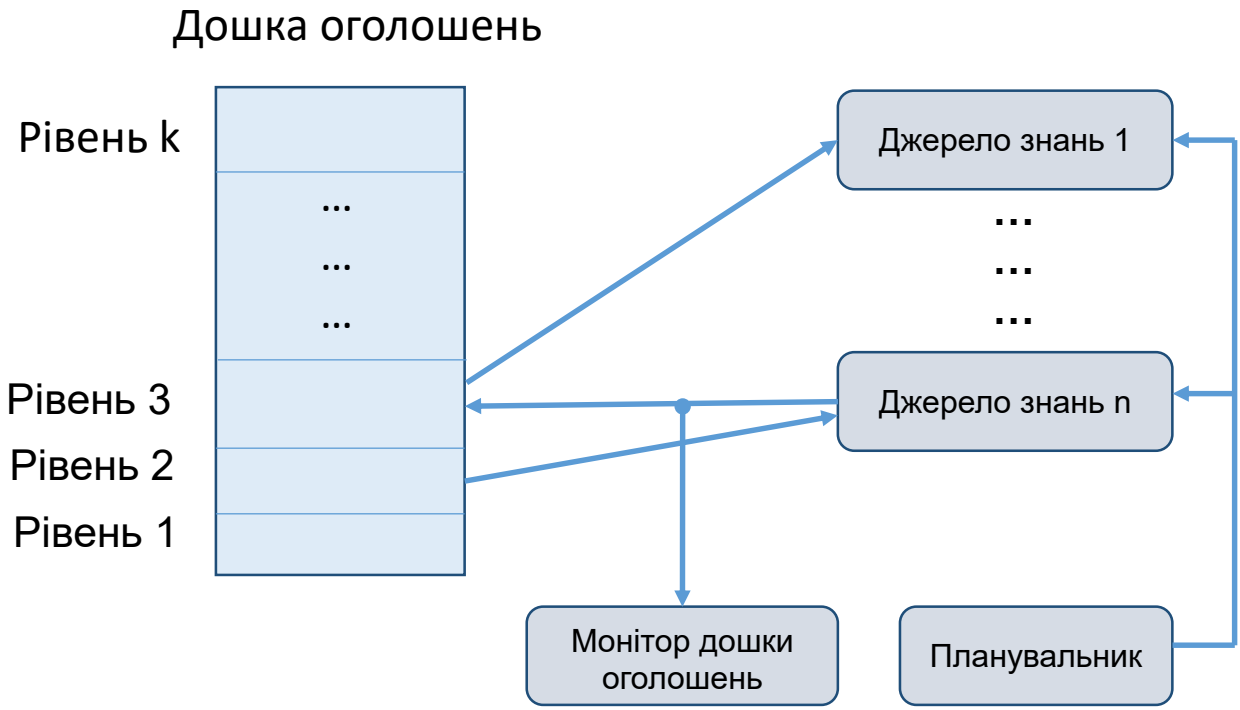


Рисунок 4.8 – Спрощена структурна схема системи HEARSAY-II

ЕС HEARSAY-II (Розпізнавання мови)

Уявімо, наприклад, що, користуючись методами обробки акустичних сигналів, ми змогли розкласти вихідний звук на фонеми. Але без додаткової інформації все одно не вдасться виділити сенс виразів, подібних наступним: *I scream (я вигукую)* і

ice cream (морозиво) або *please let us know (будь ласка, дайте нам знати)* і *please lettuce no* (будь ласка, – без салату) . Таким чином, хоча кожен окремий

вид (набір) знань грає істотну роль у вирішенні проблеми і кожен з них може бути представлений в програмі більш-менш незалежно від інших, автоматичне розпізнавання мови вимагає використання всіх цих знань спільно.

Уявімо, наприклад, що, користуючись методами обробки акустичних сигналів, ми змогли розкласти вихідний звук на фонеми. Але без додаткової інформації все одно не вдасться виділити сенс виразів, подібних наступним: *I scream (я вигукую)* і

ice cream (морозиво) або *please let us know (будь ласка, дайте нам знати)* і *please lettuce no* (будь ласка, – без салату). Таким чином, хоча кожен окремий вид (набір) знань грає істотну роль у вирішенні проблеми і кожен з них може бути представлений в програмі більш-менш незалежно від інших, автоматичне розпізнавання мови вимагає використання всіх цих знань спільно.

При розпізнаванні мови дослідникам доводиться стикатися ще з однією ключовою проблемою, – проблемою невизначеності:

- ❖ дані неповні і зашумлені;
- ❖ відсутня однозначна відповідність між даними на сусідніх рівнях; прикладом може служити відповідність між рівнями фонем і лексичних одиниць при аналізі фраз */ scream* і *ice cream*;
- ❖ важливу роль відіграють лінгвістичний і смисловий контексти;
- ❖ інтерпретація сусідніх елементів робить більш-менш імовірними різні варіанти інтерпретації поточного сегмента.

Більш традиційні підходи до розпізнавання мови засновані на використанні статистичних моделей з теорії передачі інформації для визначення кореляційного зв’язку між сегментами. Підхід, що базується на знаннях, зажадав істотного перегляду методів обробки невизначеності.

Джерело знань базі оболонки HEARSAY-III, повинно складатися з пускового зразка (trigger), первинної програми (immediate code) і тіла (body). Виявивши відповідність між поточним вмістом дошки оголошень і пусковим зразком, оболонка створює вузол записи активізації для цього джерела знань і запускає на виконання первинну програму. Через деякий час запис активізації джерела знань вибирається планувальником і тоді запускається на виконання тіло джерела знань, яке представляє собою програму на мові LISP.

До складу HEARSAY-III входить найпростіший планувальник, який виконує базові функції планування в експертній системі: вибір чергового запису в списку актуальних і запуск на виконання програмного коду відповідного джерела знань.

4.8. Переваги і недоліки експертних систем

Переваги:

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

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

- легкість передачі або відтворення: передача знань від однієї людини до іншої – довгий і дорогий процес, передача штучної інформації – це простий процес копіювання програми або файлу даних;
- підвищена доступність: експертна система – засіб масового виробництва експертних знань, що дозволяє багатьом користувачам одержати доступ до експертних знань;
- можливість одержання й об'єднання експертних знань з багатьох джерел: можуть бути зібрані знання багатьох експертів і притягнуті до роботи над задачею, виконуваної одночасно і безупинно у реальному часі; рівень експертних знань, скомбінованих шляхом об'єднання знань декількох експертів, може перевищувати рівень знань окремого експерта-людини;
- стійкість і відтворюваність результатів: експерт-людина може приймати в тотожних ситуаціях різні рішення через емоційні фактори або втому, у той час, як результати експертних систем стабільні і являють собою незмінно правильні, позбавлені емоцій і повні відповіді за будь-яких обставин;
- низька вартість: експерти, особливо висококваліфіковані, обходяться дуже дорого, у той час, як експертні системи, навпаки, є порівняно недорогими – їхня розробка є дорогою, але вони є дешевими в експлуатації: вартість надання експертних знань у розрахунку на окремого користувача істотно знижується;

Недоліки:

- експертні системи погано вміють: подавати знання про часові та просторові відносини, розмірковувати, виходячи зі здорового глузду, розпізнавати межі своєї компетентності, працювати із суперечливими знаннями;
- інструментальні засоби побудови експертних систем погано вміють: виконувати набуття знань, уточнювати бази знань, працювати зі змішаними схемами подання знань;
- побудова експертних систем не під силу кінцевому користувачу, який не володіє експертними знаннями про проблемну галузь;
- необхідність залучення людини-експерта з проблемної галузі, що є носієм знань; неможливість повного відмовлення від експерта-людини;
- можливі труднощі взаємодії експерта зі спеціалістом-когнітологом, який шляхом діалогу з експертом оформляє отримані від експерта знання в обраному формалізмі подання знань;
- необхідність повної переробки програмного інструментарію, у випадку, якщо наявна оболонка експертної системи та / або використовувана нею модель подання знань погано підходять для обраної проблемної галузі, задачі;
- тривалість процесів витягу знань з експерта, їхньої формалізації, перевірки на несуперечність і усунення протиріч.

5. МАШИННЕ НАВЧАННЯ

5.1 Загальні питання машинного навчання

Машинне навчання (МН) – це великий підрозділ штучного інтелекту, що вивчає методи побудови алгоритмів, здатних «навчатись». Термін «Машинне навчання» ввів американський вчений у галузі ШІ Arthur Samuel (Артур Самуель) в 1959 як «здатність комп'ютера вчитися, не будучи явно запрограмованим.

Мета МН – передбачити результат за вхідними даними. Чим різноманітніші вхідні дані, тим простіше алгоритму знайти закономірності і тим точніший результат.

На рис. 5.1 представлені основні складові машинного навчання.

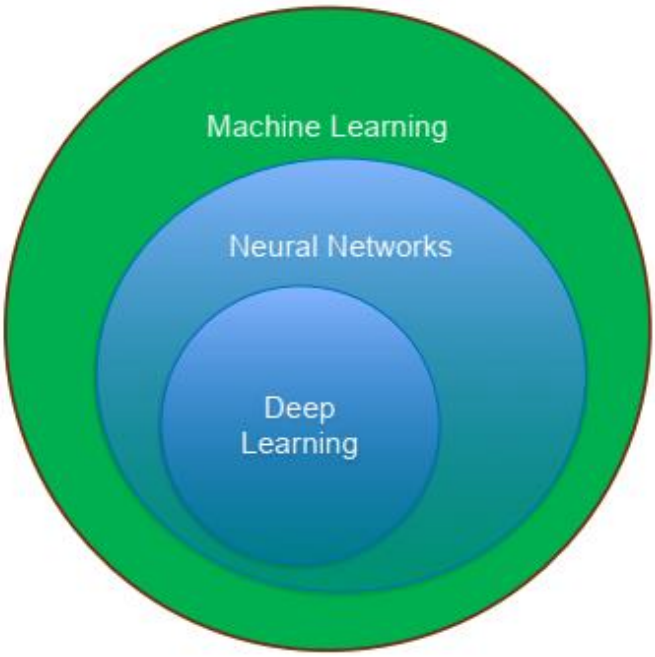


Рисунок 5.1 – Структура машинного навчання

МН включає класичну теорію розпізнавання образів, штучні нейронні мережі (ШНМ) і підгалузь ШНМ – глибоке навчання.

Три кити машинного навчання

Data (Дані). Хочемо виявляти спам – потрібні приклади спам-листів, передбачати курси акцій – потрібна історія цін, дізнатися інтереси користувача – потрібні його лайки або пости. Даних потрібно якомога більше.

Дані збирають як тільки можуть. Хтось вручну, даних менше, зате без помилок. Автоматично –все, що знайшлося. Великі компанії типу Google, використовують своїх же користувачів для безкоштовної розмітки. За хорошими наборами даних (datasets) йде велике полювання.

Features (Ознаки). Ознаками можуть бути: пробіг автомобіля, стать користувача, ціна акцій, навіть лічильник частоти появи слова в тексті. Алгоритму потрібно, з чим конкретно працювати. Добре, коли дані просто лежать в таблицях – назви їх колонок і є features. Коли ознак багато, модель працює повільно і неефективно. Найчастіше відбір правильних ознак займає більше часу, ніж все інше навчання. Але бувають і зворотні ситуації, коли

розробник сам відбирає тільки «правильні» на його погляд ознаки і вносить в модель суб'єктивність, що теж може призвести до неефективності моделі.

Algorithm (Алгоритм). Зазвичай, одну й ту ж задачу майже завжди можна розв'язати різними методами (способами). Від вибору методу залежить точність, швидкість роботи і розмір готової моделі. В кожній підгалузі МН алгоритми мають свою специфіку, можуть називатись правилами розв'язку в теорії розпізнавання образів або алгоритмами навчання в штучних нейронних мережах.

5.2 Типи машинного навчання.

Класифікація основних задач розпізнавання образів:

- Image recognition (Розпізнавання зображень).
- Speech recognition (Розпізнавання мовлення).
- Situation recognition (Розпізнавання ситуацій).
- State recognition (Розпізнавання станів).

Всередині цих основних напрямів теорії розпізнавання образів виділяються піднапрями і їх дуже багато, наприклад, в розпізнавання зображень виділяють:

- розпізнавання рентгенограм;
- розпізнавання раку шкіри;
- розпізнавання літер (Optical character recognition (OCR));
- розпізнавання штрих-кодів;
- розпізнавання автомобільних номерів;
- розпізнавання обличь;
- розпізнавання жестів;
- розпізнавання емоцій
- розпізнавання локальних ділянок земної кори

На рис. 5.2 подана класифікація МН за способом формування даних для алгоритму навчання.

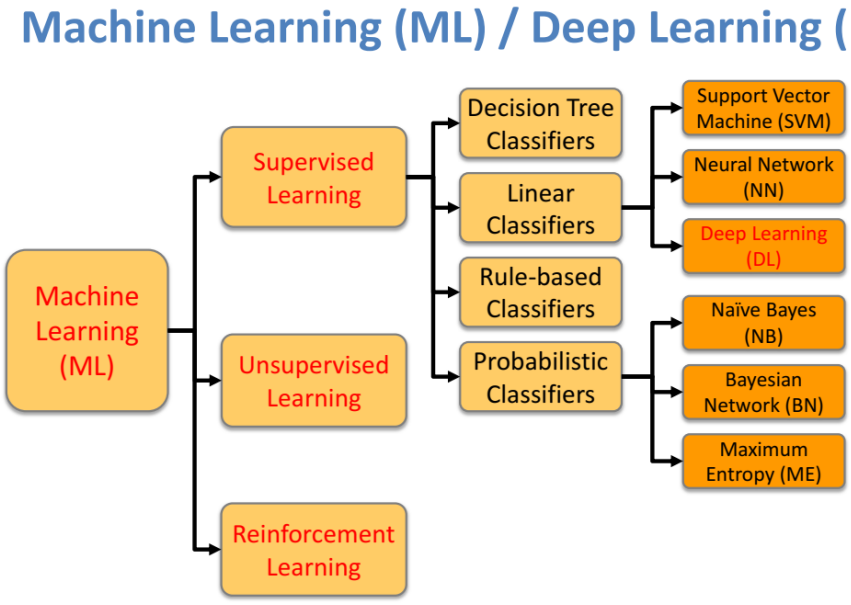


Рисунок 5.2 – Класифікація машинного навчання за типом даних навчання

1. Supervised Learning (Навчання з учителем)

У цьому випадку алгоритм має «наставника» – учителя, повідомляє йому як правильно вчинити. Вчитель заздалегідь відмічає всі потрібні дані і відповідний цим даним результат, щоб алгоритм навчився на конкретних прикладах. Наприклад, він показує, що на цій світлині автомобіль, на іншій – велосипед.

У термінах машинного навчання вчитель – це саме втручання людини в процес обробки інформації. З учителем машина вчиться набагато краще й швидше, тому для вирішення практичних завдань такі алгоритми використовують частіше.

2. Unsupervised Learning (Навчання без вчителя)

Завдання алгоритму при навчанні без вчителя – знайти зв'язки між окремими даними, виявити закономірності, підібрати шаблони, упорядкувати дані або описати їх структуру, виконати класифікацію даних. Навчання без вчителя використовується, наприклад, в рекомендаційних системах, коли в інтернет-магазині на основі аналізу попередніх покупок покупцеві пропонуються товари, які можуть зацікавити його з більшою ймовірністю, ніж інші.

У реальності добре розмічені дані – це велика рідкість, тому для їх розмітки зазвичай використовують або спеціальні сервіси, у яких реальні люди з країн з дешевою робочою силою (зазвичай Індії або Китаю) за мінімальну плату вручну класифікують дані, або спеціальні алгоритми для розмітки (які, в свою чергу, можуть також використовувати машинне навчання).

3. Reinforcement Learning (Навчання з підкріпленням)

Таке навчання є окремим випадком навчання з вчителем, але вчителем в даному випадку є «середовище». Алгоритм (в цій ситуації часто називають «агент») не має попередньої інформації про середовище, але має можливість сприймати це середовище через сенсори і здійснювати в ньому деякі дії. Середовище реагує на ці дії і тим самим надає агенту через сенсори агента дані, які дозволяють йому реагувати на них і вчитися. Фактично агент і середовище утворюють систему зі зворотним зв'язком.

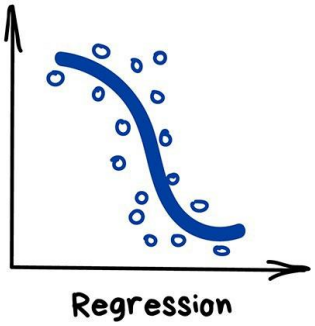
Навчання з підкріпленням використовується для вирішення більш складних завдань, ніж навчання з учителем і без вчителя. Воно використовується, наприклад, в системах навігації для роботів, які навчаються уникати зіткнень з перешкодами шляхом набуття досвіду, отримуючи зворотний зв'язок при кожному зіткненні. Навчання з підкріпленням використовується також в логістиці, при складанні графіків і плануванні завдань, при навчанні машини логічним іграм (покер, нарди, го і ін.).

5.3 Типи задач машинного навчання:

Регресія

Регресію дуже полюбляють аналітики та фінансисти, тому що вона може показати залежності між різними факторами процесу. Наприклад, як на ціну будинку впливає той чи інший район розташування? Або що більше впливає на вартість авто: рік випуску чи пробіг? Алгоритм намагається побудувати криву на

графіку, яка відображає залежність. Але, на відміну від людини з крейдою та дошкою, робить вона це з застосуванням математики.



Моделі регресійного аналізу зазвичай використовують, щоб показати або передбачити взаємозв'язок між процесом та тим, що цей процес може спровокувати. Тут варто пам'ятати, що така кореляція – не завжди причинність, тобто навіть пряма у простій лінійній регресії, яка добре відображає залежності між даними, може не дати конкретної відповіді про причинно-наслідковий зв'язок. Саме тому регресійний аналіз не використовують для **інтерпретації** причинно-наслідкових зв'язків між змінними.

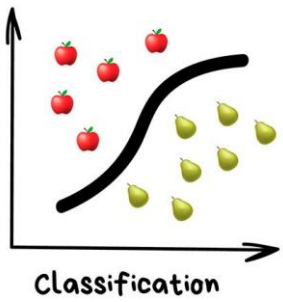
Алгоритмів регресійного аналізу існує багато, і використовують їх дуже давно. Сьогодні регресію використовують для:

- прогнозу вартості цінних паперів;
- аналізу попиту, обсягу продажів;
- медичних діагнозів;
- будь-яких залежностей даних від часу (часові ряди).

Популярні алгоритми: Лінійна, Поліноміальна Регресія, Логістична Регресія.

Класифікація (розпізнавання образів)

Алгоритми класифікації дозволяють розділити об'єкти відповідно до зазначених заздалегідь класів, наприклад розділити котів і собак, музику за жанром, шкарпетки за кольорами. Сьогодні алгоритми класифікації використовують для великої кількості завдань: визначення мови, спам-фільтрів, визначення шахрайства (коли зловмисник сплачує за послуги вкраденими коштами), пошуку схожих документів тощо.



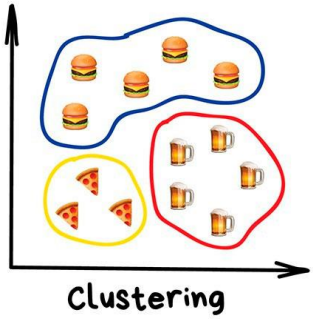
Щоб класифікація спрацювала, потрібно мати розмічені дані з категоріями і ознаками, які алгоритм буде вчитися визначати. Залежно від певних ознак алгоритм визначає, до якого з класів можна віднести об'єкт.

Найпростішим із технічного погляду завданням класифікації є бінарна класифікація – коли об'єкти потрібно розподілити між двома класами. Наприклад, на вході є транспортні засоби, та ми знаємо, що це або автомобілі, або велосипеди. У багатокласовій класифікації кількість класів може досягати декількох тисяч, і рішення стає значно складнішим. Також бувають класи, що перетинаються, – у таких випадках об'єкт може одночасно належати до декількох класів, і нечіткі класи – коли належність до того чи іншого класу визначається ступенем (зазвичай від 0 до 1).

Популярні алгоритми: Наївний Баєс, Дерева Рішень, K-найближчих сусідів, Метод Опорних Векторів.

Кластеризація

Алгоритми кластеризації дозволяють розділити об’єкти у випадку, коли класи заздалегідь не зазначені, а кластери мають бути сформовані за схожістю елементів. Алгоритми кластеризації використовують у завданнях сегментації ринку, для аналізу нових даних, стиснення зображень.



Алгоритм визначає схожі ознаки у об’єктів та групує їх у кластери. Кількість кластерів, як із класами, може бути визначена дослідником або самою машиною. Також дослідник може задати ознаки, за якими потрібно розділити вибірку, або машина визначить їх сама.

Кількість кластерів можна задати вручну або довірити це алгоритму. Теоретично, об’єкти, які перебувають в одному і тому самому кластері, повинні мати схожі властивості і/або особливості, тоді як об’єкти в різних групах повинні мати дуже різні властивості і/або особливості.

Прогнозування

Прогностичних моделей існує багато, і всі вони вирішують завдання передбачення часових рядів – знаходження майбутніх значень залежно від часу. Алгоритм при побудові прогнозу включає в себе наступні елементи:

а) Аналіз часової послідовності на предмет наявності пропущених значень і значень, що випадають, та корекція цих значень.

б) Визначення наявності тренду і його типу. Визначення періодичності в послідовності.

в) Перевірка послідовності на стаціонарність, тобто що процес у майбутньому буде розвиватися так само, як в минулому і сьогоденні. У реальній природі таких процесів не існує, але процеси, характеристики яких змінюються дуже повільно, можна зарахувати до стаціонарних.

ґ) Аналіз послідовності на предмет необхідності попередньої обробки. Усі дані повинні мати однакові характеристики й не відрізнятися один від одного.

д) Визначення параметрів моделі. Прогноз на підставі обраної моделі.

е) Оцінка точності прогнозування моделі.

є) Аналіз похибки обраної моделі.

ж) Визначення адекватності обраної моделі і, якщо результат виявиться незадовільним (недостовірним чи не достатньо точним), її заміна і повернення до попередніх пунктів.

Прогнозування використовують у медичній діагностиці, оцінці кредитоспроможності, передбаченні попиту, під час прийняття рішень на фінансових ринках.

Зменшення розмірності

Зменшення розмірності – це зведення більшого числа ознак до меншого для зручності їх подальшого використання. Використовується для побудови рекомендаційних систем, пошуку схожих документів чи візуалізації. Практична користь методів зменшення розмірності в тому, що можна отримати абстракцію, об’єднавши кілька ознак в одну.

Ще одне застосування завдань зменшення розмірності – рекомендаційні системи, алгоритми, що намагаються передбачити, які об’єкти будуть цікаві користувачеві, маючи певні дані про його профіль. Такі системи не враховують оцінку всіх користувачів, а спираються лише на переваги самого користувача або користувачів зі схожим портретом, і пропонують вам, приміром, подивитися мультфільм, серіал про маніяка або фільм про пошук дзену.

Виявлення аномалій

Мета – виявити аномальні відхилення від стандартних випадків. На перший погляд, завдання дуже схоже на завдання класифікації, проте воно має істотну відмінність: аномалії – явище рідкісне, тому вибірок, на яких можна навчити машинну модель або дуже мало, або немає зовсім. Саме тому методи класифікації тут не працюють. На практиці аномалії допомагають виявити шахрайство в банку, медичні проблеми або помилки в тексті.

Пошук правил

Завдання пошуку правил шукає закономірності в потоці даних. Істотним є передбачення поведінки користувача на інтернет-ресурсах. За яким товаром він повернеться? Які товари можна продати «навздогін» до вже заданого? На які розділи сайту направити користувача, щоб він залишив у вас більше грошей? На ці та інші питання може відповісти машинне навчання, коли алгоритм проаналізує інших користувачів, їхній досвід та поведінку.

5.4. Розпізнавання образів (Pattern recognition)

Задача розпізнавання образів (РО) – це задача віднесення вхідних даних до певного класу за допомогою виділення істотних ознак, що характеризують ці дані, із загальної маси несуттєвих даних або автоматичне розпізнавання шаблонів і закономірностей у даних. Іноді мова йде про теорію розпізнавання образів – напрям штучного інтелекту, що створює теоретичні основи класифікації та ідентифікації предметів, явищ, процесів, сигналів, ситуацій тощо. Віднесенням об’єкта до того чи іншого класу може бути, наприклад, задача розпізнавання літер або прийняття рішення про наявність дефекту у деякій технічній деталі. Віднесення об’єкта до певного класу відображає найтиповішу проблему класифікації, і, коли говорять про розпізнавання образів, найчастіше мають на увазі саме цю проблему.

Терміни теорії розпізнавання образів

Поняття класу

Класом у теорії розпізнавання образів прийнято називати сукупність об’єктів, які мають ті чи інші спільні ознаки. Про ознаки мова йшла в попередньому розділі. Клас може об’єднувати фізично існуючі сутності (наприклад, людина, яблуко) або бути абстрактним поняттям (горе, економічний крах і т.і.). Ознаки, що дають можливість відрізнити представників одного класу від іншого, прийнято називати інформативними ознаками. Ознаки, спільні для всіх представників класу, називатимемо інваріантами класу.

Іноді вживають більш формальне визначення класу: класом називається сукупність об’єктів, пов’язаних між собою деякими відносинами еквівалентності, або, в крайньому разі, толерантності. Відносинами

еквівалентності називається відношення, яке є симетричним, рефлексивним і транзитивним.

Можна виокремити такі основні властивості класів:

1. Усі представники класу мають певний набір спільних ознак (випливає з визначення).

2. Змінюваність реалізацій класів. По-перше, різні об'єкти, що належать одному класу, можуть бути не схожими між собою. Вони повинні мати спільні інваріантні ознаки, але всі інші ознаки можуть як завгодно варіювати. По-друге, один і той самий об'єкт може змінюватися з часом і навіть поступово переходити від одного класу до іншого (наприклад, перетворення пуголовка на жабу). Усе це свідчить про те, що розпізнати чіткі межі класу часто неможливо.

3. Ознайомлення з деякою скінченою кількістю представників одного класу дає можливість впізнавати інших представників цього класу. Взагалі кажучи, ця властивість може не виконуватися. Але якщо ця властивість виконується (принаймні, теоретично), це є дуже сильним і важливим твердженням. Воно по суті означає можливість навчатися на прикладах, тобто на основі спостереження певної кількості прикладів (можливо, разом з контрприкладами), сформулювати правило розпізнавання, яке дає змогу відрізняти представників даного класу від представників іншого (можливо, з певною достовірністю тобто з певним процентом помилок).

Якщо навчання на прикладах неможливе або неефективне, правило розпізнавання інколи можна задати явно. Якщо це зробити не вдається, єдиною можливістю для надійного розпізнавання залишається запам'ятовування всіх можливих представників даного класу. Цей випадок не становить інтересу з теоретичної точки зору, і його можна реалізувати лише, якщо кількість можливих представників не є надто великою.

Означення 1. Образом (*Pattern*) називається модель, яка відображає властивості об'єкта, що розпізнається. Образ характеризується множиною ознак розпізнавання, які утворюють структурований вектор-реалізацію образу. Математично образ розглядається як точка в N-вимірному просторі, де N – кількість ознак образу.

Означення 2. Ознакою (*Feature*) розпізнавання називається характеристика певної властивості об'єкта.

Означення 3. Вектором-реалізацією образу називається структурована послідовність ознак розпізнавання, що подається у вигляді вектора-рядка.

$x=(x_1, x_2, \ldots, x_N)$

або вектора-колонки

$x=(x_1, x_2, \ldots, x_N)^T$

Означення 4. Правилом розв'язку (classifier) називається математичний вираз або алгоритм визначення належності реалізації образу одному з класів розпізнавання із заданого алфавіту.

Означення 5. Міра відстані (метрика, міра близькості) відстань між двома об'єктами в N -вимірному просторі ознак. В алгоритмах розпізнавання найчастіше використовуються такі міри:

Евклідова $d(x_m, x_l) = \sqrt{\sum_{1=1}^{N} (x_m^i - x_l^i)^2}$

Манхеттенська $d(x_m, x_l) = \sum_{i=1}^{N} \left| x_m^i - x_l^i \right|$

Узагальнена $d(x_m, x_l) = \sqrt[r]{\sum_{i=1}^{N} \left(x_m^i - x_l^i \right)^r}$

Означення 6. Помилка 1-го роду – це віднесення образу не до того класу, до якого він належить в дійсності.

В задачі діагностики має назву «Помилкова тривога».

Приклад: авторизований *користувач* класифікується як *порушник*.

Означення 7. Помилка 2-го роду – це віднесення образу до якогось певного класу, до якого він насправді не належить.

В задачі діагностики має назву «Пропуск цілі».

Приклад: *порушник* класифікується як авторизований *користувач*.

З наведених означень випливає, що в просторі ознак клас є множиною точок. Якщо в просторі ознак ми маємо два класи, то правилом розв'язку буде гіперплощина або інша поверхня, що розділяє ці класи. Тепер, коли ми хочемо визначити до якого класу належить невідомий об'єкт необхідно визначити по який бік поверхні знаходиться цей об'єкт.

Роздільність класів розпізнавання

Гіпотеза компактності

В задачах класифікації це – припущення про те, що образи схожих об'єктів набагато частіше знаходяться в одному класі, ніж в різних; або – класи утворюють компактно локалізовані підмножини в просторі образів об'єктів (рис.5.3). Це також означає, що межа між класами має досить просту форму.

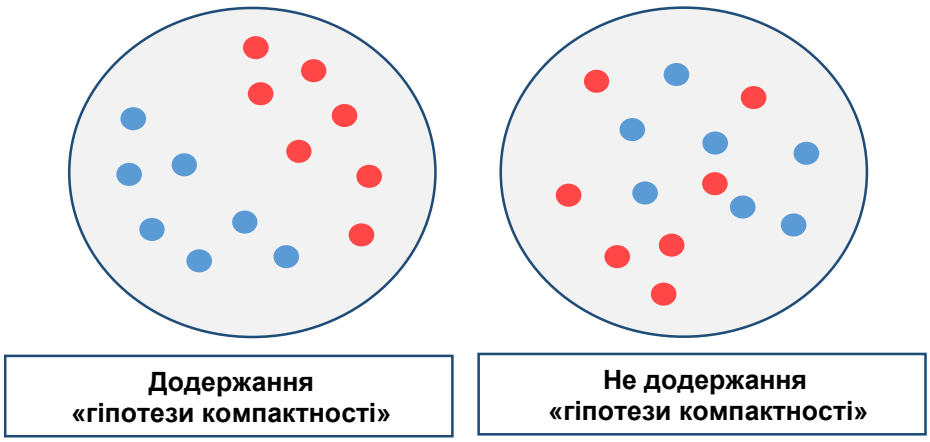


Рисунок 5.3 – Ілюстрація гіпотези компактності

5.5. Постановка задачі розпізнавання.

Загальна постановка задачі розпізнавання

Знайти правило розв'язку, що відносить образ об'єкту або процесу у виді істотних ознак, що характеризують цей об'єкт або процес до певного класу.

Математична постановка задачі розпізнавання

Нехай Ω – простір образів;

$\omega \in \Omega$ – образ;

$M = \{1, 2,..., m\}$ – номери класів Ω_1 , Ω_2 , Ω_m, таких що $\Omega_i \cap \Omega_j = \emptyset$, якщо $j \neq i$ та $\bigcup_{i=1}^{m} \Omega_i = \Omega$;

$p: \Omega \rightarrow M$ – правило розв'язку, що є невідомим;

X – простір ознак, тобто векторний простір, точками якого є вектори ознак образів;

$x: \Omega \rightarrow X$ – функція, що ставить у відповідність образу ω його вектор ознак $x(\omega)$.

Задача розпізнавання з учителем (supervised classification) полягає у тому, щоб на підставі множини прецедентів (p_j, x_j), $j = 1, \ldots, N$, яка називається навчальною вибіркою (training sample) побудувати правило розв'язку p, що мінімізує кількість помилок.

Алгоритм розпізнавання представляється як абстрактна функціональна система R з трьох компонент:

$R = \{\Omega, X, P\}$,

де

$\Omega = \{\Omega_k\}, k = 1, \ldots, K$ – алфавіт класів – множина категорій за якими розподіляються образи,

$X = \{x_j\}, j = 1, \ldots, N$ – словник ознак – множина характеристик, з яких складається опис образу,

$P = \{p_i\}, i = 1,...,L$ – множина правил розв'язку (прийняття рішень).

Функціонування системи розпізнавання зводиться до наступного:

- на вхід подається образ – деяка конфігурація з елементів множини X;
- до X застосовується певна послідовність правил з P;
- в результаті конфігурація позначається індексом, що відповідає одному з елементів множини Ω.

Якість функціонування системи визначається тим, наскільки часто присвоєний образу індекс збігається з очікуваним результатом.

Найчастіше система розпізнавання образів за аналогією з біонічною системою функціонує в двох режимах:

- навчання;
- класифікації (інтерпретації).

Процес створення системи розпізнавання образів

Для розпізнавання образів необхідно:

- визначити множину ознак, які відображають істотні властивості об'єктів, що будуть розпізнаватись;

• визначити класи об’єктів, до яких треба відносити образи при розпізнаванні;

• на множині відомих образів визначити правило розв’язку, що відносить реалізацію образу до відповідного класу (навчити систему розпізнавання).

Процес створення системи розпізнавання образів складається з таких кроків (рис.5.4):

• Вибір ознак (feature selection) – визначення найбільш інформативних ознак образів, які відображають істотні властивості об’єктів (в цей набір можуть входити функції від ознак).

• Генерування ознак (feature generation) – вимірювання або обчислення числових ознак, що характеризують об’єкт.

• Побудова класифікатора (classifier construction) – конструювання правила розв’язку, на підставі якого здійснюється класифікація.

• Оцінка якості класифікації (classifier estimation) – обчислення показників правильності класифікації (точність, чутливість, специфічність, помилки першого та другого роду).

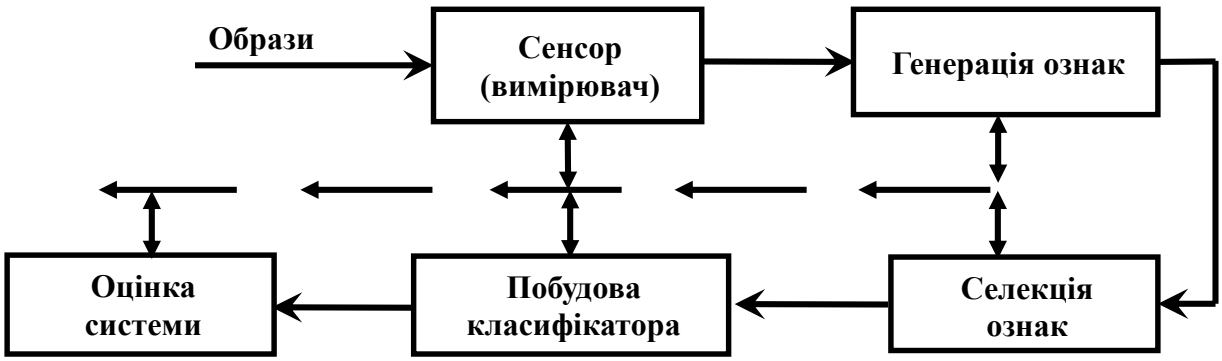


Рисунок 5.4 – Процес створення системи розпізнавання образів

5.6. Методи теорії розпізнавання.

Класифікація методів розпізнавання за математичною моделлю:

1) Алгебраїчний

Основа – прості правила розв’язку у виді нерівностей, що зв’язують ознаки. Недолік – невисока достовірність розпізнавання, оскільки не враховує неконтрольовані фактори.

2) Геометричний

Основа – образ представляється точкою в n-вимірнім просторі ознак. Класи в n-вимірнім просторі розділяються гіперплощинами. Належність до класу визначається відстанню образу до центру класу або до гіперплощини. Характеризується універсальністю, наочністю та простотою інтерпретації алгоритмів розпізнавання.

3) Статистичний

Базується на статистичних характеристиках даних. Використовується формула Байеєса.

4) Біологічний

Штучні нейронні мережі. Алгоритми в рамках цього підходу моделюють когнітивні процеси, що відбуваються у нервових клітинах мозку. Недолік – висока чутливість до багатовимірності простору ознак розпізнавання.

5) Мережевий

Семантичні мережі, фрейми, мережі Петрі, дерево рішень тощо.

Перевага – простота моделі, можливість її розширення та ускладнення.

Недолік – складність побудови правил розв'язку.

6) Нечіткий

Дозволяє моделювати процеси розпізнавання образів, що апріорно перетинаються в просторі ознак розпізнавання. Але не пристосований до оптимізації параметрів функціонування системи розпізнавання;

Класифікація методів розпізнавання за типом даних

Методи порівняння з еталоном

Застосовуються у випадках, коли кожному класу Ω_k можна зіставити кінцевий набір еталонних образів

$E_k = \{e_m, m = 1, ..., M_k\}$.

Процес розпізнавання полягає в зіставленні образів, що надходять на вхід пристрою розпізнавання або алгоритму, з еталонами E_k класів Ω_k, на основі обраної міри схожості (близькості).

Методи загальних властивостей

Застосовується в тих випадках, коли множина образів кожного класу занадто велика, щоб отримати надійний опис кінцевого числа еталонів, але можна виявити достатню кількість відмінних рис класів за кінцевими вибірками образів. Виявлені властивості кодуються на основі відповідної моделі і зберігаються в пам'яті у вигляді деяких структур, функцій або відносин.

Структурний метод

При структурних методах, реалізації образу описуються не множиною їх значень (навчальна вибірка типу об'єкт-властивість) або їх відношень (навчальна вибірка типу відношення-подібність), а їх структурою.

Структурні методи використовуються при сегментації (визначенні меж) різних текстів, при усномовному розпізнаванні за послідовністю фонем, тощо.

Головна перевага:

можливість подати велику кількість реалізацій у вигляді малопотужної множини непохідних елементів і граматичних правил.

Недоліки:

- відсутність прямих вирішальних правил;
- обмеженість використання через те, що аналізується не весь образ, а лише його фрагмент, що ускладнює процес прийняття рішень.

Метод січних площин

Алгоритм навчання, заснований на методі січних площин, полягає у відокремленні образів за допомогою частин гіперплощин. Алгоритм складається з наступних етапів.

1. Навчання (формування гіперплощин для відокремлення множин):

a. проведення січних площин;
b. вилучення зайвих гіперплощин;
c. вилучення зайвих частин площини.

2. Розпізнавання нових об’єктів.

Припустимо, що потрібно навчити комп’ютер розпізнавати 3 образи *a*, *b*, та *c*.

Проведення січних площин. У комп’ютер вводять коди двох точок (рис. 5.5), які належать різним образам та проводять довільну пряму, яка їх відокремлює.

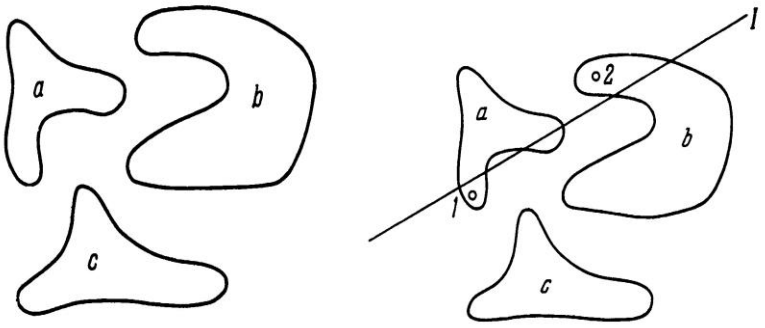


Рисунок 5.5 – Проведення січної площини

Далі береться третій об’єкт (рис. 5.6) і перевіряється правильність класифікації відносно проведеної прямої.

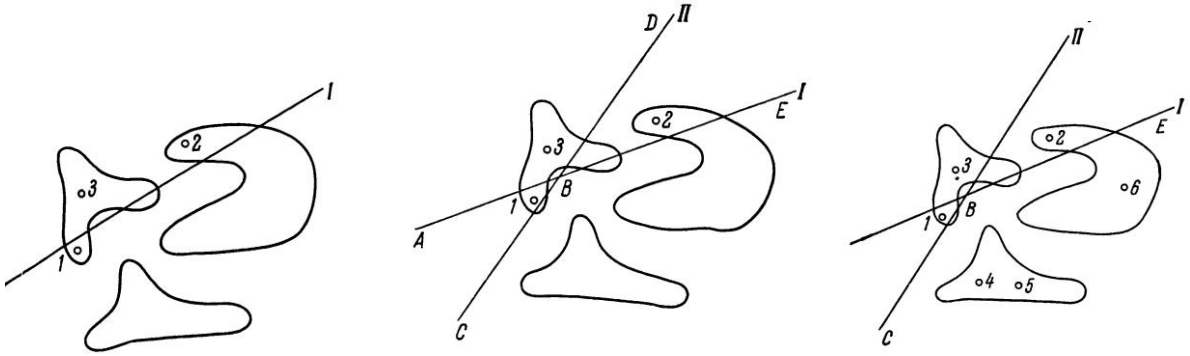


Рисунок 5.6 – Проведення другої січної площини

Оскілки третя точка належить тій самій півплощині, що й друга, то приходимо до висновку, що поточний класифікатор працює невірно. Для відокремлення точок 2 та 3 проводиться пряма II. Після цього площина розбивається на 4 частини. Ті частини, у який лежать об’єкти 1 та 3 (області *ABC* та *ABD*) відносимо до класу *a*, область *DBE* — до класу *b*.

При перевірці нової точки можливі три випадки:

1) виникає протиріччя (як це було для точки 3);
2) протиріччя не виникає, точка потрапляє у свою частину простору;
3) протиріччя не виникає, точка потрапляє у вільну (непозначену) частину простору (точки 4 і 5); У третьому випадку (точки 4 та 5 на рис. 5.6) машина відносить вільну частину площини до відповідного образу.

Для відокремлення точки 6 (образ *b*) від точок 4 та 5 (образ *c*) потрібно провести дві нові прямі (прямі III та IV на рис. 5.7).

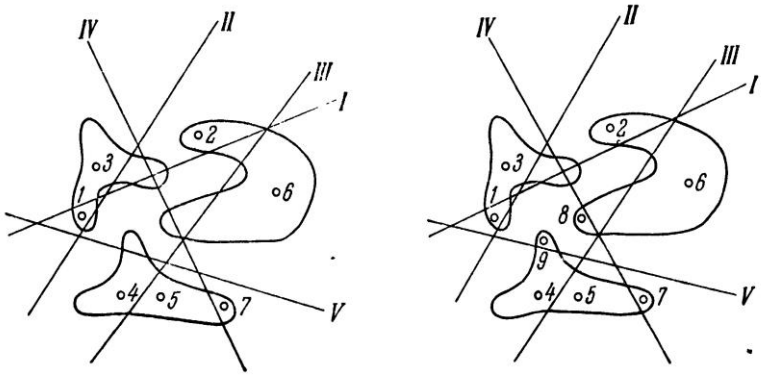


Рисунок 5.7 – Проведення нових січних площин

Подальші кроки вищенаведеного процесу навчання наведені на рис. 5.8., де вказано проміжкові та кінцевий результат стадії а). Незафарбовані ділянки не відносяться розпізнавальною системою до жодного із трьох наведених образів

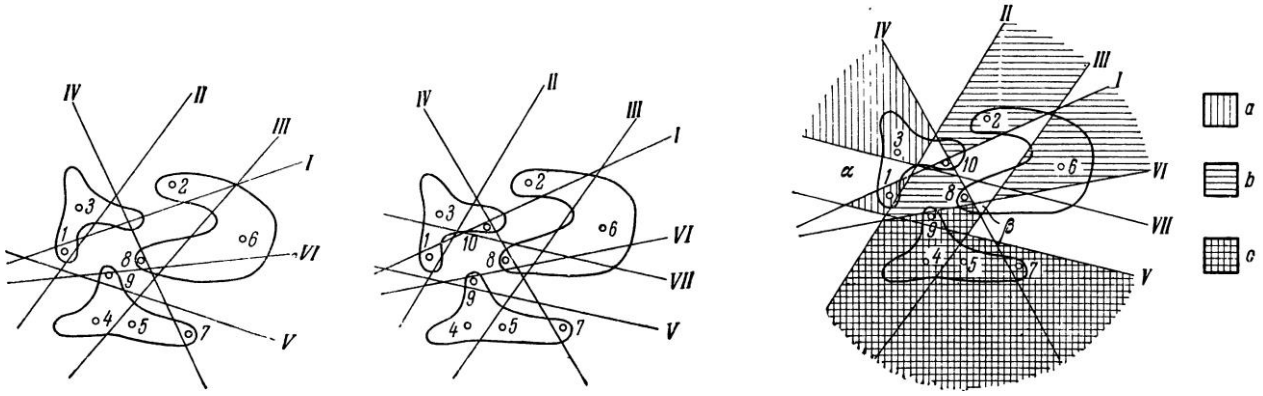


Рисунок 5.8 – Результат проведення площин

Видалення зайвих площин. З рис. 5.8 видно, що гіперплощини (прямі) I, III та V можуть бути вилученні цілком, оскільки вони не містять частин, відкидання яких призводить до появи протиріч. Наприклад, пряма I використовується лише для відокремлення об’єктів 1 та 2 — представників різних образів *a* та *b*. Але ці об’єкти відокремлює також пряма IV (або пряма VI). Розбиття після відкидання зайвих площин наведено на рис. 5.9. Слід зазначити, що у результаті частка вільних частин площини зменшилися.

Проте не можна вважати, що процес навчання завершений, оскільки ще залишилися "порожні" частини площини. Для їх вилучення використовується гіпотеза компактності. Природнім є віднести "порожні" області до того образу, що і яка-небудь суміжна зафарбована частина. Цей процес називають процесом вилучення зайвих частин. Для деяких вільних частин площини цей процес однозначний (область α на рис. 5.8), для інших — ні (усі області на рис. 5.9). Рекомендується проводити це вилучення поступово проглядаючи усі гіперплощини (прямі на рис. 5.9) та "порожні шматки", які до них прилягають.

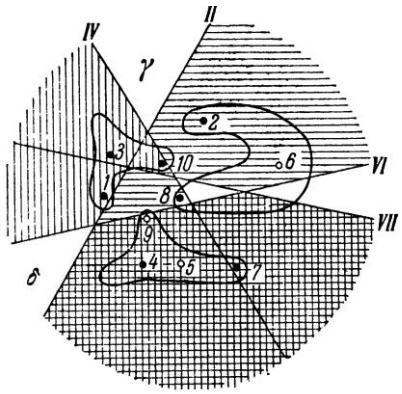


Рисунок 5.9 – Вилучення зайвих частин

Тому спочатку виконуємо перевірку усіх частин площини II, потім усіх частин площини IV і т.д. Кінцевий результат наведений на рис 5.10.

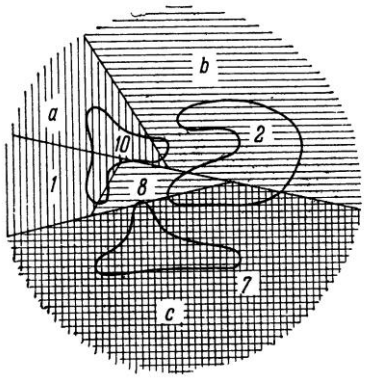


Рисунок 5.10 – Кінцевий результат злиття

Метод потенціалів для розпізнавання образів

Точковий електричний заряд у однорідному середовищі створює електричне поле, зображене на рис. 5.11. Радіальні лінії — це силові лінії поля, концентричні кола – лінії однакового потенціалу.

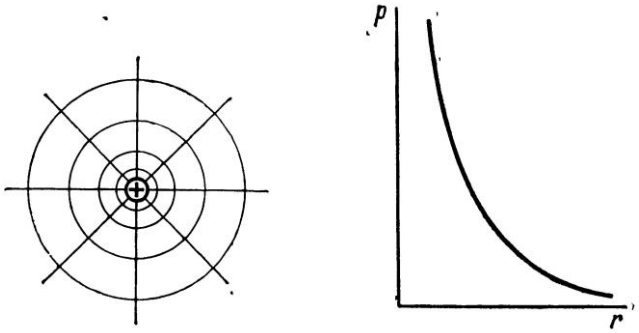


Рисунок 5.11 – Електричне поле точкового заряду

Потенціал p у кожній точці простору визначається співвідношенням

$$p = a\frac{q}{r^2},$$

де a – деяка стала, q – величина заряду, r – відстань від заданої точки до заряду.

Крива зміни потенціалу як функції відстані наведена на рис. 5.11. Потенціал зменшується по мірі віддалення від його джерела. Якщо поле утворене кількома зарядами, то потенціал в кожній точці рівний сумі потенціалів, які створюються кожним із зарядів.

Припустимо, що у просторі розташовані дві компактні групи зарядів. У одній групі — позитивні, у другій — негативні. На рис. 5.12 показаний розподіл потенціалів у околі цих зарядів, на рис. 5.13 ці потенціали алгебраїчно просумовані.

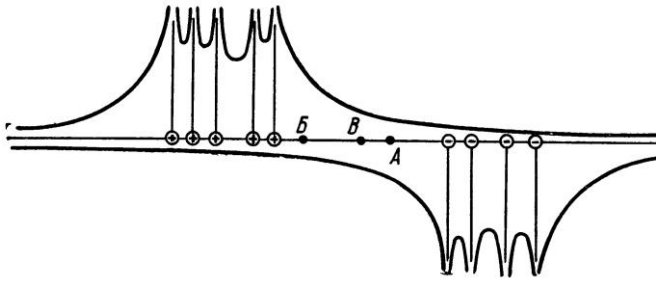


Рисунок 5.12 – Розподіл потенціалів кількох зарядів

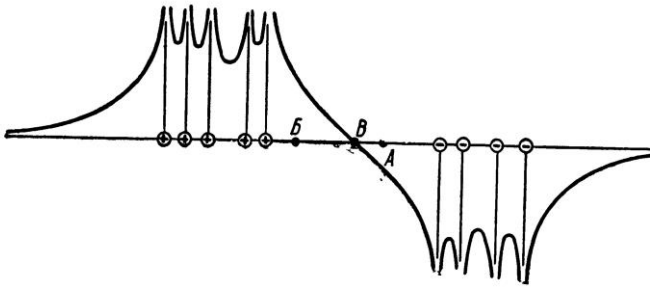


Рисунок 5.13 – Результат підсумовування потенціалів

Точку можна віднести до тієї чи іншої множини точок у залежності від того, який знак має сумарний потенціал поля у цій точці.

Вищенаведені міркування по аналогії можна перенести на точки простору рецепторів. Кожній точці, яка з'являється у процесі навчання, поставимо у відповідність деяку функцію, аналогічну по формі до електричного потенціалу. Такою функцією може бути, наприклад, функція

$$\varphi(D) = \frac{1}{1+\alpha D^2},$$

де α – деякий коефіцієнт, D – відстань (у деякій метриці) між точкою-джерелом потенціалу та точкою, у якій обчислюється потенціал. Наприклад, у якості D можна використовувати евклідову відстань, віддаль Хеммінга, манхеттенську метрику, тощо.

Нехай джерелами потенціалів є група точок, які відповідають деякому образу a. Тоді можна вважати, що середній потенціал, які створюють у деякій точці простору джерела цього образу, характеризує віддаль від цієї точки до усього образу у цілому.

Спираючись на гіпотезу компактності, можна запропонувати наступне правило розпізнавання: точку відносимо до того образу, середній потенціал якого у цій точці є максимальним.

Алгоритм розпізнавання на основі методу потенціалів

1. У процесі навчання запам'ятовуються координати усіх точок та належність їх до відповідних образів $a_1,....,a_m$.

2. Розпізнавання.

a. Для точки **x**, яка підлягає розпізнаванню, обчислюється потенціали кожного образу, тобто суми

$$\Phi_i(x) = \frac{1}{n_i} \sum_{a \subset \alpha_i} \varphi_a(x), \quad i = 1, 2, ..., m,$$

де m – кількість різних образів, n_i – кількість точок відповідного образу, використаних у процесі навчання, $\varphi_a(x) = \frac{1}{1 + \alpha D^2(a, x)}$ – потенціал, який створює точка **a** у точці **x**.

b. Порівнюються величини $\Phi_1(x), \Phi_2(x), \ldots, \Phi_m(x)$, і точка **x** відноситься до того образу, потенціал якого є найбільшим.

У випадку двох образів a_1 та a_2 рішення можна приймати на основі значення знаку функції

$$\Delta\Phi(x) = \Phi_{a_1}(x) - \Phi_{a2}(x)$$

Застосування вищенаведеного алгоритму для розпізнавання цифр, зображених за допомогою рецепторного поля 6×10 у середньому дало змогу досягнути правильного розпізнавання у 85% випадків за умови, що $n_1 = n_2 = ... = n_{10} = 12$, причому подальше збільшення обсягу навчальної вибірки практично не впливає на якість розпізнавання.

Покращена модифікація алгоритму навчання. Алгоритм розпізнавання може бути покращений, шляхом введення поняття «ваги точки» та доповненням кроку навчання наступною операцією. Після запам'ятовування усіх об'єктів їм присвоюють початкову вагу 1. Далі до елементів навчальної вибірки застосовують крок 2 алгоритму навчання, причому потенціали середні потенціали обчислюють за формулою

$$\Phi_i(x) = \frac{1}{n_i} \sum_{a \subset \alpha_i} \varphi_a(x) w(a), \quad i = 1, 2, ..., m,$$

де w(**a**) — вага точки **a**.

Якщо при розпізнаванні об'єкту **a** відбувається помилка, то вагу w(**a**) відповідного об'єкту збільшують на деяку величину, наприклад на одиницю. Потім застосовується ще один такий самий цикл перевірки та корекції ваг і т.д. Цикли повторюються до тих пір, поки усі відомі фігури не будуть розпізнані правильно.

Застосування покращеного алгоритму до розпізнавання дало змогу підвищити відсоток правильного розпізнавання нових зображень цифр до 89%.

Метод найближчого сусіда.

1. Випадок одного еталону.

В деяких задачах об'єкти деяких класів (образів) мають *тенденцію до групування навколо деякого об'єкту, який є типовим* або репрезентативним для відповідного образу. Типовим прикладом є задача зчитування банківських чеків, для якої вектори, які відповідають образам кожного класу, будуть майже ідентичними.

Розглянемо M класів, які допускають зображення за допомогою еталонних представників $\mathbf{z}_1,\ldots, \mathbf{z}_M$. Евклідова відстань між довільним вектором $\mathbf{x}$ та цими образами обчислюється за формулою

$$D_i = \sqrt{\sum_{j=1}^{n}(x_j - z_{ij})^2}$$

або $D_i = \|x - z_i\| = \sqrt{(x-z_i)'(x-z_i)}$.

Вектор $\mathbf{x}$ відноситься до класу ω_i, якщо умова $D_i < D_j$ виконується для усіх $j \neq i$. Перетворимо формулу обчислення відстані

$$D_i^2 = \|x - z_i\|^2 = (x-z_i)'(x-z_i) = x'x - 2x'z_i + z_i'z_i = x'x - 2\left(x'z_i - \frac{1}{2}z_i'z_i\right).$$

Остання формула показує, що вибір мінімальної відстані до класу еквівалентний максимізації величини

$$x'x - 2\left(x'z_i - \frac{1}{2}z_i'z_i\right).$$

Тому функції рішень можна визначити як

$$d_i = x'z_i - \frac{1}{2}z_i'z_i, \quad i = 1, 2, ..., M.$$

де $d_i(\mathbf{x})$ – лінійні функції рішень. Якщо покласти

$$w_{ij} = z_{ij}, \quad j = 1, 2, \ldots, n,$$
$$w_{i,n+1} = -(1/2)\,\mathbf{z}_i'\mathbf{z}_i$$

і $\mathbf{x} = (x_1, x_n, 1)$, то функції рішень можна записати у виді:

$$d_i(x) = w'x, \quad i = 1, 2, ..., M$$

де $\mathbf{w}_i\,(w_{i1},\ldots, w_{in}, w_{i,n+1})$. На рис. 5.14 зображена відокремлююча межа для випадку двох класів.

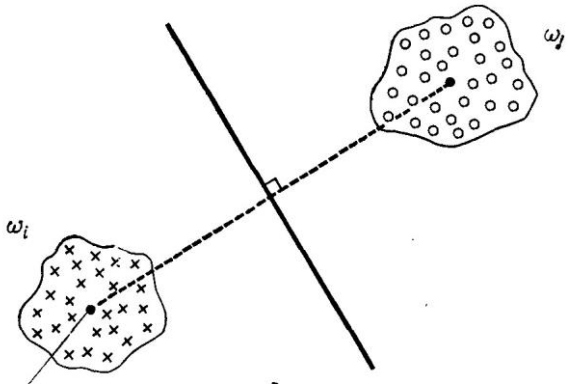


Рисунок 5.14 – Розділення двох класів

Межа між класами є гіперплощиною, яка рівновіддалена від еталонних представників класів. Описаний підхід ще називають кореляцією або співставленням із кластером.

2. Випадок кількох еталонів.

У цьому випадку кожний клас ω_i можна охарактеризувати еталонами $\mathbf{z}^1{}_i,,\ldots,\ \mathbf{z}_i^{Ni}$, N_i – кількість еталонів i-го класу. Одним із найпростіших підходів є метод "найближчого сусіда" (НС-правило). Відстань між i-им класом та об'єктом **x** можна обчислити за формулою:

$$D_i = \min_i \left\| x - z_i \right\|, \quad i = 1, 2, ..., N_i.$$

тобто з використанням відстані до найближчого до **x** еталона класу. Функції рішень можна записати у виді:

$$d_i = \max_i \{x' z_i^l - \frac{1}{2}(z_i^l)' z_i^l\}, \quad l = 1, 2, ..., N_i.$$

і процедура віднесення нового об'єкта до класу така сама, як і раніше. На рис. 5.15 показані межі класів у випадку двох класів, які містять по два еталони кожний.

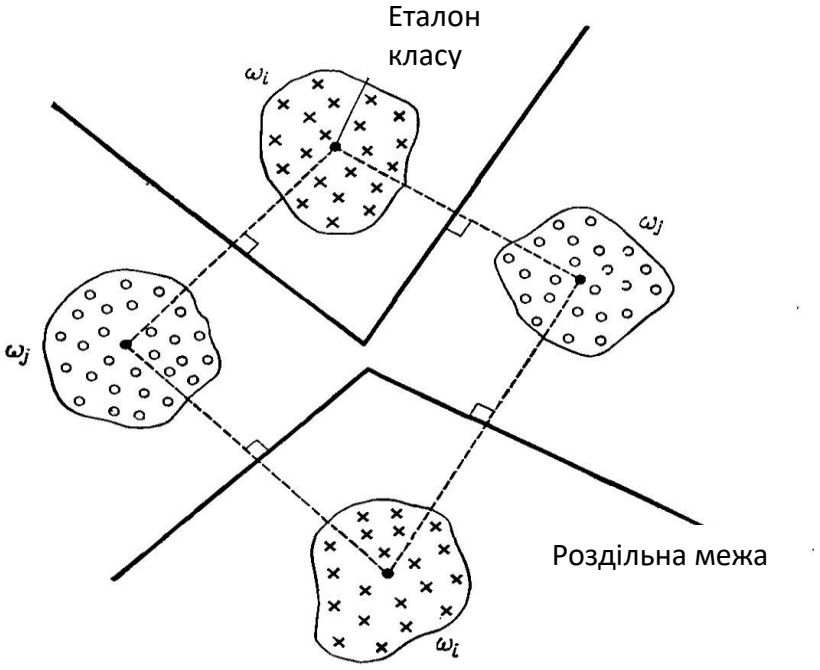


Рисунок 5.14 – Розділення двох класів з двома еталонами

Отриманий класифікатор є частинним випадком *кусково-лінійного класифікатора*.

3. Узагальнення принципів класифікації по мінімуму відстані. Метод *k*-найближчих сусідів.

НС-правило може бути узагальнено на *k-НС правило*, згідно якого обчислюються *k*-найближчих сусідів і об'єкт відноситься до того класу, до якого відноситься найбільша кількість з *k*-найближчих еталонів.

5.7. Класифікація на основі байєсівської теорії рішень

Байєсовський підхід виходить із статистичної природи спостережень. За основу береться припущення про існування ймовірнісного заходу на просторі образів, який або відомий, або може бути оцінений. Мета полягає у розробці такого класифікатора, який правильно визначатиме найбільш ймовірний клас для пробного образу. Тоді завдання полягає у визначенні "найвірогіднішого" класу.

Задано M класів $\Omega_1, \Omega_2, ..., \Omega_M$, і навіть $P(\Omega_i|x)$, i = 1, 2, ..., M – ймовірність те, що невідомий образ, представлений вектором ознак x, належить класу Ω_i .

$P(\Omega i|x)$ називається апостеріорною ймовірністю, оскільки задає розподіл ознаки класу після експерименту (a posteriori – тобто після того, як значення вектора ознак x було отримано).

Розглянемо випадок двох класів Ω_1 та Ω_2. Природно вибрати вирішальне правило в такий спосіб: об'єкт відносимо до того класу, у якого апостеріорна ймовірність вище. Таке правило класифікації максимально апостеріорної ймовірності називається Байєсовським: якщо $P(\Omega_1|x) > P(\Omega_2|x)$, то x класифікується в Ω_1, інакше в Ω_2 . Таким чином, для Байєсовського вирішального правила необхідно отримати апостеріорні ймовірності $P(\Omega_i|x)$, i = 1, 2. Це можна зробити за допомогою формули Байєса.

Формула Байєса, отримана Т. Байєсом в 1763 році, дозволяє обчислити апостеріорні ймовірності подій через апріорні ймовірності та функції правдоподібності.

Нехай $A_1, A_2, ..., A_n$ – а повна група несумісних подій.

$$\bigcup A_i = \Omega \; . \; A_i \cap A_j = \emptyset,$$

при $i \neq j$. Тоді апостеріорна ймовірність має вид:

$$P(A_i \mid B) = \frac{P(A_i)P(B \mid A_i)}{\sum_{i=1}^{n} P(A_i)P(B \mid A_i)},$$

де $P(A_i)$ – апріорна ймовірність події A_i , $P(B|A_i)$ – умовна ймовірність події B за умови, що сталася подія A_i .

Розглянемо отримання апостеріорної ймовірності $P(\Omega|x)$, знаючи $P(\Omega)$ та $P(x|\Omega)$.

Для незалежних подій A і B можна записати:

$$P(AB) = P(A|B)P(B), \qquad P(AB) = P(B|A)P(A),$$

звідси:

$$P(A|B)P(B) = P(B|A)P(A),$$

або

$$P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A)},$$

Якщо P(A) і P(A|B) описуються щільностями p(x) і p(x|B), то

$$P(B \mid x) = \frac{p(x \mid B)P(B)}{p(x)} \Rightarrow P(\Omega_i \mid x) = \frac{p(x \mid B)P(\Omega_i)}{p(x)}$$

При перевірці класифікації порівняння $P(\Omega_1|x)$ та $P(\Omega_2|x)$ еквівалентно порівнянню $p(x|\Omega_1)P(\Omega_1)$ та $p(x|\Omega_2)P(\Omega_2)$. У випадку, коли $P(\Omega_1|x) = P(\Omega_2|x)$ вважається, що міра множини x дорівнює нулю.

Таким чином, завдання порівняння за апостеріорною ймовірністю зводиться до обчислення величин $P(\Omega_1)$, $P(\Omega_2)$, $p(x|\Omega 1)$, $p(x|\Omega_2)$. Вважатимемо, що в нас достатньо даних для визначення ймовірності приналежності об'єкта кожному з класів $P(\Omega_i)$, i = 1, 2. Такі ймовірності називаються апріорними ймовірностями класів. А також вважатимемо, що відомі функції розподілу вектора ознак для кожного класу $P(x|\Omega i)$, i = 1, 2.

Вони називаються функціями правдоподібності x стосовно Ω_i. Якщо апріорні ймовірності та функції правдоподібності невідомі, їх можна оцінити методами Якщо апріорні ймовірності та функції правдоподібності невідомі, їх можна оцінити методами математичної статистики на множині прецедентів. Наприклад,

$$P(\Omega_i) \approx \frac{N_i}{N},$$

де N_i число прецедентів з Ω_i, i = 1,2. N – загальна кількість прецедентів. $P(x|\Omega_i)$ може бути наближено до гістограми розподілу вектора ознак для прецедентів з класу Ω_i.

У теорії статистичних рішень всі види вирішальних правил для $M \geq 2$ класів засновані на формуванні відношення правдоподібності L і його порівняння з певним порогом c, значення якого визначається обраним критерієм якості:

$$L = \frac{f_n(x_1,\ldots,x_n \mid \Omega_1)}{f_n(x_1,\ldots,x_n \mid \Omega_2)} \geq c$$

де $f_n(x_1,\ldots,x_n \mid \Omega_1)$ – умовна n-мірна щільність ймовірності вибіркових значень $x_1\ldots,x_n$ при умові їх належності до класу Ω_i. В статистичному розпізнаванні ці щільності, в принципі, не відомі, і в формулу) підставляються їх оцінки $\hat{f}_n(x_1,\ldots,x_n \mid \Omega_1)$.

Таким чином, в правилі розв'язку з порогом c порівнюється оцінка відношення правдоподібності

Отже, Байєсовський підхід до статистичних завдань ґрунтується на припущенні про існування певного розподілу ймовірностей для кожної ознаки. Недоліком цього є необхідність постулювання як існування апріорного розподілу для невідомого параметра, і знання його форми.

Мінімізація середнього ризику

Розглянемо правила прийняття рішень у задачі статистичної класифікації на прикладі двох класів та однієї ознаки, заданої дійсним значенням вимірювання X. Завдання полягає у визначенні на шкалі X інтервалів Ω_1 та Ω_2, на яких прийматимуться рішення на користь першого та другого класу

відповідно. Для простоти надалі класи та відповідні їм області прийняття рішення позначатимемо одним і тим самим символом Ω.

Припустимо, що отримана вся необхідна статистична інформація про класи: функції щільності статистичного розподілу $f_1(x)$ і $f_2(x)$ (рис. 5.15), а також апріорні ймовірності $P(\Omega_1)$ та $P(\Omega_2)$ появи даних класів.

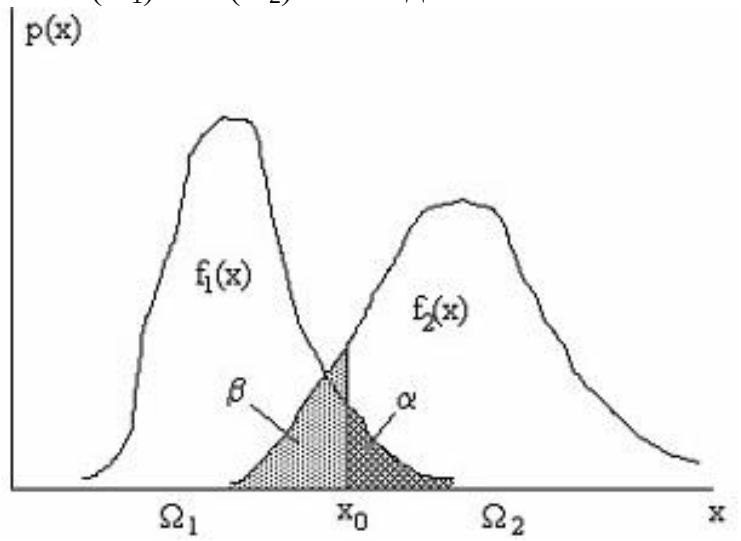


Рисунок 5.15 – Статистична гіпотеза для двох класів

Нехай ми вибрали деяку точку x_0, що розділяє всю шкалу значень ознаки x на два інтервали: $(-\infty, x_0]$ відповідає області рішень на користь класу Ω_1, (x_0, ∞) – області рішень на користь класу Ω_2. несуперечна безліч припущень про властивості випадкової величини ξ називається статистичною гіпотезою, в даному випадку ми розглядаємо дві статистичні гіпотези: ξ, що приймає значення x, має розподіл із щільністю $f_1(x)$, тобто належить класу Ω_1, – це гіпотеза H_1 проти альтернативи H_2: ξ має розподіл із щільністю $f_2(x)$, тобто належить класу Ω_2.

Імовірність появи будь-якого значення x відрізняється від нуля і для першого, і для другого класу на всій множині X, тому при прийнятті рішення щодо належності деякого значення x до одного з класів можуть виникнути 4 ситуації.

1. Приймаємо гіпотезу H1 і вона вірна.
2. Приймаємо H2, але правильна H1.
3. Приймаємо H2 і вона вірна.
4. Приймаємо H1, але правильна H2.

Імовірність виникнення ситуації 1 відповідає площі під $f_1(x)$ на напівінтервалі $(-\infty, x_0]$, ситуації 2 – площі під $f_1(x)$ на інтервалі (x_0, ∞), ситуації 3 – площі під $f_2(x)$ на (x_0, ∞), ситуації 4 – площі під $f_2(x)$ на $(-\infty, x_0)$ Сумарна площа під $f_1(x)$ та $f_2(x)$ для ситуацій 2 і 4 – це повна ймовірність помилок у нашій схемі прийняття рішень. У випадку двох альтернативних гіпотез помилку, що відповідає ситуації 2, зазвичай називають **помилкою першого роду** (α), помилку, що відповідає ситуації 4, – **помилкою другого роду** (β).Взагалі кажучи, поняття помилок першого та другого роду симетрично і залежить від

того, яка гіпотеза є основною, а яка – альтернативною. Якби H_2 була основною гіпотезою, помилка першого роду відповідала б ситуації 4.

Байєсівська стратегія мінімального середнього ризику.

Щоб побудувати класифікатор на основі описаної схеми, ми повинні виробити правило оптимального розбиття всієї множини X на ділянці прийняття рішення на користь кожного класу. У нашому конкретному випадку – правило вибору точки x_0, що розділяє. Оскільки відмінність між класами пов'язана з частотою появи тих чи інших значень x цих класів, природно вибрати точку x_0 в такий спосіб, щоб за багаторазовому пред'явленні класифікатору образів x усереднена за сукупністю рішень помилка була мінімальною.

Для цього запровадимо "вартість" кожного прийнятого рішення. Для описаних вище чотирьох ситуацій із класами Ω_1 та Ω_2 запишемо відповідні вартісні коефіцієнти у виді матриці:

$$c = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

Діагональні елементи такої вартісної матриці відповідають випадкам правильної класифікації (ситуації 1 та 3), два інших – помилкам (ситуації 2 та 4). Вартісні коефіцієнти можна поставити так, щоб вони відображали наші переваги щодо тієї чи іншої ситуації. Найчастіше обмежуються умовою
$c_{11} = c_{22} = 0$ і $c_{12} = c_{21} = 1$, тобто розглядають лише втрати від помилок.

Розглядаючи вартісні коефіцієнти як наш "ризик" у кожній із можливих ситуацій (які можна розглядати як випадкові події), введемо поняття середнього ризику для описаних чотирьох випадків:

$$R = \sum_{k=1}^{K} P(\Omega_k) \sum_{l=1}^{K} c_{kl} P(c_{kl})$$

Тут c_{kl} – це платіжні коефіцієнти з (8.1), а $P(c_{kl})$ – ймовірність кожної з чотирьох подій (ймовірність відповідної плати подію). У разі коли $c_{11}=c_{22}=0$ і $c_{12}=c_{21}=1$ (тобто «плата» – це штраф за помилки), функцію R називають також функцією втрат.

Розглянемо, що є ймовірності $P(c_{kl})$. Наприклад, ймовірність плати c_{11} є ймовірність одночасного здійснення двох подій: ξ належить до $f_1(x)$ (ймовірність цієї події у Ω_1 є $\int_{-\infty}^{x_0} f_1(x)dx +$ та появи самого класу Ω_1 (апріорна ймовірність цієї події – $P(\Omega_1)$.

Отже, формулу ризику можна записати так:

$$R = c_{11}P(\Omega_1)\int_{-\infty}^{x_0} f_1(x)dx + c_{12}P(\Omega_1)\int_{x_0}^{\infty} f_1(x)dx + c_{22}P(\Omega_2)\int_{x_0}^{\infty} f_2(x)dx + c_{21}P(\Omega_2)\int_{-\infty}^{x_0} f_1(x)dx$$

Мінімум R у точці x_0 досягається за умови $\left.\frac{dR}{dx}\right|_{x=x_0} = 0$.

Візьмемо похідну в точці x_0, враховуючи, що $\int f(x)dx=F(x)$, $F(-\infty)=0$, $F(\infty)=1$:

$$\frac{dR}{dx}\Big|_{x=x_0} = c_{11}P(\Omega_1)f_1(x) - c_{12}P(\Omega_1)f_1(x) - c_{22}P(\Omega_2)f_2(x) + c_{21}P(\Omega_2)f_2(x)$$

Звідси маємо наступне співвідношення для $x = x_0$:

$$\frac{P(\Omega_1)f_1(x)}{P(\Omega_2)f_2(x)} = \frac{c_{21} - c_{22}}{c_{12} - c_{11}} = \lambda$$

Цей вираз називається відношенням правдоподібності, а величина λ – коефіцієнтом правдоподібності. При значеннях $\frac{P(\Omega_1)f_1(x)}{P(\Omega_2)f_2(x)} \geq \lambda$ рішення приймається на користь Ω_1, при $\frac{P(\Omega_1)f_1(x)}{P(\Omega_2)f_2(x)} < \lambda$ – на користь Ω_2.

Якщо покласти, що $c_{11} = c_{22} = 0$ і $c_{12} = c_{21} = 1$, отримаємо:

$$\frac{P(\Omega_1)f_1(x)}{P(\Omega_2)f_2(x)} = 1$$

або, у логарифмічній формі,

$$\ln\frac{P(\Omega_1)f_1(x)}{P(\Omega_2)f_2(x)} = 0$$

Якщо значення ознаки для обох класів розподілено за нормальним законом

$$f(x) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

із середніми m_1, m_2 та середньоквадратичними відхиленнями σ_1, σ_2 відповідно, відношення правдоподібності у логарифмічній формі має вид:

$$\ln\frac{P(\Omega_1)}{P(\Omega_1)} + \ln\frac{\sigma_2}{\sigma_1} - \frac{(x-\mu_1)^2}{2\sigma_1^2} + \frac{(x-\mu_2)^2}{2\sigma_2^2} = 0$$

Тобто x_0 є розв`язком квадратного рівняння. Випадок, коли рівняння має два дійсні корені, представлений на рис. 5.16.

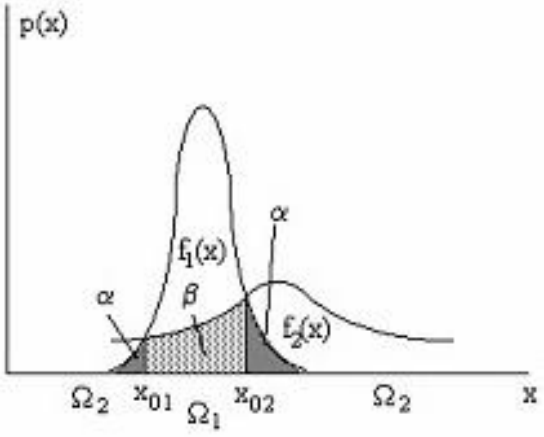


Рисунок 5.16 – Поділ класів із різними дисперсіями ознаки.

5.8. Ансамблеві методи

Типи ансамблевих методів у машинному навчанні

Методи ансамблю допомагають створити декілька моделей, а потім об'єднати їх для отримання поліпшених результатів. Деякі ансамблеві методи класифікуються на такі групи:

1. Послідовні методи

У такому методі ансамблю існують послідовно генеровані базові учні, в яких існує залежність даних. Усі інші дані базового учня мають певну залежність від попередніх даних. Отже, попередні помилкові дані налаштовуються виходячи з його ваги для покращення продуктивності загальної системи.

Приклад : Підвищення, (Бустінг, англ. Boosting)

2. Паралельний метод

У такому методі ансамблю базовий учень генерується в паралельному порядку, в якому залежності від даних немає. Усі дані базового учня формуються незалежно.

Приклад : укладання (стекінг, stacking)

3. Однорідний (гомогенний) ансамбль

Такий метод ансамблю - це поєднання однотипних класифікаторів. Але набір даних різний для кожного класифікатора. Це змусить більш точно працювати комбіновану модель після узагальнення результатів від кожної моделі. Цей тип ансамблевого методу працює з великою кількістю наборів даних. У гомогенному методі метод вибору особливостей однаковий для різних даних тренувань.

Приклад: Народні методи, такі як розфасування та підсилення, входять в однорідний ансамбль.

4. Гетерогенний ансамбль

Такий метод ансамблю - це поєднання різних типів класифікаторів або моделей машинного навчання, в яких кожен класифікатор будується на одних і тих же даних. Такий метод працює для невеликих наборах даних. У неоднорідному методі вибір особливостей для одних і тих же навчальних даних різний. Загальний результат цього ансамблевого методу здійснюється шляхом усереднення всіх результатів кожної комбінованої моделі.

Чотири перевірених способи робити ансамблів.

Рисунок 5.17 – Основні типи ансамблевих методів

Bagging (пакування в мішок чи сумку)

Bagging, він же Bootstrap AGGregatING. Ідея – навчаємо один алгоритм багато разів на випадкових вибірках з вхідних даних (навчальної вибірки). В кінці усереднюємо відповіді.

Дані в випадкових вибірках можуть повторюватися. Тобто з набору 1-2-3 ми можемо робити вибірки 2-2-3, 1-2-2, 3-1-2 і так поки не набридне. На них ми навчаємо один і той же алгоритм кілька разів, а в кінці знаходимо відповідь простим голосуванням.

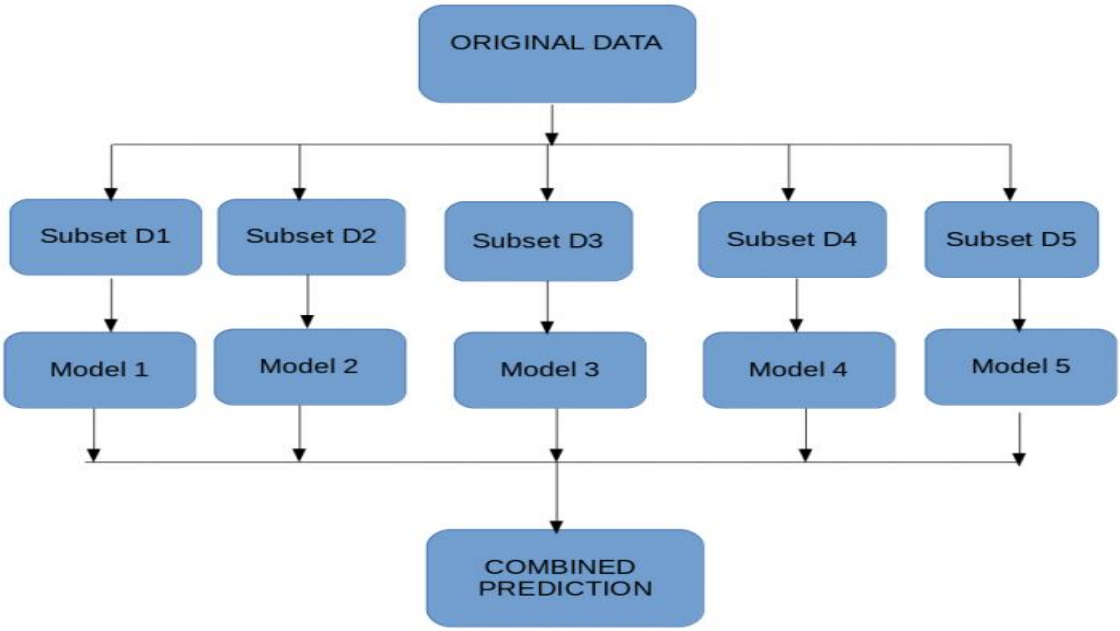


Рисунок 5.18 – Структура алгоритмів Bagging

Random Forest

Найпопулярніший приклад бегінга – алгоритм Random Forest (рис. 5.19).

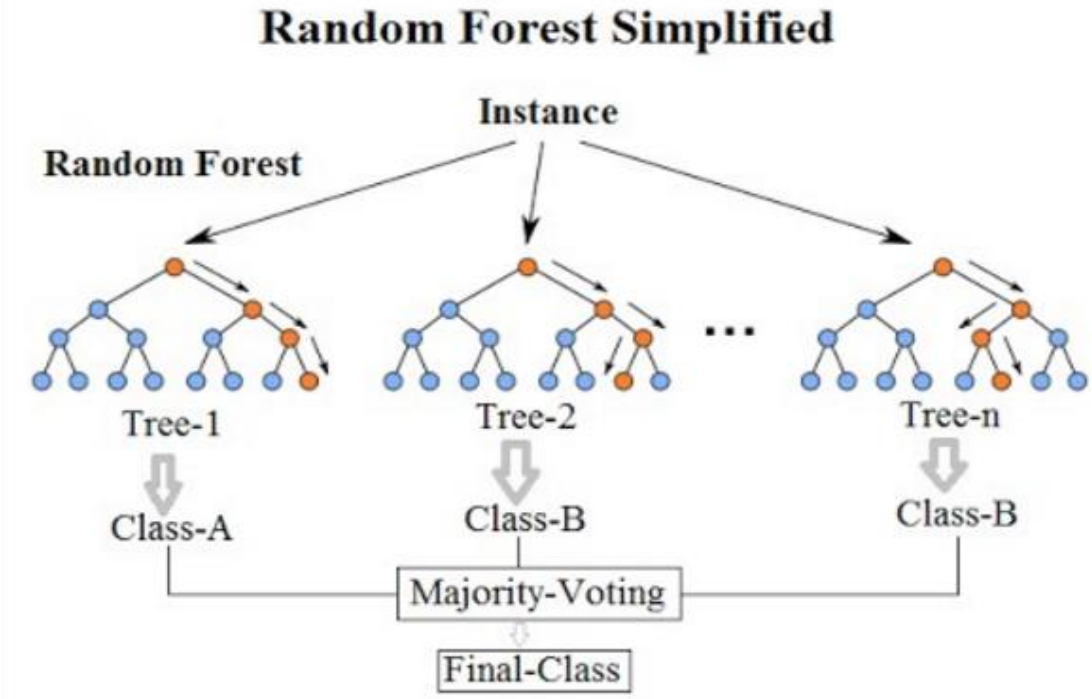


Рисунок 5.19 – Структура алгоритмів Random Forest

Цей метод ансамблю поєднує дві моделі машинного навчання, тобто завантаження та агрегацію, в єдину модель ансамблю. Мета методу – зменшити велику дисперсію моделі. Нехай є великий набір даних (скажімо, 1000 зразків) підвибіркою (скажімо, 10 підвібірок, кожна містить 100 зразків даних). Кілька дерев рішень будуються на кожній підвибірки. Для ефективності моделі кожне окреме дерево рішень нарощується в глибину. Результати кожного дерева рішень узагальнюються, щоб отримати остаточний прогноз. Дисперсія узагальнених даних зменшується. Точність прогнозування моделі в методі упаковки залежить від кількості використовуваного дерева рішень. Різні підвідбірки даних створюються випадковим чином. Вихід кожного дерева має високу кореляцію.

Перевага бегінгу, наприклад, перед нейромережами – можливість працювати паралельно.

Boosting (підвищення)

Boosting – ітераційний алгоритм, що реалізує «сильний» класифікатор, який дозволяє досягти довільно малої помилки навчання (на навчальній вибірці) на основі композиції "слабких" класифікаторів, кожен з яких кращий, ніж просто вгадування, тобто ймовірність правильної класифікації більше 0,5.

Ключова ідея: використання вагових версій одних і тих же навчальних даних замість випадкового вибору їх підмножини.

У Boosting рівна вага (рівномірний розподіл ймовірностей) надається вибірковим навчальним даним (скажімо, D1) на самому початку раунду. Потім ці дані (D1) передаються базовому учню (скажімо, L1). Неправильно класифікованим екземплярам L1 присвоюється вага, що вища, ніж правильно класифікованим екземплярам, але з урахуванням того, що загальний розподіл ймовірностей буде дорівнювати 1. Ці покращені дані (скажімо, D2) потім передаються другому базовому учню (скажімо, L2) і так далі. Потім результати об’єднуються у формі голосування.

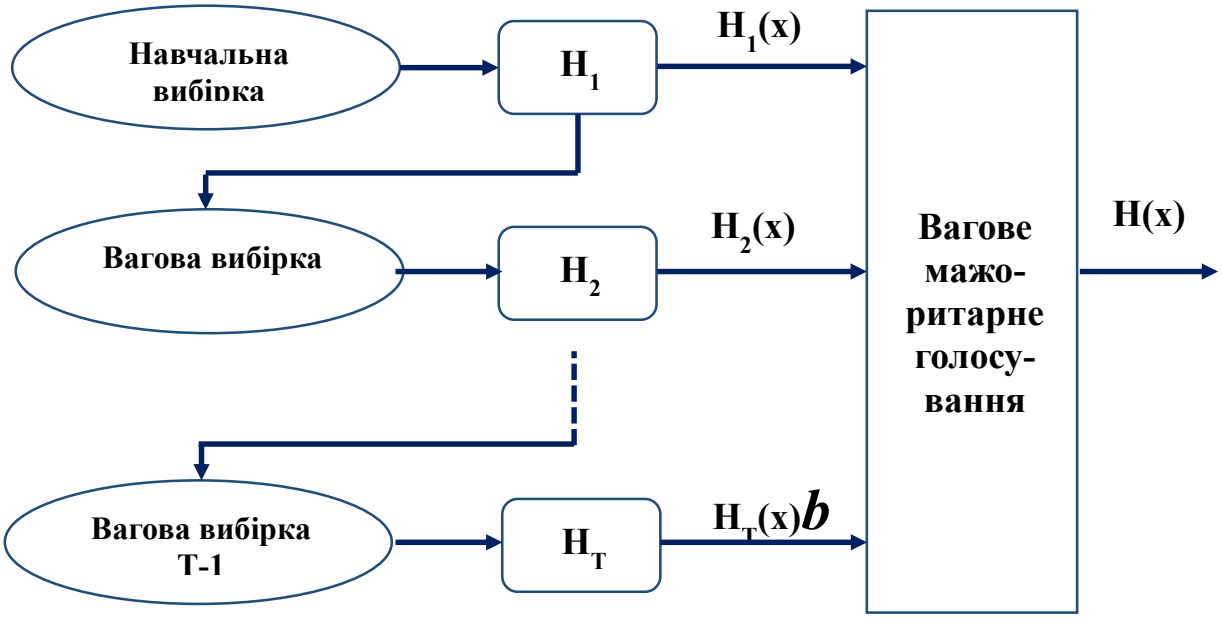


Рисунок 5.20 – Boosting (загальна схема)

Слабкі класифікатори утворюються послідовно, розрізняючись тільки вагами навчальних даних, які залежать від точності попередніх класифікаторів.

Базові класифікатори повинні бути слабкими, з сильних хорошу композицію не збудувати ("бритва Оккама ").

Причини цього: сильний класифікатор, даючи нульову помилку на навчальних даних, адаптується і композиція буде складатися з одного класифікатора один, навіть сильний, класифікатор може дати "поганий" прогноз на даних тестування, даючи " хороші" результати на навчальних даних.

Stacking (укладання)

Ідея – навчаємо кілька різних алгоритмів і передаємо їх результати на вхід останньому, який приймає остаточне рішення (рис. 5.21).

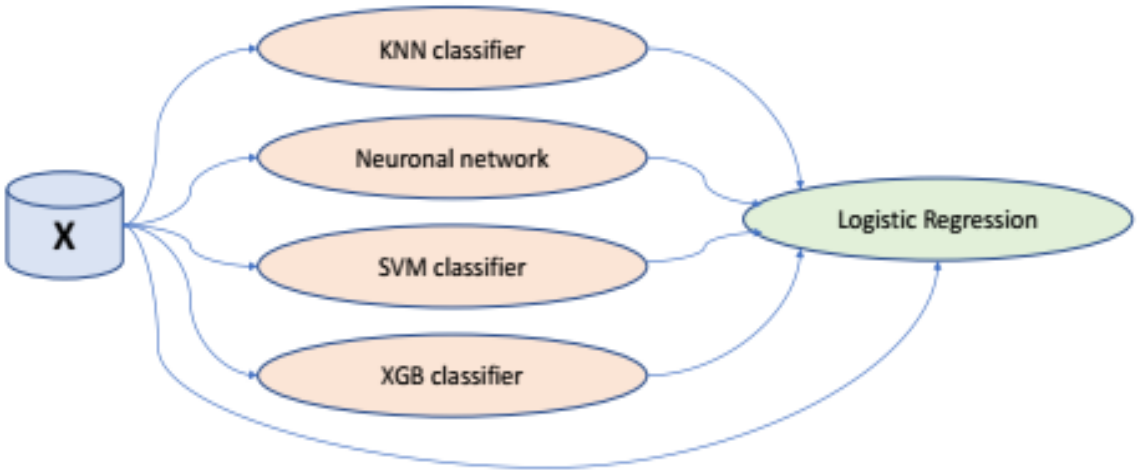


Рисунок 5.21 – Stacking (загальна схема)

Перед початком стекованого навчання нам потрібно вибрати кількість M алгоритмів, які ми хочемо об'єднати: L_1,. … ., L_M. I ми позначаємо через L алгоритм навчання для нашої метамоделі.

- Нехай D = f(X_1,Y_1),. … .,(X_n,Y_n)g позначають наш набір даних. На першому етапі, щоб створити так звані моделі рівня 0 або узагальнювачі, ми застосовуємо алгоритми навчання рівня 0 до вихідного набору даних, випадковим чином розділеного за допомогою перехресної перевірки для навчання та тестування майбутніх моделей. Це h_m=L_m(D) для m = 1,. . ., M.
- Для кожного X_i тестових зразків рівня 0 ми використовуємо кожен узагальнювач для прогнозування їх змінної відповіді. Тоді ми позначимо вихід як z_{mi}=h_m(Xi).
- Ці виходи разом із їхніми справжніми значеннями Y_i утворюють вектор передбачення з M + 1 координатами (Y_i, z_{1i} ,. ... .,$z_{M,i}$). Для кожного екземпляра тестової вибірки ці вектори збираються в новий набір даних D_0, дані рівня 1.
- Останній дозволяє навчати та тестувати L, алгоритм навчання рівня 1. Ми застосовуємо його до нового набору даних так само, як і раніше. У позначеннях h=L(D_0).
- Прогнозувати за новими даними, отримані узагальнювачі h_1,. …, h_M. Остаточним прогнозом для нових даних x є: f(x) = h(h_1(x), ..., h_M(x))

6. ШТУЧНІ НЕЙРОННІ МЕРЕЖІ

6.1. Біологічний нейрон

Відомо, що розумова діяльність людини породжується клітинами мозку людини під назвою нейрони. Їх дуже багато (по різним оцінкам – 10^{10}-10^{12}). Між собою вони зв'язані тисячами зв'язків. Нейрон є особливою біологічною клітиною, що обробляє інформацію. Від нейрона до нейрона інформація передається у виді електричних імпульсів, що нагадує 0 та 1 в комп'ютері. Таким чином, маючи комп'ютерну модель роботи нейронів можна їх з'єднати між собою в моделі комп'ютерної нейронної мережі і змоделювати роботу мозку при розв'язанні інтелектуальних завдань. Розглянемо, як працює біологічний нейрон.

Нейрон (рис. 6.1) складається з дендритів (dendrites), по яким поступають сигнали (імпульси) від інших нейронів, тіла клітини соми (soma) і одного відростка – аксона (axon) з деревоподібними відгалуженнями, вздовж якого передається згенерований тілом клітки потенціал дії або імпульс. Тіло клітини вміщує ядро (nucleus), що містить інформацію про властивості нейрона, і плазму, яка продукує необхідні для нейрона матеріали. На закінченнях волокон знаходяться синапси (synapses).

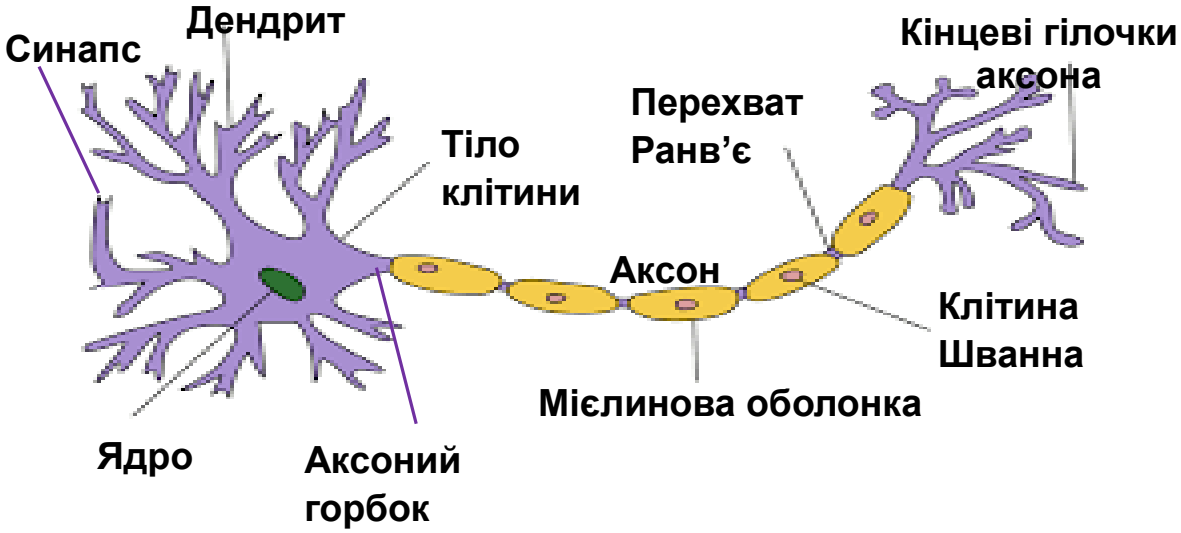


Рисунок 6.1 – Структура біологічного нейрона

Потенціал дії (імпульс), що генерує нейрон представлений на рис. 6.2.

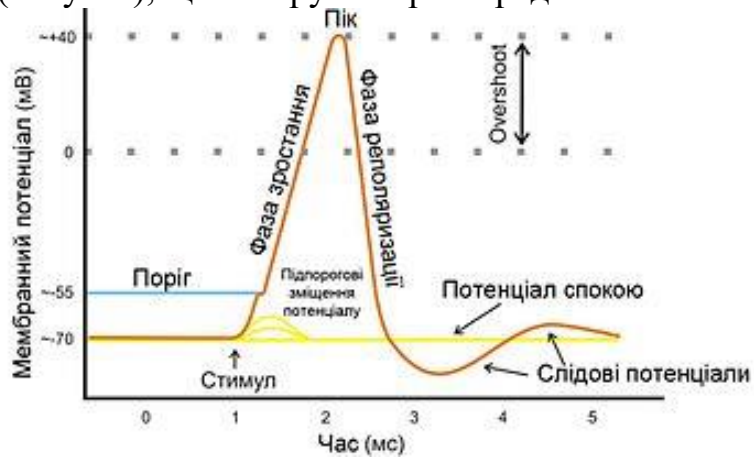


Рисунок 6.2 – Схематичний потенціал дії

Синапс є функціональним вузлом між аксоном одного нейрона і дендритом іншого). Коли імпульс досягає синаптичного закінчення, там продукуються хімічні речовини – нейромедіатори (рис. 6.3). Нейромедіатори дифундують через синаптичну щілину, збуджуючи або пригнічуючи, залежно від типу синапса, здатність нейрона-приймача ґенерувати електричні імпульси.

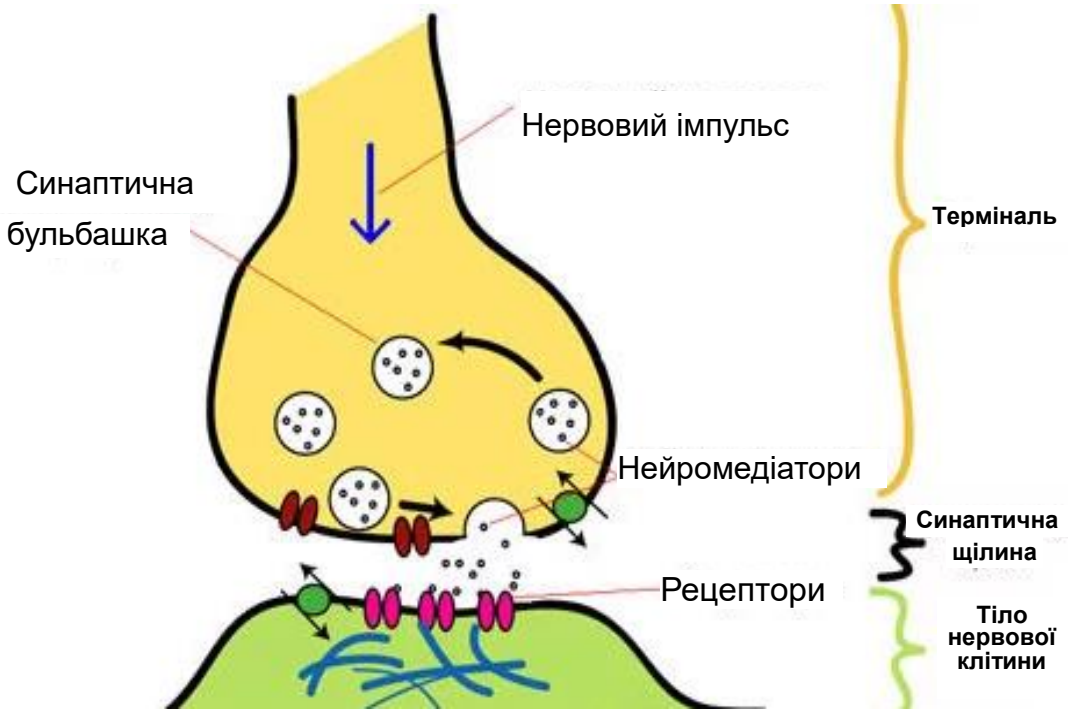


Рисунок 6.3 – Передача нервового імпульсу між нейронами

Результативність синапса може настроюватися сигналами, що проходять через нього, так що синапси можуть навчатися залежно від активності процесів, у яких вони беруть участь. Ця залежність від передісторії діє як пам’ять. Важливо зазначити, що вага синапсів може змінюватися згодом, що змінює й поведінку відповідного нейрона.

У сомі виконується сумування усіх сигналів із врахуванням їх посилення чи послаблення. Сумарний вхідний сигнал співставляється із так званими пороговим значенням активації нейрона: якщо відбувається перевищення порогу то у аксонному горбку генерується сигнал певної величини – нейрон переходить в активний стан; якщо перевищення порогу активації немає, то нейрон залишається в пасивному стані – сигнал не генерується.

6.2 Моделі штучного нейрона

Моделей нейронів існує доволі багато. Вони моделюють властивості нейрона з різною подібністю до природного. Як правило, ці моделі роблять це дуже наближено.

Нейрон Мак-Каллока–Піттса

У 1943 році Воррен Мак-Каллок, нейробіолог, і Уолтер Піттс, логік, опублікували статтю «Логічний підрахунок ідей, іманентних нервовій діяльності» в Bulletin of Mathematical Biophysics (1943), де запропонували математичну модель штучного нейрона, яка імітує найважливіші риси

біологічного нейрону. Штучний нейрон є базовим модулем нейронних мереж. Він моделює основні функції біологічного нейрона.

$$y_j = f\left(\sum_{i=o}^{n} W_{ji} x_i + \theta_j\right)$$

де:

y_j – сигнал на виході j-го нейрона;

f – активаційна (передатна) функція;

n – кількість входів;

W_{ji} – синаптичні ваги;

x_i – вхідні сигнали (i=1, n);

θ_j – зсув (*Bias*)

Часто цю модель зображують наступним чином (рис. 6.4.)

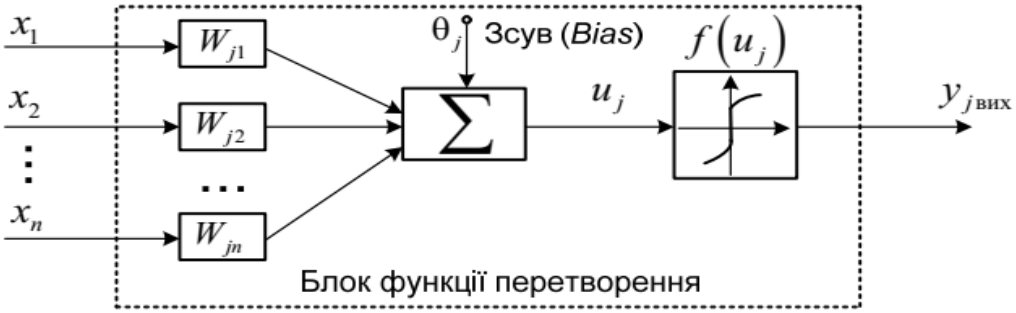


Рисунок 6.4 – Структура нейрона Мак-Каллока–Піттса

Якщо прийняти $\theta_j = W_{j0}\, x_0$ (зазвичай x_0=1). Можна записати:

$$y_j = f\left(\sum_{i=o}^{n} W_{ji} x_i\right)$$

Нейрон Фукушими

Моделює входи, що збуджують ($a_{ji}x_i$) і гальмують ($b_{ji}v_i$) нейрон.

$$y_j = \left(\frac{1 + \sum_{i=1}^{n} a_{ji} x_i}{1 + \sum_{i=1}^{p} b_{ji} v_i}\right)$$

ADALINE (ADAptive LINer Element)

Інша назва **ADAptive LInear NEuron** (Адаптивний ЛІНійний Нейрон**).**

Дозволяє “навчити” нейрон шляхом обчислення коефіцієнтів $\boldsymbol{w_{ji}}$, при яких нейрон правильно реагує на вхідні дані. Математично це формулюється так: знайти такі коефіцієнти $\boldsymbol{w_{ji}}$, при яких досягається задана помилка $\boldsymbol{e_j}$ між бажаним (правильним) значенням $\boldsymbol{d_j}$ і обчисленим на виході нейрона $\boldsymbol{u_j}$.

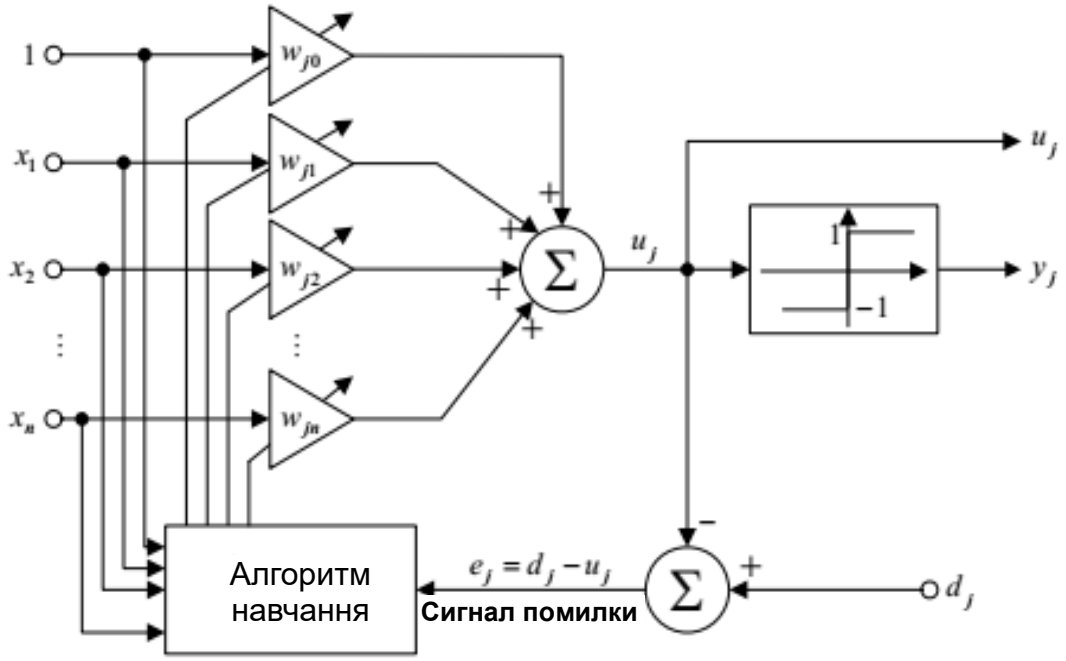


Рисунок 6.5 – Структура нейрона ADALINE

Алгоритм підбору ваг:

$$y_i(u_i) = \begin{cases} 1, u_i > 0 \\ -1, u_i \le 0 \end{cases}$$

Квадратична помилка:

$$E(u_i) = \frac{1}{2} e_i^2 = \frac{1}{2} \left(d_i - \sum_{j=1}^{n} w_{ij} x_j \right)^2$$

Адаптивний підбір вагових коефіцієнтів здійснюється у процесі мінімізації квадратичної помилки.

Незважаючи на нелінійний характер моделі, в цільовій функції присутні лише лінійні елементи, що є сумою зважених вхідних сигналів. У зв'язку з виконанням умови безперервності цільової функції, стало можливим застосування алгоритму градієнтного навчання. Значення вагових коефіцієнтів можуть утонюватися або дискретним способом: $w_{ij}(t+1)=w_{ij}(t)+ne_ix_j$, або безперервним:

$$\frac{dw_{ij}}{dt} = \mu e_i x_j$$

У практичних додатках нейрони типу Adaline, завжди використовують групи, утворюючи шари, звані **Madaline** (Many Adalines). Кожен нейрон, що входить у шар, навчається за правилом Adaline.

N-Adaline (Non linear Adaline) є різновидом Adaline, на вхід якої крім звичайних вхідних сигналів x_j подаються їх нелінійні комбінації (у вигляді, наприклад, x_1^2, x_2^2, $x_1 x_2$, $x_1^2 x_2^2$, тощо), які множаться на відповідні вагові коефіцієнти та підсумовуються. Цими нейронами реалізується розроблений О.

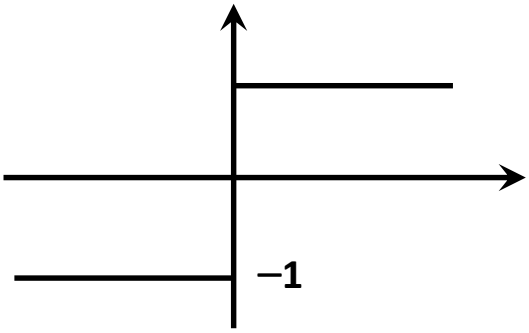
Г. Івахненком (Україна**)** ***метод групового урахування аргументів (МГУА)*** з виходом, наприклад, у виді

$$u = NET = w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2$$

6.3. Функції активації штучного нейрона

Функція активації моделює нелінійну реакцію нейрона на сумарний вхід. З іншого боку, якщо це – функція, то в алгоритмах можна застосовувати потужний апарат функціонального аналізу. Розглянемо найбільш відомі функції.

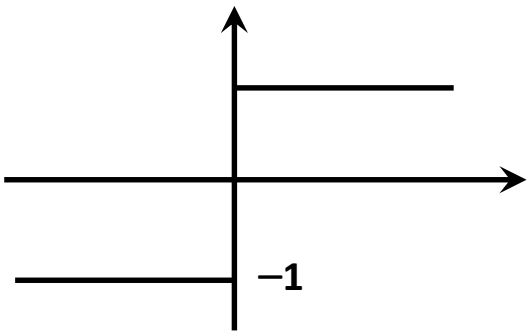
Порогова



$$F(s) = \begin{cases} 1 & якщо \quad s > 0 \\ -1 & якщо \quad s \le 0 \end{cases}$$

Застосовувалась однією з найперших. Недолік – в точці 0 похідна дорівнює нескінченності, що не можливо реалізувати чисельними методами, отже запрограмувати.

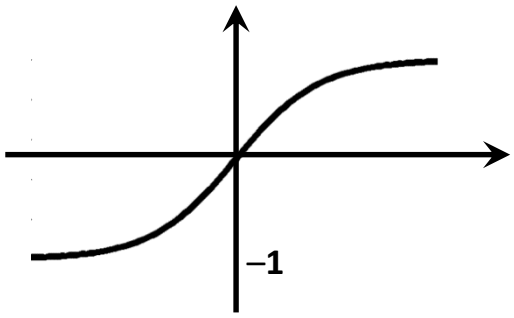
Сигмоїда



$$F(s) = \frac{1}{1 + e^{-as}}$$

Найбільш вживана в алгоритмах навчання нейронних мереж. У всіх точках має похідну. Друга перевага – похідна і інтеграл функції $\boldsymbol{e^x}$ дорівнює цій же функції, що спрощує кінцеві аналітичні вирази. Ступінь нахилу функції визначається коефіцієнтом $\boldsymbol{a}$.

Гіперболічний тангенс

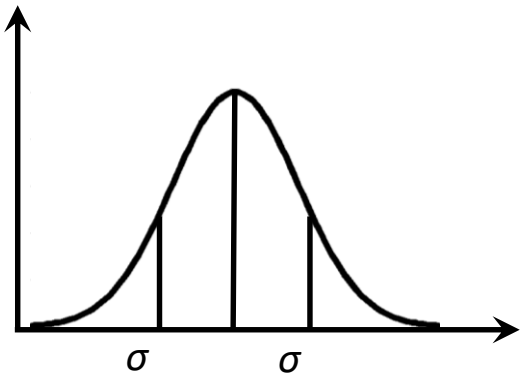


$$F(s)=\frac{e^{as}-e^{-as}}{e^{as}+e^{-as}}$$

До переваг сигмоїди додається симетричність відносно вісі **s**.

В нейронних мережах розглянуті функції активації найчастіше застосовуються до задач класифікації (розпізнавання образів).

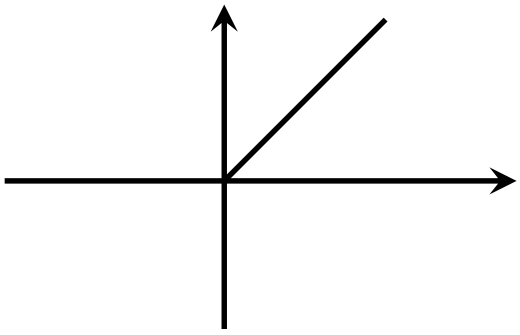
Функція Гауса



$$F(s)=e^{\frac{-(s-a)^2}{2\sigma^2}}$$

Застосовується в задачах регресії.

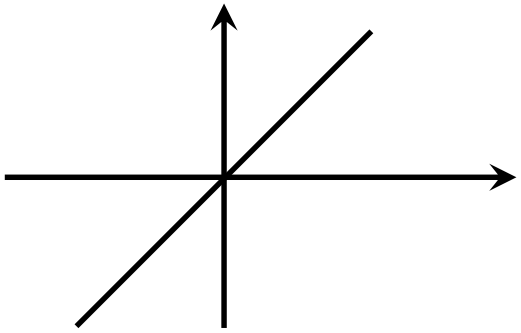
ReLU (rectified linear unit)



$$F(s)=\begin{cases}1 & якщо \quad s<0\\ s & якщо \quad s\geq 0\end{cases}$$

Застосовується в мережах Deep Learning (глибокого навчання).

Лінійна



$F(s) = s$

Застосовується в задачах апроксимації, прогнозування і на виході деяких типів штучних нейронних мереж.

6.4. Класифікація штучних нейронних мереж

Штучна нейронна мережа (ШНМ) – сукупність штучних нейронів, зв'язаних між собою. Якщо брати за аналогію кору головного мозку, то там у першому наближенні ми маємо шість шарів нейронів. Кожен нейрон у шарі має зв'язки з нейронами інших шарів і можна сказати, що під час роботи мозку інформація передається з одного шару на інший, обробляється на кожному шарі, поки не буде досягнутий результат. Можна сказати, що так приблизно діє зоровий канал: нейрони в оці збуджуються, сигнали від них поступають в кору головного мозку, і там, крок за кроком розпізнаються ознаки образу, їх ансамблі і сам образ. За таким принципом будуються ШНМ прямого поширення рис. 6.6. Якщо всі нейрони попереднього шару зв'язані з усіма нейронами наступного, то такі мережі називають повнозв'язними мережами прямого поширення

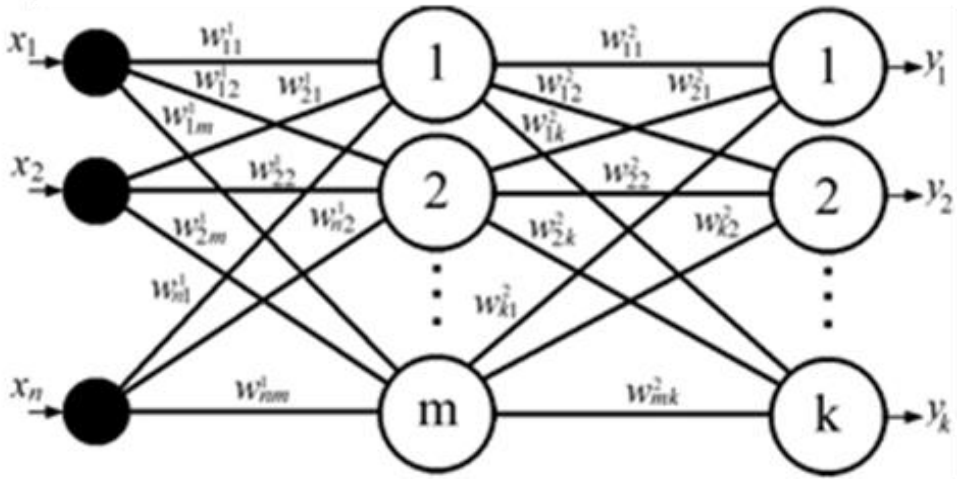


Рисунок 6.6 – Топологія ШНМ прямого поширення

Існує багато різновидів структур (топології) таких мереж. Є мережі, в яких присутні зворотні зв'язки між шарами і в середині шару, що краще моделює зв'язки біологічних нейронів в мозку.

Найбільш відомі мережі: одношаровий персептрон (Peceptron), багатошаровий персептрон або (Feed Forward Neural Network – FFNN), радіально базисні мережі (Radial Basis Network), аутоенкодери (Auto Encoder), глибокі згорткові нейромережі (Deep Convolutional Network), мережа і самоорганізаційні карти Кохонена (Kohonen Network, Self-organizing Map – SOM). Серед мереж із

зворотними зв’язками відзначимо Recurrent Neural Network – зворотні зв’язки в середині шару, рекурентні мережі з пам’яттю (Long/Short Term Memory – LSTM).

Невелику групу утворюють мережі, де всі нейрони зв’язані між собою, в тому числі мають зворотні зв’язки і з собою і називаються повнозв’язними (рис. 6.7).

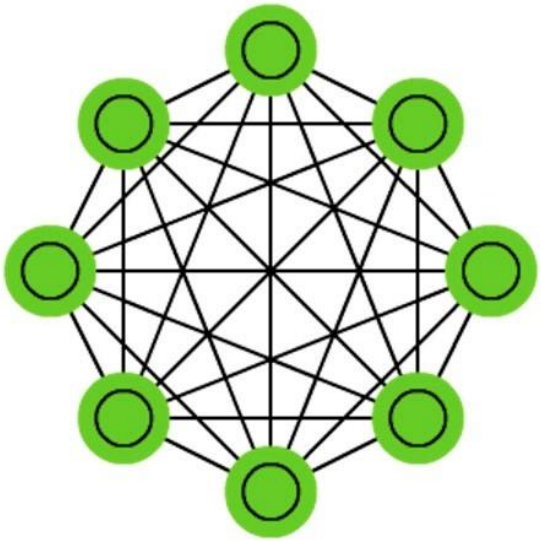


Рисунок 6.7 – Топологія повнозв’язної ШНМ

Найбільш відомі: мережа Хопфілда (Hopfield Network), машина Больцмана (Boltzmann Machin), мережі Маркова (Markov Chain – MC).

Загальна класифікація ШНМ навена на рис.6.8.

На рис. 6.8 наведена сучасна класифікація ШНМ.

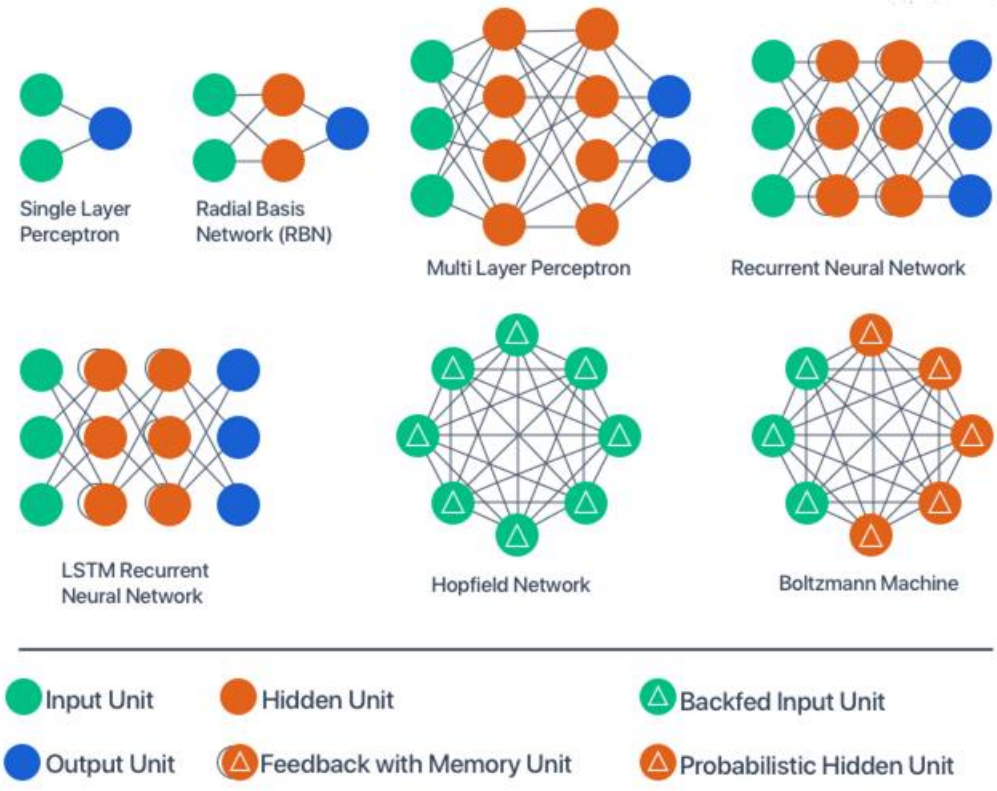


Рисунок 6.8 – Топології штучних нейронних мереж

6.5 Класифікація задач, що розв'язують ШНМ

З точки зору МН штучні нейронні мережі розв'язують два типи задач: задачу класифікації і задачу регресії. В свою чергу їх можна розбити на підзадачі (табл. 6.1), (рис. 6.9)

Таблиця 6.1 – Класифікація задач, що розв'язують ШНМ

Математична постановка	Застосування
Задача класифікації	• Розпізнавання образів • Кластерний аналіз • Прийняття рішень • Керування • Кодування і декодування
Задача регресії	• Прогнозування • Апроксимація функцій

Рисунок 6.9 – Застосування ШНМ

6.6 Навчання штучних нейронних мереж

Терміни і поняття навчання ШНМ

Універсум **(**Генеральна сукупність**)** – вся множина однорідних за певними ознаками даних про об'єкт чи процес, які є предметом дослідження.

Для універсуму при навчанні ШНМ вводяться поняття:

Вектор – набір числових ознак даних на вході або на виході, що характеризують об'єкт чи процес дослідження.

Навчаюча пара $\{x^{\alpha}, y^{\alpha}\}$ – екземпляр вектору на вході і екземпляр бажаного вектору на виході ШНМ.

Навчальна вибірка (*Train*) – множина екземплярів векторів (образів), на яких відбувається навчання ШНМ.

Перевірочна вибірка (*Validation*) – множина вхідних векторів для контролю появи ефекту перенавчання на векторах навчальної вибірки.

Тестова вибірка (*Test*) – множина вхідних векторів для перевірки якості навченої ШНМ.

Функція втрат (*loss function*) показує якість навчання нейронної мережі або ступінь відповідності множини виходів ШНМ бажаній множині виходів. Це метод оцінки того, наскільки добре наш алгоритм моделює дані. Функції втрат є основним джерелом оцінювання в сучасному машинному навчанні.

Епоха – крок навчання ШНМ, за який на вхід послідовно представляються усі вектори навчальної вибірки.

Умови припинення навчання:

- Досягнута наперед задана кількість епох.
- Досягнуте задане значення функції втрат.
- На навчальній вибірці функція втрат зменшується, а на перевірочній почала зростати.
- Досягнутий мінімальний рівень градієнту для функції втрат.

Суть алгоритмів навчання з вчителем:

- У загальному виді ШНМ вирішують задачу апроксимації багатовимірних функцій, тобто побудови багатовимірного відображення F: X ⇒ Y, що узагальнює заданий набір прикладів $\{x^{\alpha}, y^{\alpha}\}$.
- На кожному кроці навчання на вхід мережі подається один вектор x_i або вся множина X;
- Вектор Y, що обчислюється на виході ШНМ порівнюється з цільовим (бажаним) вектором $Y_ц$, що задається учителем;
- Якщо $Y \neq Y_ц$ або не досягнуті умови припинення навчання то ваги нейронів ШНМ змінюються згідно з із вибраним алгоритмом навчання і процес навчання повторюється.

Життєвий цикл ШНМ:

- Створення (вибір з типів мереж для прикладних задач) (Create *Phase*);
- Навчання (*Learning Phase*);
- Застосування (*Recall-Phase*).

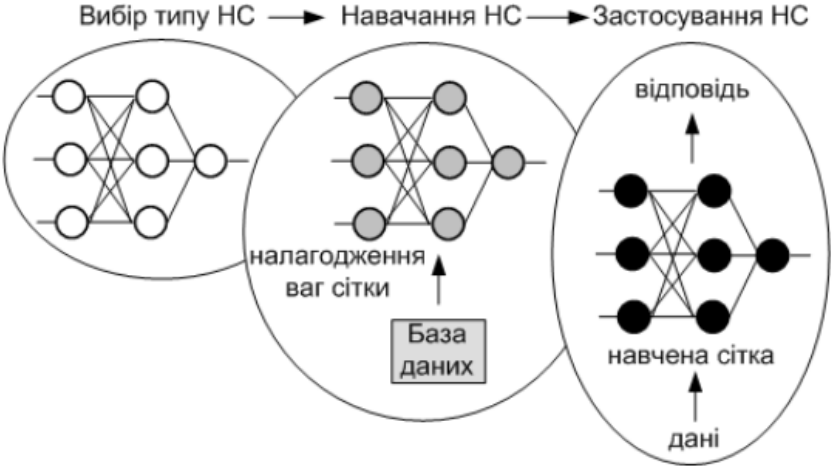


Рисунок 6.9 – Життєвий цикл ШНМ

Навчити нейромережу означає, що для деякої множини входів знайти бажану (або, принаймні, відповідну до неї) множину виходів.

Навчання здійснюється шляхом послідовного пред'явлення вхідних векторів з одночасним корегуванням ваг ШНМ у відповідності з алгоритмом навчання.

В процесі навчання ваги ШНМ поступово стають такими, щоб кожний вхідний вектор формував відповідний вектор на виході. Таким чином, технічно навчання полягає у знаходженні коефіцієнтів зв'язків між нейронами.

Обчислення ваг нейронів.

Враховуючи нелінійність функції активації, обчислити w_{ij} за один крок не можливо (хоча деякі ШНМ це роблять), тому знаходження коефіцієнтів здійснюється ітеративно, поступово наближаючись до оптимальних значень. Корегування ваг нейронів і зсувів в нейроні Мас-Коллака- Пітса здійснюється по крокам за формулами:

$$w_{ij}(k+1)=w_{ij}(k)+\Delta w_{ij}$$
$$\theta_j(k+1)=\theta_j(k)+\Delta\theta_j$$

де:

k – номер кроку навчання;

i – номер нейрона попереднього шару;

j – номер нейрона поточного шару;

Δw_{ij} та $\Delta\theta_j$ обчислюється за деяким правилом.

Друга форма запису:

$$w_{ij}(k+1)=\alpha w_{ij}(k)+\beta\Delta w_{ij}$$
$$\theta_j(k+1)=\alpha\theta_j(k)+\beta\Delta\theta_j$$

де:

α – коефіцієнт моменту $(0<\alpha<1)$

β – коефіцієнт швидкості навчання $(0<\beta<1)$

Правила навчання.

Правила навчання визнають спосіб обчислення Δw_{ij}, $\Delta\theta_j$.

Правило Хебба (Hebb rule)

Дональд Хебб – канадський фізіолог та нейропсихолог висловив гіпотезу про те, що мозок збільшує підсилювальні властивості синапсів тоді, коли нейрони перед і за синапсами одночасно активовані. Гіпотеза Хебба у більш строгому формулюванні: властивість синапсу (посилення чи пригнічення) змінюється пропорційно добутку перед- і після синаптичної активності.

Правила навчання Хебба:

- Нейрон генерує сигнали {0, 1}
- Початкові ваги встановлюються випадковим чином
- Якщо вихід нейрона неправильний і дорівнює нулю, то необхідно збільшити ваги тих входів, на які була подана одиниця
- Якщо вихід нейрона неправильний і дорівнює одиниці, то необхідно зменшити ваги тих входів, на які була подана одиниця

Для принципу корегування ваг правило Хебба записується так:

$$\Delta w_{ij}=\beta x_i y_j$$

де:

β – коефіцієнт швидкості навчання $(0<\beta<1)$;

y_j – обчислене значення виходу;

x_i – значення входу

Крім цього використовується **Дельта правило** (Widrow-Hoff rule)

$$\Delta w_{ij}=\beta (y_j - d_j) x_i$$

де:

β – коефіцієнт швидкості навчання $(0<\beta<1)$;

y_j – обчислене значення виходу;

d_j – бажане значення виходу;

x_i – значення входу

Крім цього в алгоритмах навчання може застосуватись **Узагальнене дельта-правило:**

$$\Delta w_{ij}=\beta (y_j - d_j) y_j (1 - y_j) x_i$$

Етапи навчання штучних нейронних мереж

Основні етапи розв'язання задач за допомогою нейромереж:

- Збір даних для навчання;
- Підготовка і нормалізація даних;
- Вибір топології мережі;
- Експериментальний підбір характеристик мережі;
- Експериментальний підбір параметрів навчання;
- Власно навчання;
- Перевірка адекватності навчання;
- Коректування параметрів, остаточне навчання;
- Вербалізація мережі з метою подальшого використання.

Розглянемо докладніше ці етапи. Від якості виконання кожного з етапів залежить якість результату.

Збір даних для навчання

Вибір даних для навчання мережі і їхня обробка є самим складним етапом розв'язання задачі. Набір даних для навчання повинний задовольняти декільком критеріям:

- Репрезентативність – дані повинні ілюструвати дійсне положення речей у предметній галузі;
- Несуперечність – суперечливі дані в навчальній вибірці призведуть до поганої якості навчання мережі;
- Обсяг – як правило, кількість записів у вибірці повинна на кілька порядків перевершувати кількість зв'язків між нейронами в мережі. У протилежному випадку мережа просто «запам'ятає» усю навчальну вибірку і не зможе виконати узагальнення.

Підготовка і нормалізація даних

Навчання мережі на «сирому» наборі, як правило, не дає якісних результатів. Існує ряд способів поліпшити «сприйняття» даних мережею. Для покращення результатів навчання на даними перед навчанням здійснюють:

- ***нормалізацію*** – приведення всіх вхідних і вихідних змінних до одного діапазону. Найчастіше застосовують мінімакс-нормалізацію:

$$x_{norm}\frac{x-\min(X)}{max(X)-\min(X)}$$

Це дозволяє зменшити арифметичну помилку обчислення коефіцієнтів w_{ij} коли чисельні значення вхідних змінних різняться на декілька порядків (наприклад, дохід фірми обчислюється в мільйонах,

або Z-нормалізацію:

$$x_{norm}=\frac{x-M(X)}{\sigma(X)}$$

де:

$M(X)$ – середнє значення;

$\sigma(X)$ – середньоквадратичне відхилення.

При такій нормалізації більшість значень потрапляє в діапазон (–3σ; +3σ)

- ***фільтрацію*** – виправлення спотворень даних, викликаних шумами і іншими причинами.

Вибір топології мережі

Вибирати тип мережі необхідно виходячи з постановки задачі і наявних даних для навчання. Для навчання з учителем потрібна наявність для кожного елемента вибірки «експертної» оцінки. Іноді одержання такої оцінки для великого масиву даних просто неможливо. У цих випадках природним вибором є мережа, що навчається без учителя, наприклад, така як самоорганізуюча карта Кохонена або нейрона мережа Хопфілда. При розв'язанні інших задач, таких як прогнозування часових рядів, експертна оцінка вже присутня у вхідних даних і може бути виділена при їхній обробці. У цьому випадку можна використовувати багатошаровий перцептрон.

Експериментальний підбір характеристик мережі

Після вибору загальної структури потрібно експериментально підібрати параметри мережі. Для мереж, подібних перцептрону, це буде число шарів, число блоків у схованих шарах, наявність або відсутність обхідних з'єднань, передатні функції нейронів. При виборі кількості шарів і нейронів варто виходити з того, що здатності мережі до узагальнення тим вище, чим більше сумарне число зв'язків між нейронами. З іншого боку, число зв'язків обмежене зверху кількістю записів у навчальних даних.

Експериментальний підбір параметрів навчання

Після вибору конкретної топології, необхідно вибрати параметри навчання нейронної мережі. Цей етап особливо важливий для мереж, які навчаються с учителем. Від правильного вибору параметрів залежать не тільки те, наскільки швидко відповіді мережі будуть сходитися до правильних відповідей. Наприклад, вибір низької швидкості навчання β збільшить час сходження, однак іноді дозволяє уникнути паралічу мережі. Збільшення моменту навчання α може привести як до збільшення, так і до зменшення часу збіжності. Виходячи з такого суперечливого впливу параметрів, можна зробити висновок, що їх значення

потрібно вибирати експериментально, керуючись при цьому критерієм завершення навчання (наприклад, мінімізація похибки або обмеження за часом навчання).

Власне навчання мережі

У процесі навчання мережа у визначеному порядку переглядає навчальну вибірку. Порядок перегляду може бути послідовним, випадковим і т.д. Мережі, які навчаються без учителя, переглядають вибірку тільки один раз. При навчанні з учителем мережа переглядає вибірку багато разів, при цьому один повний прохід по вибірці називається епохою навчання.

Зазвичай набір вихідних даних поділяють на три частини – власне навчальну вибірку *(Train*), перевірочну *Validation ()* і тестову *(Test).* Можливі співвідношення між *Train, Validation ma Test :*70:20:10, хоча принцип поділу може бути довільним. Навчальні дані подаються мережі для навчання, перевірочні для фіксації ситуації «перенавчання», тестові використовуються для розрахунку похибки мережі (перевірочні і тестові дані для навчання мережі не повинні застосовуватись).

«Перенавчання» (overfitting) – це коли похибка на навчальній вибірці продовжує зменшуватися, а похибка на перевірочних даних збільшується. Це означає, що мережа перестала виконувати узагальнення і просто «запам'ятовує» навчальні дані. У таких випадках навчання зазвичай припиняють. У процесі навчання можуть проявитися інші проблеми, такі як параліч або попадання мережі в локальний мінімум поверхні похибок. Неможливо заздалегідь передбачити прояв тієї або іншої проблеми, так само як і дати однозначні рекомендації щодо їх розв'язання.

Перевірка адекватності навчання

Навіть у випадку успішного, на перший погляд, навчання мережа не завжди навчається саме тому, чого від неї хоче розробник. Відомий випадок, коли мережа навчалася розпізнаванню зображень танків по фотографіях, однак пізніше з'ясувалося, що всі танки були сфотографовані на тому самому тлі. У результаті мережа «навчилася» розпізнавати цей тип ландшафту, замість того, щоб «навчитися» розпізнавати танки.

Таким чином, мережа «розуміє» не те, що від неї було потрібно, а те, що найпростіше узагальнити.

Принципи алгоритмів навчання

Математично навчання мережі розглядається як задача оптимізації.

При такому підході мережа буде навченою, коли функція втрат досягає глобального мінімуму. При цьому коефіцієнти w_{ij} отримані для цього випадку будуть оптимальними. Таким чином треба аналітично визначити функцію втрат, так щоб вона залежала від коефіцієнтів і інших параметрів нейронів і знайти її мінімум. Для цього використовуються метод градієнта.

Градієнтні методи

Базовою ідеєю всіх алгоритмів навчання є облік локального градієнта для вибору траєкторії якнайшвидшого спуску по функції втрат. Функція втрат, проте, може мати кілька локальних мінімумів, що представляють суб-оптимальні рішення. Тому градієнтні методи зазвичай доповнюються елементами

стохастичної оптимізації, щоб запобігти попаданню мережі в такі локальні мінімумаи. Ідеальний метод навчання повинен знайти глобальний оптимум конфігурації мережі.

Якщо використовувані функції активації нейронів є диференційованими, то вводиться деяка опукла цільова (вартісна, енергетична, помилка) функція *E*, і методом градієнтного спуску обчислюються коефіцієнти w_{ij}, при яких *E* досягає мінімуму рис. 6.10, 6.11.

В якості функції втрат (помилки) найчастіше використовують *суму квадратів помилок*. Для однієї навчальної пари вона виглядає наступним чином:

$$E = \frac{1}{2}\sum_{j=1}^{m}(y_j - d_j)^2 \to \min$$

де:

y_j – обчислене значення виходу;

d_j – бажане.

Для всієї навчальної вибірки так:

$$E_s = \sum_{k=1}^{L}E(k) = \frac{1}{2}\sum_{k=1}^{L}\sum_{j=1}^{m}(y_j^k - d_j^k)^2 \to \min$$

Щоб «дістатися» до мінімуму помилки, нам необхідно «рухатися» в бік, протилежний градієнту функції втрат, тобто, на кожній групі вірних відповідей, до кожної ваги w_{ij} додавати величину:

$$\Delta w_{ij} = \beta\frac{\partial E}{\partial w_{ij}(k)}$$

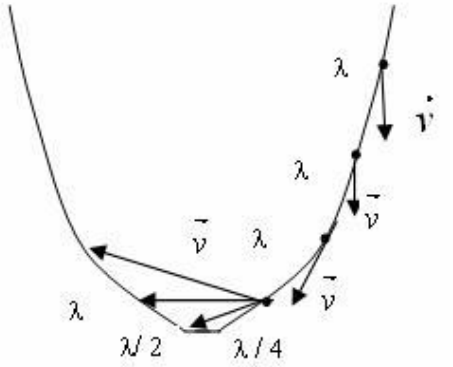


Рисунок 6.10 – Метод градієнту.

Недоліком методу є те, що ми можемо попасти в локальний мінімум рис.6.12., тобто попадання в глобальний мінімум не гарантовано.

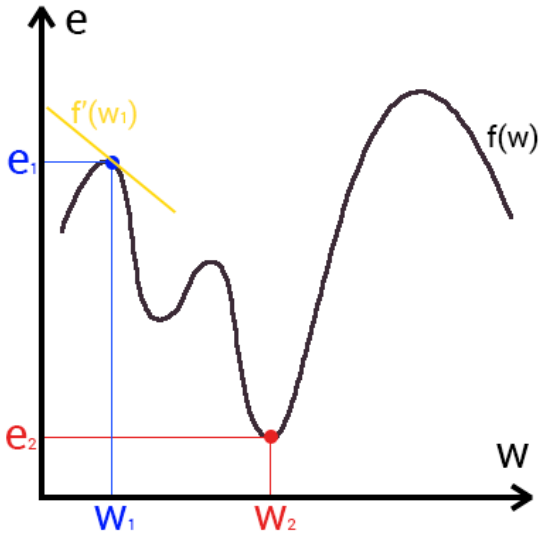


Рисунок 6.11 – Локальний і глобальний мінімуми функції втрат.

За цим методом розроблений Румерхартом і колегами (Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. *Nature* 323, 533–536 (1986)) алгоритм навчання під назвою алгоритм зворотного поширення помилки. Суть полягає в тому, що на кроці (епохи) навчання обчислюється помилка мережі на виході і від її значення обчислюються w_{ij} для вихідного шару, потім помилка пропорційно передається на попередній шар і так до вхідного шару. Розглянемо алгоритм зворотного поширення помилки по крокам.

Алгоритм зворотного поширення помилки

Крок 1. Ініціалізація мережі: вагові коефіцієнти і зсуви мережі приймають малі випадкові значення.

Крок 2. Задання випадкових образів – визначення елемента навчальної множини: пари “вхід – вихід” для досліджуваної функції. Входи (x_1, x_2... x_N), повинні розрізнятися для всіх прикладів навчальної множини.

Крок 3:

Крок 3.1. Обчислення вихідного сигналу:

$$S_{i_m} = \sum_{i_{m-1}}^{N_{m-1}} w_{i_m j_{m-1}} y_{i_{m-1}} - \theta_{i_m}, \quad y_{i_m} F(S_{j_m}), \quad i_m = 1,2,...,N_m$$

де:

S – вихід суматора;

w – вага зв’язку;

y – вихід нейрона;

θ – зсув;

i – номер нейрона;

N – кількість нейронів у шарі;

m – номер шару,

– кількість шарів;

F –функція активації.

Крок 3.2. Обчислення δ^y для вихідного шару:

$$\delta_j^y = y_j(1-y_j)(e_j - y_j)$$

де:

e_j – бажаний вихід нейрона j;

y_j – поточний вихід нейрона j.

Для сигмоїдної функції активації на попередньому шарі:

Крок 3.3. Обчислення для прихованих шарів:

$$\delta_j^h = y_j(1-y_j)\sum_k \delta_k w_{jk}$$

Крок 4. Корегування ваг:

$$\Delta w_{ij}(k+1) = \Delta w_{ij}(k) + \beta \delta_j x_i$$
$$w_{ij}(k+1) = \alpha w_{ij}(k) + \Delta w_{ij}(k)$$

Крок 5. Повернення до кроку 2

6.7 Нейронна мережа Хопфілда

Мережу Хопфілда називають асоціативною, тому що, як людина може згадати образ по його деталях, так і ця мережа при подачі на вхід пошкодженого образа на виході з'являється еталонний образ, що зберігається в її пам'яті. У 1982 р. біофізик США Хопфілд вперше представив свою асоціативну мережу у Національній Академії Наук, і вона отримала його прізвище.

Завдання, розв'язуване даною мережею як асоціативною пам'яттю, формулюється в такий спосіб. Відомий деякий набір двійкових сигналів (зображень, звукових оцифровок, інших даних, що описують якісь об'єкти або характеристики процесів), які вважаються еталонними. Мережа повинна вміти з довільного неідеального сигналу, поданого на її вхід, виділити ("згадати" за частковою інформацією) відповідний еталон (якщо такий є) або "дати висновок" про те, що вхідні дані не відповідають жодному зі еталонів. У загальному випадку, будь-який сигнал може бути описаний вектором $X = \{ x_i: i=0...n-1\}$, n – число нейронів у мережі й розмірність вхідних і вихідних векторів. Коли мережа розпізнає (або "згадає") який-небудь еталон на основі пред'явлених їй даних, її виходи будуть містити саме його, тобто $Y = X^E$, де Y – вектор вихідних значень мережі: $Y = \{ y_i: i=0,...n-1\}$. У противному випадку, вихідний вектор не збіжиться з жодним еталоном.

Нейронна мережа Хопфілда (рис. 6.12) складається з N штучних нейронів, аксон кожного нейрона зв'язаний з дендритами інших нейронів, що створює зворотний зв'язок.

Якщо, наприклад, сигнали являють собою якісь зображення, те, відобразивши в графічному виді дані з виходу мережі, можна буде побачити картинку, що повністю збігається з однією зі зразкових (у випадку успіху) або ж "вільну імпровізацію" мережі (у випадку невдачі).

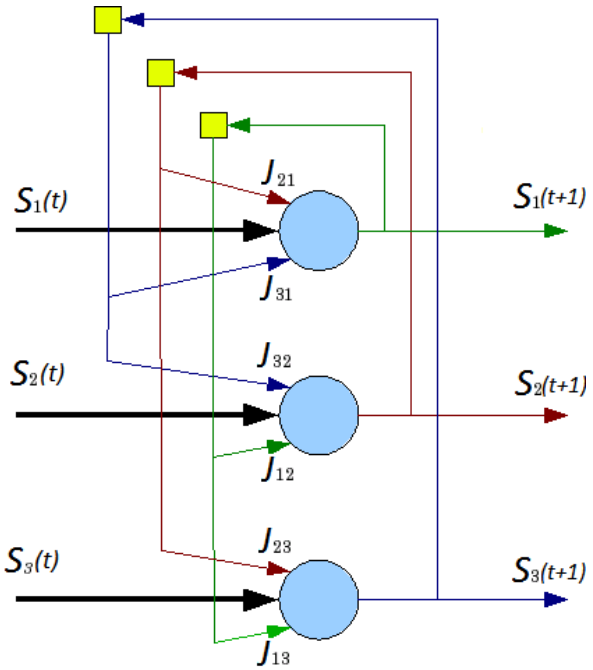


Рисунок 6.11 – Топологія мережі Хопфілда.

Кожен нейрон системи може приймати один з двох станів (що аналогічно виходу нейрона з пороговою функцією активації).

Кожен нейрон може знаходитись у двох станах $S(t) \in \{-1; +1\}$

де S (t) - стан нейрона в момент t. «збудження» нейрона відповідає +1, а «гальмування» –1.

У мережі Хопфілда матриця зв'язків є симетричною ($w_{ij} = w_{ji}$), а діагональні елементи матриці покладаються рівними нулю ($w_{ii} = 0$), що виключає ефект впливу нейрона на самого себе та є необхідною умовою стійкості для мережі Хопфілда, але не достатньою в процесі роботи мережі.

Мережі Хопфілда мають наступні властивості:

1. Симетрія дуг: мережі містять n нейронів, з'єднаних один з одним.

Кожна дуга (з'єднання) характеризується вагою w_{ij}, причому:

$$\forall i, j \in N; i \neq j; \exists w_{ij}$$

де N – множина нейронів N = {1, 2, …, n}..

2. Симетрія ваг: вага з'єднання нейрона n_i з нейроном n_j дорівнює вазі зворотного з'єднання:

$$w_{ij} = w_{ji} \; ; \; w_{ii} = 0 \; .$$

3. Бінарні входи: мережа Хопфілда обробляє бінарні входи {0,1} або {–1, 1}. У літературі зустрічаються моделі мереж як зі значеннями 0 й 1, так й –1, 1. Для структури мережі це байдуже. Однак формули для розпізнавання образів (зображень) при використанні значень –1 й 1 для входів і виходів нейронів мережі Хопфілда виходять наочніше.

Перед початком роботи експерт по *еталонних сигналах входу тільки раз* розраховує вагові коефіцієнти нейронів, і потім ваги не змінюються. Тому кажуть, що еталонні сигнали «записані в її вагах». А дехто стверджує, що ця

мережа функціонує *«без навчання»* (бо вони під навчанням розуміють не одноразове встановлення ваг людиною, а поступове уточнення їх величин).

Значення вагових коефіцієнтів розраховуються у вигляді квадратної матриці W[n:n] по підсумку добутків *n*-вимірних еталонних векторів $X_i^E(i=1,2,...,m)$, які запам'ятовуються мережею Хопфілда

$$W[n:n]=\sum_{E=1}^{m}\left[X^E(X^E)^T\right]$$

де X^E – *E*-й еталонний вектор; $E=1, 2,..., m$ – порядковий номер еталонного вектора X^E; m – загальна кількість еталонних векторів, які запам'ятовується (звичайно $m < 0{,}15\ n$, n – кількість елементів еталонних векторів X^E).

Елементи квадратної матриці W[n:n] мають вигляд вагових коефіцієнтів зворотних зв'язків між нейронами w_{ij} ($i=1,2,..., n$ – індекс нейрону, від якого отримується зворотній зв'язок; $j=1,2,..., n$ – індекс нейрону, який отримує зворотній зв'язок; n – загальна кількість елементів еталонних векторів X^E) і визначаються наступним чином:

- Порядковий номер колонки симетричної квадратної матриці W[n:n] дорівнює порядковому номеру i= 1, 2,…, n нейрона, від якого отримується зворотній зв'язок.
- Порядковий номер рядка симетричної квадратної матриці W[n:n] дорівнює порядковому номеру j= 1, 2,…, n нейрона, який отримує зворотній зв'язок (i≠ j).

Розрізняють три стадії (фази) функціонування мережі:

1. Ініціалізація. Встановлюються вагові коефіцієнти за наведеною. формулою.

2. Подання і запам'ятовування вхідного образу X^E. Вимкнення вхідного образу X^E.

3. Обчислення станів нейронів шляхом циклічних перерахунків з визначенням елементів виходу Y до отримання їх сталого значення.

Припустимо, що на вхід навченої мережі Хопфілда ми подали на короткий час сигнал X^C від об'єкта, який потрібно віднести до найближчого еталона (X^{E1} або X^{E2}).

Навчена мережа Хопфілда запам'ятовує цей введений сигнал X_C, вимикає входи і після кількаразової циклічної процедури перерахунків елементів $У^T=(y_1, y_2, y_3, y_4)$, дає на виходах стабільні елементи $У^T=(y_1, y_2, y_3, y_4)$ еталону, який є найближчим до введеного сигналу X_C.

Існує *асинхронний та синхронний алгоритм* роботи НМ Хопфілда:

1. *В синхронному режимі* всі нейрони одночасно змінюють свій стан в момент часу «t+1» відносно стану на момент «t».

2. *В асинхронному режимі роботи*:

- в момент часу «t+1» змінює свій стан лише один нейрон відносно стану на момент «t»;

- потім в момент часу «t+2» змінює свій стан інший нейрон відносно стану мережі на момент «t+1» і т. д.

Досягнутий стаціонарний стан мережі є однаковим і не залежить від синхронності алгоритму роботи нейронів.

Елементи вихідного сигналу $Y^T=(y_1, y_2, y_3, y_4)$ циклічно замінюються на нові перераховані значення. Для цього розраховуються значення *NET* нейронів і у виходи $Y^T=(y_1, y_2, y_3, y_4)$ записуються нові значення з врахуванням дії нелінійних активаційних функцій нейронів.

Далі виконується перевірка, чи змінились (порівняно з попереднім станом) вихідні значення нейронів $Y=(y_1, y_2, y_3, y_4)$ за останню ітерацію. Якщо $Y^T=(y_1, y_2, y_3, y_4)$ змінились, то перерахунки повторюються, а в іншому випадку (якщо отримуємо стабільні виходи) – зупиняються. У цьому випадку вихідний вектор являє собою еталон, який найліпщим чином відповідає вхідним даним. Тобто ми повинні переконатися у тому, що мережа Хопфілда через кілька ітерацій не змінює отриманий еталон, бо це є ознакою завершення розпізнавання.

Якщо мережа Хопфілда не розпізнає образ, то це може бути пов’язаним з такими причинами:

1. Кількість образів-еталонів m, що запам’ятовуються, не повинна перевищувати значення величини, приблизно рівної $0{,}15n$, де n – кількість нейронів $N_1, N_2, \ldots, N_n$, яка дорівнює кількості ознак об’єктів.
2. Із-за великої схожості образів, мережа ***їх плутає***.

У деяких публікаціях навчання мережі Хопфілда відносять до навчання без учителя.

6.7 Інструментальні засоби ШНМ

В даний час відомо більше 200 нейропакетів, що випускаються рядом фірм та окремими дослідниками і дозволяють конструювати, навчати і використовувати нейронні мережі для вирішення практичних завдань. Трудомісткість розробки НЕС скорочується в разі застосування готових нейромережевих програм. Можливість використання нейромереж включена також практично у всі відомі статистичні пакети. Критерії порівняння нейропакетів: простота вживання, наочність інформації, що представляється, можливість використовувати різні структури, швидкість роботи, наявність документації. Вибір визначається кваліфікацією і вимогами користувача

Для застосування методів нейронних мереж в процесі інтелектуального аналізу даних в бізнес-додатках розроблений ряд інструментальних засобів високого рівня. До них відносяться в першу чергу системи MathLab, STATISTICA Neural Networks, NeuroSolutions, BrainMakerPro та інші.

MathLab

Пакет MathLab надає користувачам можливість роботи з нейронними мережами. Стандартне постачання MathLab «Neural Network Toolbox» надає широкі можливості для роботи з нейронними мережами всіх типів. Перевага цього пакету полягає в тому, що при його використанні користувач не обмежений моделями нейронних мереж і їх параметрами, жорстко закладеними в нейросимуляторі, а має можливість самостійно сконструювати ту мережу, яку

вважає оптимальною для вирішення поставленого завдання. Пакет містить функції командного рядка і графічний інтерактивний майстер для швидкого покрокового створення нейромереж. Ключовими можливостями Neural Network Toolbox є: графічний інтерфейс користувача і майстер покрокового створення нейронних мереж; підтримка найбільш поширених мережевих парадигм; повний набір засобів для тренування нейромереж з вчителем і без; нейромережі з динамічним навчанням, включаючи нейромережі з запізненням, нелінійні і авторегресійні (NARX); підтримка Simulink для моделювання нейромережі, створення блоків на основі розроблених нейромережевих структур для адаптивних систем управління; модульне представлення мережі, що дозволяє створювати необмежене число шарів і міжмережевих зв'язків; візуалізація топології нейронної мережі.

STATISTICA Neural Networks – потужне і надзвичайно швидке середовищем аналізу нейромережевих моделей, що надає наступні можливості:

пре- і пост-процесування, включаючи вибір даних, кодування номінальних значень, нормалізація, видалення пропущених даних, регресії і завдання часових рядів; потужні методи розвідувальних і аналітичних технологій, зокрема (Аналіз головних компонент і Пониження розмірності); найсучасніші, оптимізовані і потужні алгоритми навчання мережі, повний контроль над всіма параметрами, що впливають на якість мережі, такими як функції активації і помилок, складність мережі; підтримка комбінацій нейромереж практично необмеженого розміру, створених в наборах мереж - Network Sets, вибіркове навчання нейромережевих сегментів; об'єднання, і збереження наборів мереж в окремих файлах; повна інтеграція з системою STATISTICA).

NeuroSolutions

NeuroSolutions – це середовище розробки програмного забезпечення для нейронних мереж, розроблене NeuroDimension.

Сімейство продуктів NeuroSolutions – це програмне забезпечення для аналізу даних і створення високоточних і прогнозних моделей з використанням інтелектуального автоматичного пошуку топології нейронної мережі за допомогою передових розподілених обчислень. Це інтерфейс з алгоритмами навчання за допомогою інтуїтивно зрозумілих майстрів або простого у використанні інтерфейсу Excel. Програмне забезпечення надає три окремі майстри для автоматичного створення моделей нейронних мереж: Data Manager (Менеджер даних), Neural Builder (Конструктор нейронної мережі), Neural Expert.

Neuroph

Neuroph – це проект з відкритим кодом, розміщений у SourceForge за ліцензією Apache. Neuroph – це легкий каркас нейронної мережі Java для розробки загальних архітектур нейронних мереж. Користувачі можуть взаємодіяти з Neuroph за допомогою графічного інтерфейсу GUI-based tool і бібліотек Java. Обидва підходи спираються на основну ієрархію класів, яка будує штучні нейронні мережі з шарів нейронів.

Darknet

Darknet – це фреймворк нейронної мережі з відкритим кодом, написаний на C і CUDA і підтримує CPU і GPU обчислення. Це згортка нейронна мережа, яка має дев'ятнадцять шарів глибини. Попередньо навчена мережа може класифікувати зображення на 1000 категорій об'єктів, таких як клавіатура, миша, олівець і багато тварин. В результаті мережа навчилася багатому представленню функцій для широкого діапазону зображень.

Darknet встановлюється лише з двома необов'язковими інструментами, такими як OpenCV, якщо користувачі хочуть мати більш широкий спектр підтримуваних типів зображень, або CUDA, для використання GPU (графічних процесорів). Користувачі можуть почати, просто встановивши базову систему, яка була протестована лише на комп'ютерах Linux і Mac.

Stuttgart Neural Network Simulator

Stuttgart Neural Network Simulator (SNNS) – нейронний симулятор, спочатку розроблений в Штутгартському університеті. Метою проекту SNNS є створення ефективного та гнучкого середовища моделювання для дослідження та застосування нейронних мереж. Ядро симулятора працює на внутрішніх мережевих структурах даних нейронних мереж і виконує всі операції навчання. Він підтримує довільні топології мережі і підтримує концепцію сайтів. SNNS може бути розширений користувачем за допомогою визначених користувачем функцій активації, вихідних функцій, функцій сайту та процедур навчання, які написані як прості програми на C і пов'язані з ядром симулятора.

Переваги ШНМ:

- Формалізація заміняється навчанням на прикладах;
- Сталість до шумів у вхідних даних;
- Адаптація до змін навколишнього середовища;
- Універсальність;
- Велика швидкодія (орієнтація на паралельні обчислення)
- Простота застосування;

Недоліки ШНМ:

- Не гарантована повторюваність і однозначність результатів;
- Труднощі пояснення прийнятих рішень.

7. ГЛИБОКІ НЕЙРОННІ МЕРЕЖІ

Deep learning *(Глибоке навчання)* – це розділ машинного навчання, спрямований на побудову ієрархічних моделей шляхом використання високорівневих масових абстракцій даних на основі глибинного графу з багатьма обробними шарами, які здійснюють лінійні або нелінійні перетворення, тобто це – прогресуючий високорівневий витяг ознак з сирих (необроблених) первісних вхідних даних.

У центрі глибого навчання є глибокі нейронні мережі та методи їхньої побудови.

Deep neural network (DNN) *Глибока нейронна мережа* (ГНМ) – це різновид ШНМ, що має багато шарів обробки даних, яка перетворює вхідні дані у вихідні, ієрархічно виділяючи та агрегуючи ознаки, підвищуючи рівень абстракції даних в напрямку від входів до виходів.

Порівняно з класичними ШНМ, ГНМ за рахунок збільшення кількості нейроелементів та зв’язків отримують більшу обчислювальну потужність і здатність моделювати більш складні залежності, а за рахунок спеціалізації шарів та високої ієрархічності обробки даних стають більш зручними для сприйняття та аналізу людиною. При цьому спеціалізація шарів обробки даних у ГНМ робить їх більш пристосованими до інтеграції в мережеву модель апріорної інформації про предметну область. Проте, як правило, конкретні парадигми ГНМ мають більш обмежене застосування у конкретних задачах (наприклад, деякі архітектури можуть застосовуватися лише для розпізнавання зображень і не придатні для інших задач).

7.1. Згорткові нейромережі

Convolutional Neural Network, CNN, ConvNet (Згорткові нейронні мережі) – це глибинні ШНМ прямого поширення. Фактично вони є різновидом багатошарової нейронної мережі, адаптованої до обробки зображень (рис. 7.1).

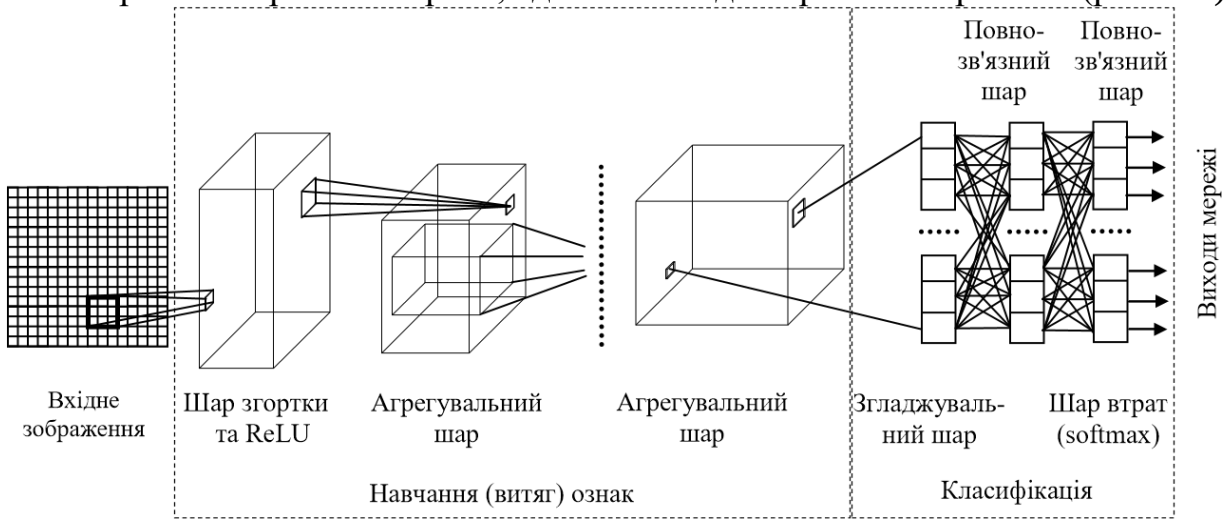


Рисунок 7.1 – Топологія згорткової нейронної мережі

Згорткові нейронні мережі (ЗНМ) інспіровані організацією зорової кори тварин, де окремі нейрони кори реагують на стимули лише в обмеженій області

зорового поля, так званому рецептивному полі. Рецептивні поля різних нейронів частково перекриваються таким чином, що вони покривають усе зорове поле.

ЗНМ складається з шарів входу та виходу, а також із декількох прихованих шарів. На відміну загальної парадигми БНМ, приховані шари ЗНМ зазвичай є спеціалізованими і складаються зі згорткових шарів, агрегувальних шарів, повноз'єднаних шарів та шарів нормалізації.

Convolutional layer (Згортковий шар) застосовує до входу операцію згортки в двовимірному просторі операція згортки між двома функціями визначається наступним чином:

$$g(x, y) = f(x, y) \odot h(x, y)$$

де: f(x, y) – вхідне зображення;

h(x, y) – ядро згортки;

$\odot$ – операція згортки.

На рис. 7.2 наведений приклад обчислення одного значення функції g(x,y) для рецептивного поля розміром 3x3

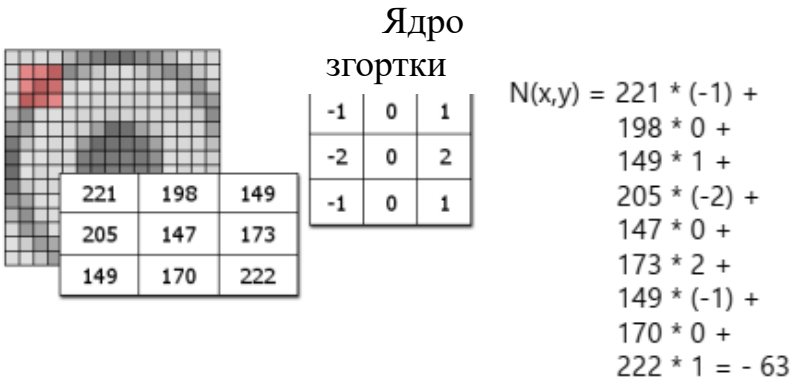


Рисунок 7.2 – Обчислення згортки

Pooling layer (Агрегувальний шар) призначений для локального або глобального агрегування та об'єднує виходи кластерів нейронів одного шару до одного нейрону наступного шару (рис. 7.3).

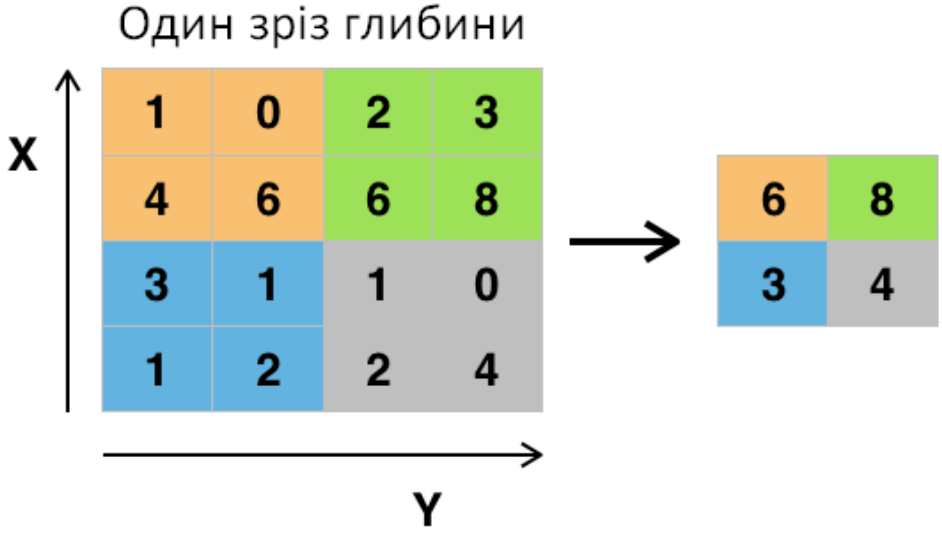


Рисунок 7.3 – Обчислення максимізаційного агрегування

Існує декілька нелінійних функцій для реалізації агрегування, серед яких найпоширенішою є **max pooling** (максимізаційне агрегування). Воно розділяє вхідне зображення на набір прямокутників без перекриттів, і для кожної такої підобласті виводить її максимум. На рис.7.3 наведено приклад max pooling.

Агрегування є різновидом нелінійного зниження дискретизації, ідея якого полягає у тому, що точне положення ознаки не так важливе, як її грубе положення відносно інших ознак. Агрегувальний шар слугує поступовому скороченню просторового розміру подання для зменшення кількості параметрів та обсягу обчислень у мережі, і відтак також для контролю перенавчання. В архітектурі ЗНМ є звичним періодично вставляти агрегувальний шар між послідовними згортковими шарами.

Найбільш відомим застосуванням згорткових мереж є розпізнавання рукописних символів (рис. 7.4).

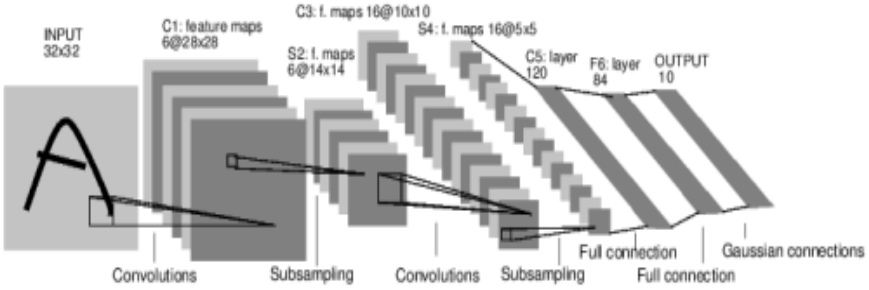


Рисунок 7.3 – Структура CNN мережі для розпізнавання рукописних символів

Вхідні дані мережі:

- Інтенсивність піксела в зображенні
- Кількість значень 782 (28*28 пікселів)

Вхідний шар:

- 800 нейронів

Вихідний шар:

- 10 нейронів
- Ймовірність того, що на зображенні дана цифра

7.2. Приклади згорткових мереж

Є багато варіантів застосування CNN, такі як Deep Convolutional Neural Network (DCNN), Region-CNN (R-CNN), Fully Convolutional Neural Networks (FCNN), Mask R-CNN та інші.

Підхід Transfer Learning

Використання попередньо навченої мережі для інших завдань.

Приклади попередньо навчених мереж:

- AlexNet
- VGG16 і VGG19
- Google Inception (кілька версій)
- Microsoft ResNet (50 шарів і більше) Перенесення навчання:
- Модифікація архітектури попередньо навченої мережі
- донавчання (fine tuning) на своєму наборі даних

Параметри деяких мереж подані на рис. 7.4., 7.5

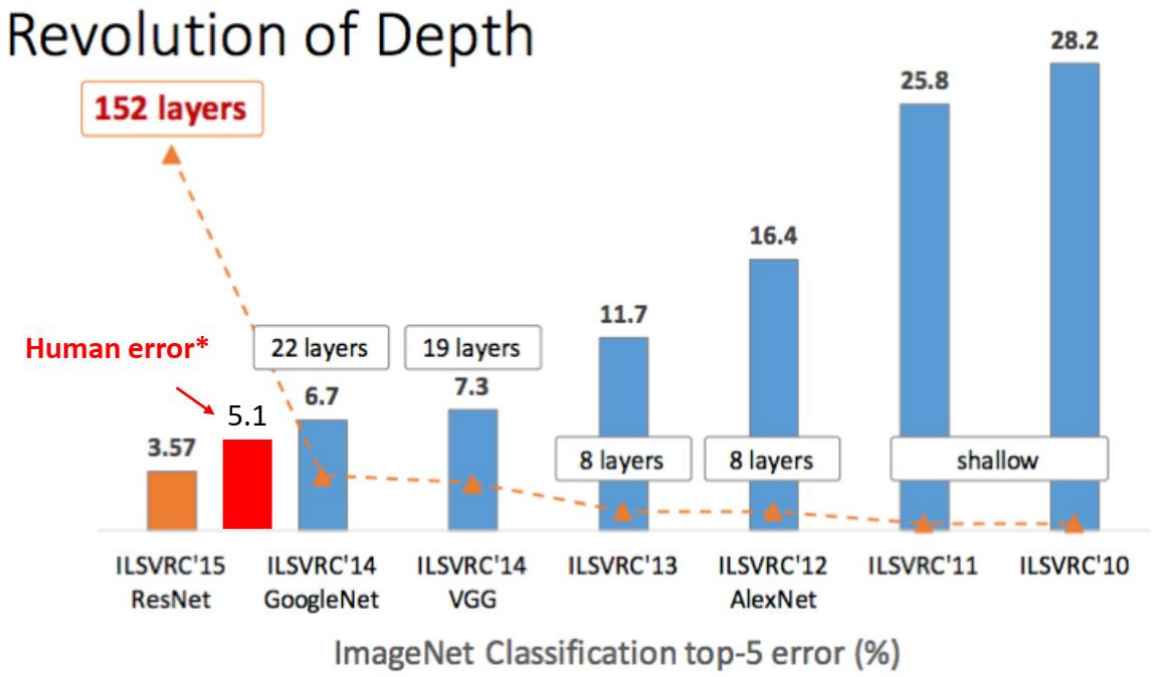


Рисунок 7.4 – Приклади мереж глибокого навчання

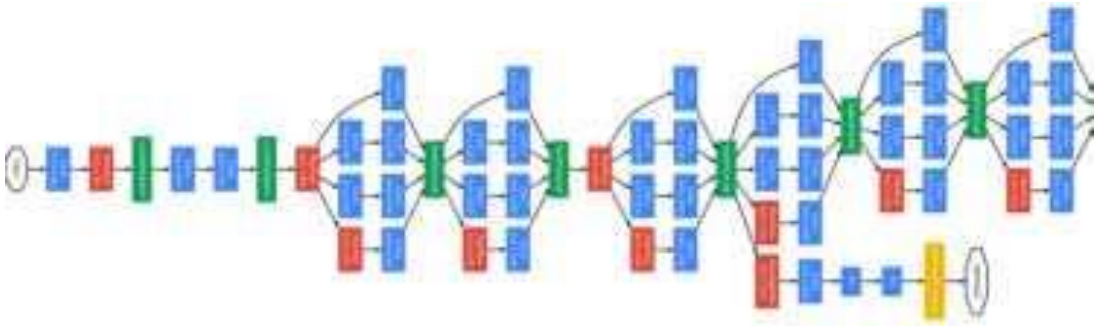


Рисунок 7.5 – Архітектура мережі GoodleNet (2014)

Параметри деяких мережі GoodleNet:

- Максимальна глибина: 22 шари з параметрами.
- Повнозв’язних шарів немає.
- на 12 менше параметрів (ніж у AlexNet).
- Основний блок - Inception module (9 штук).

7.3. Програмні засоби Deep Learning

TensorFlow

Назва TensorFlow походить від операцій над багатовимірними масивами даних, які називають «тензорами».

Платформа спочатку розроблена командою Google Brain і використовуються в сервісах Google для розпізнавання мови, виділення обличь на фотографіях, визначення схожості зображень, відсіювання спаму в Gmail, підбору новин у Google News і організації перекладу з урахуванням смислу.

TensorFlow є системою машинного навчання Google Brain другого покоління стала відкритим програмним забезпеченням 9 листопада 2015 року. TensorFlow – відкрита програмна бібліотека для машинного навчання цілій низці задач, розроблена компанією Google для задоволення її потреб у системах, здатних будувати та тренувати нейронні мережі для виявлення та розшифровування образів та кореляцій, аналогічно до навчання й розуміння, які застосовують люди.

Keras

Keras – відкрита нейромережна бібліотека, написана мовою Python. Вона здатна працювати поверх TensorFlow, Microsoft Cognitive Toolkit, R, Theano та PlaidML. Keras відповідає найкращим методам зниження когнітивного навантаження. Він пропонує послідовні та прості API та мінімізує кількість дій користувача, необхідних для поширених випадків використання. Keras надає чіткі й ефективні повідомлення про помилки та має велику документацію та посібники для розробників. Бібліотека глибокого навчання Keras забезпечує легке та швидке створення прототипів завдяки повній модульності, мінімалізму та розширюваності. Він підтримує згорткові нейронні мережі та рекурентні мережі, а також їх комбінації.

Keras містить численні втілення широко вживаних нейромережних будівельних блоків, таких як шари, цільові та передавальні функції, оптимізувальники та безліч інструментів для спрощення роботи із зображеннями та текстом, щоб спрощувати кодування, потрібне для написання глибинно-нейромережного коду.

Код Keras розміщено на GitHub, а до форумів спільнотної підтримки належать сторінка питань GitHub та канал Slack. Keras дає своїм користувачам можливість виробляти продукти на основі глибинних моделей для смартфонів (iOS та Android), веб-сайтів та віртуальної машини Java. Вона також дозволяє використовувати розподілене тренування моделей глибинного навчання на кластерах графічних (ГП) та тензорних (ТП) процесорів переважно у зв’язці з CUDA.

NVIDIA DIGITS

Навчальна система GPU Deep Learning NVIDIA (DIGITS) передає можливості глибокого навчання в руки інженерів і науковців з даних. Програмна програма використовує високоточні глибокі нейронні мережі для швидкої класифікації зображень, сегментації та виявлення об’єктів.

DIGITS спрощує звичайні завдання глибокого навчання, такі як керування даними, проектування й навчання нейронних мереж у системах із кількома графічними процесорами, моніторинг продуктивності в режимі реального часу за допомогою розширених візуалізацій та вибір найкращої моделі з браузера результатів для розгортання. Він повністю інтерактивний, тому науковці з даних можуть зосередитися на проектуванні та навчанні мереж, а не на програмуванні та налагодженні.

Tflearn

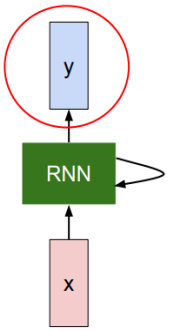
Tflearn – це модульна та прозора бібліотека глибокого навчання, побудована на основі Tensorflow. Програмне забезпечення розроблено для

надання API вищого рівня для TensorFlow, щоб полегшити та прискорити експерименти, залишаючись при цьому повністю прозорим і сумісним з ним.

API високого рівня наразі підтримує більшість останніх моделей глибокого навчання, таких як Convolutions, LSTM, BiRNN, BatchNorm, PreLU, залишкові мережі та генеративні мережі. Заглядаючи наперед, Tflearn також має на меті залишатися в курсі останніх технологій глибокого навчання, і зараз він знаходиться на ранній стадії розвитку.

7.4. Рекурентні нейронні мережі

В **Recurrent Neural Netwo**rk (рекурентних нейронних мережах) використовується зворотній зв'язок.



Усередині архітектури мережі розташовується базова рекурентна комірка. Модель приймає деякі вхідні дані x і відправляє їх в RNN, яка має прихований внутрішній стан. Цей стан оновлюється кожного разу, коли в RNN надходять нові дані. Усередині комірки обчислюється рекурентне співвідношення за допомогою функції f, яка залежить від ваг w:

$$h_t=f_w(h_{t-1}, x_t)$$

- h_t – новий стан;
- f_w – деяка функція з параметрами w;
- h_{t-1} – попередній стан;
- x_t – вхідний вектор на часовому кроці t

Новий стан h_t, залежить від прихованого стану h_{t-1} і поточного входу x_t.

Коли в модель поступають наступні вхідні дані, отриманий прихований стан h_t передається в цю ж функцію, і весь процес повторюється.

Щоб генерувати вихідні дані у кожен момент часу, в модель додаються повнозв'язні шари, які постійно обробляють стану h_t і видають засновані на них прогнози. При цьому функція f і ваги w залишаються незмінними.

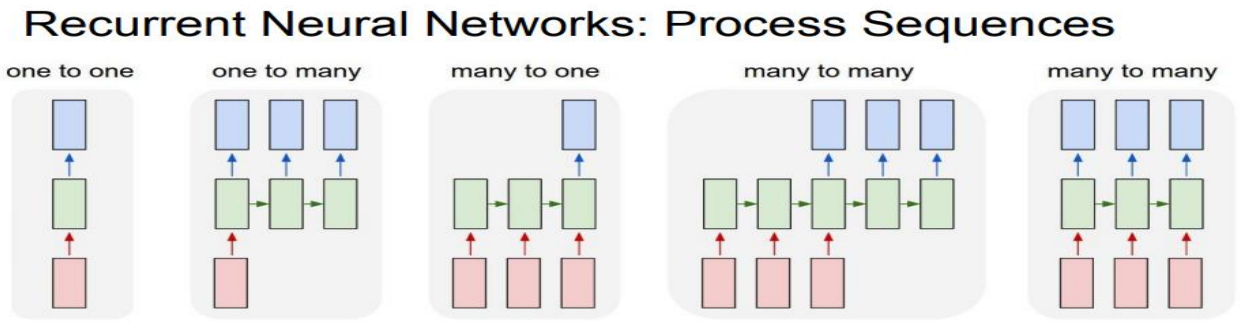


Рисунок 7.6 – Архітектури Recurrent Neural Network

❖ «one to one» – моделі з визначеним розміром вхідних і вихідних даних;

- «one to many» – при заздалегідь заданому типі і розмірі вхідного об’єкта можна отримати вихід різної довжини (такий підхід застосовується в популярній задачі опису зображень (image captioning).
- «many to one» – на вхід подаються дані нефіксованого розміру, на виході – чітко визначені їх характеристики (так, наприклад, можна за фрагментом відео визначати вид активностей або дії, які в ньому відбуваються);
- «many to many» – розміри як вхідних, так і вихідних даних варіюються (одними з задач, які вони розв’язують є машинний переклад (вхідна і перекладена фрази можуть бути різної довжини) і покадрова класифікація відео).

Рекурентні нейромережі дуже корисні навіть при вирішенні завдань «one to one». Розглянемо популярну проблему розпізнавання рукописних цифр. Замість того, щоб просто зробити один прямий прохід і відразу видати рішення, рекуррентна мережа швидко «переглядає» різні частини зображення. У термінології цей процес називається «проблиск» (glimpse). Зробивши кілька таких проблисків, модель приймає остаточне рішення про те, що зображене на фотографії. Це дозволяє істотно підвищити точність розпізнавання і краще контролювати процес навчання.

Image captioning

Суть цієї задачі полягає в тому, щоб нейромрежа склала текстовий опис фотографії. Для цього необхідно спочатку класифікувати об’єкти на зображенні, а потім передати результат (одну або кілька міток) в мовну рекурентну модель, яка зможе скласти з них осмислену фразу. При цьому ми діємо точно так, як у випадку з звичайною мовною моделлю: перетворимо мітку зображення в вектор, який обробляється декодером. Щоб рекурентна мережу розуміла, де саме починається речення, під час навчання на її вхід подається стартовий розпізнавальний знак (<START> token). Для побудови фрази використовується заздалегідь підготовлений словник.

При переході в кожний наступний прихований стан зберігаються як вже згенеровані слова, так і інформація про зображення. В кінці речення в нейромережу відправляється фінальний токен (<END>). Під час тестування модель вже самостійно визначає, де має починатися і закінчуватися опис зображення.

Image captioning + Attention

Моделі, засновані на увазі (attention) трохи більш просунуті, ніж звичайні нейромережі. Вони можуть концентруватися на окремих частинах зображення, що дозволяє уникнути зашумлення даних.

Ідея полягає в тому, що сгортовна мережа тепер буде генерувати не один вектор, що описує все зображення, а набір векторів для декількох ділянок вхідного знімка. На додаток до роботи зі словником на кожному часовому кроці модель також виробляє розподіл по точках на зображенні, які вона обробляє в даний момент. Це дозволяє їй навчитися знаходити найбільш важливі ділянки, на яких необхідно фокусуватися.

Після навчання моделі можна побачити, що вона як би переносить свою увагу по зображенню для кожного слова, що генерується.

Існують також поняття м’якої і жорсткої уваги (soft and hard attention). При м’якій увазі ми беремо зважену комбінацію ознак по всьому зображенню, тоді як у разі жорсткої уваги ми змушуємо модель вибирати тільки одну невелику ділянку для обробки на кожному кроці. При цьому жорстка увага, строго кажучи, не є функцією, що диференціюється. Тому для навчання такої моделі необхідно використовувати більш витончені прийоми, ніж звичайне зворотне поширення помилки. Приклад наведений на рис. 7.7.

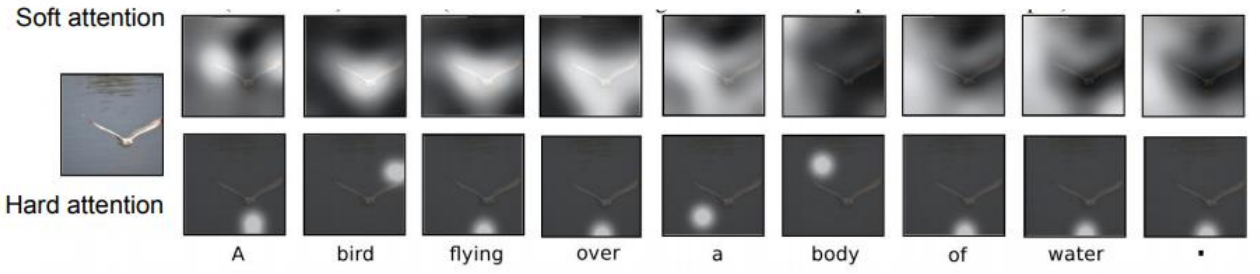
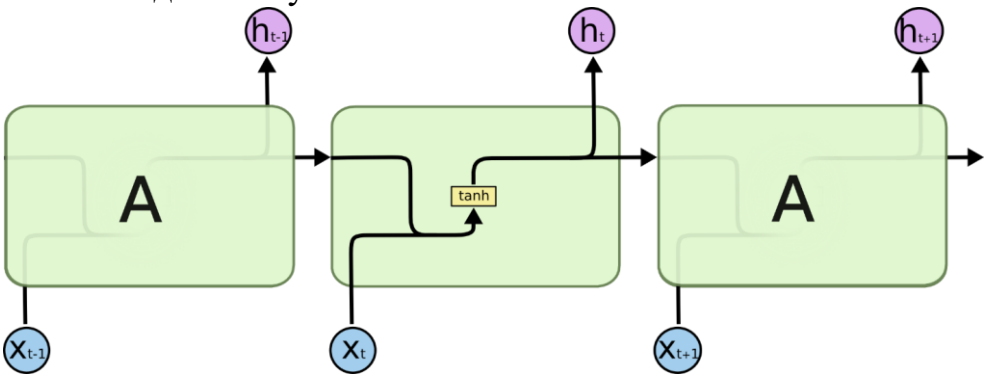


Рисунок 7.7 – Етапи створення опису птаха, що летить над водою

LSTM-мережі (Long short-term memory)

LSTM-мережа – це спеціальний тип РНС, здатний вчитися довгостроковим залежностям. LSTM-мережі були представлені в роботі Hochreiter and Schmidhuber, 1997, а потім оптимізовані та популяризовані у багатьох подальших роботах. Такі мережі чудово справляються з вирішенням багатьох завдань і знаходять широке застосування зараз.

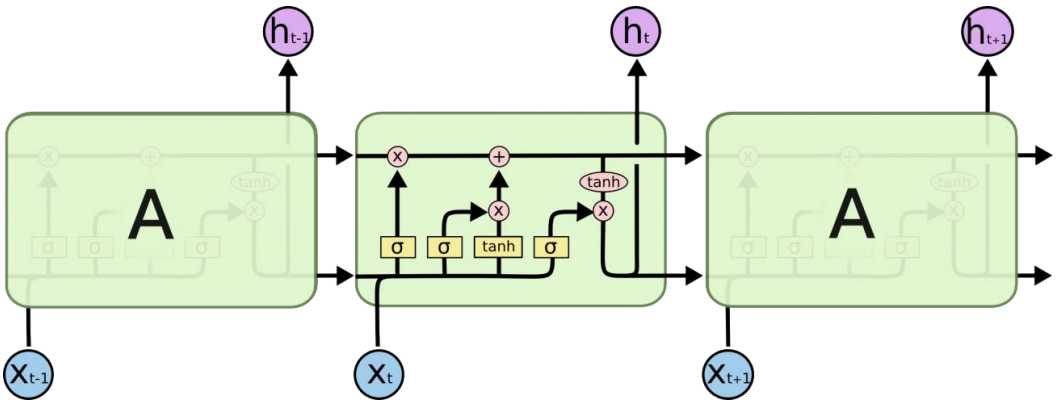
LSTM-мережі розроблені спеціально для того, щоб вирішити проблему довготривалих залежностей. Зберігання інформації протягом тривалих періодів часу – це їхня поведінка за умовчанням.



Модуль стандартної РНС, що повторюється, містить один шар.

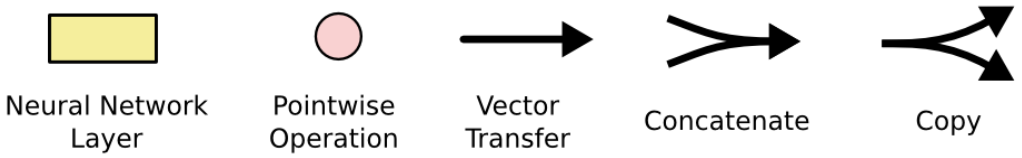
Всі РНС мають форму ланцюжка модулів, що повторюються. Модуль стандартної РНС, що повторюється, має дуже просту структуру, наприклад, єдиний tanh-шар (функція активації - гіперболічний тангенс).

LSTM-мережа є аналогічним ланцюжком, але повторюваний модуль має іншу структуру. Замість одного шару він містить чотири шари, які взаємодіють особливим чином.



Модуль LSTM-мережі, що повторюється, містить чотири взаємодіючі шари

Не хвилюйтеся, ми докладно розберемо представлену схему пізніше. А поки що давайте просто освоїмося з умовними позначеннями, які ми використовуватимемо.



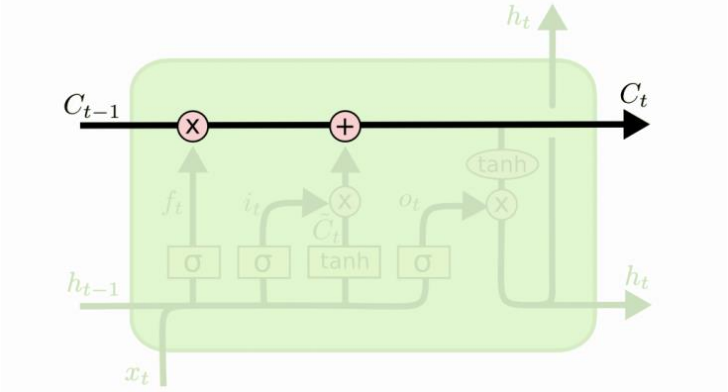
Шар нейронної мережі
Поточкова операція
Передача вектора
Об’єднання
Копіювання

На представленій схемі LSTM-мережі кожна лінія позначає передачу вектора з виходу одного вузла на входи інших. Рожеві кружки позначають крапкові операції, наприклад, складання векторів, а жовті прямокутники - навчені шари. Злиття ліній передбачає об'єднання, а розгалуження лінії свідчить, що інформація копіюється, і копії надсилаються різні точки призначення.

Ідея, що лежить в основі LSTM-мереж

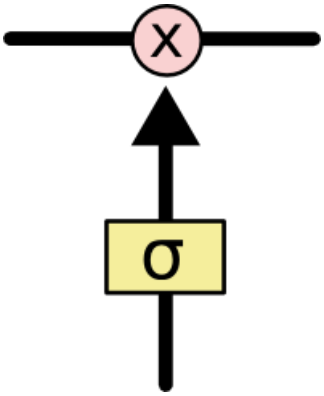
Ключовою особливістю мережі LSTM є стан комірки (cell state), представлений горизонтальною лінією у верхній частині наступного малюнка.

Стан комірки можна порівняти з конвеєрною стрічкою. Він проходить по всьому ланцюжку, лише з незначними лінійними взаємодіями.



LSTM-мережа має можливість видаляти та додавати інформацію у стан комірки. Цей процес регулюється спеціальними структурами, які називаються гейтами (gate).

Гейт – це механізм, що дозволяє пропускати інформацію вибірково. Він складається з sigmoid-шару (функція активації – сигмоїда) та операції поточкового множення.



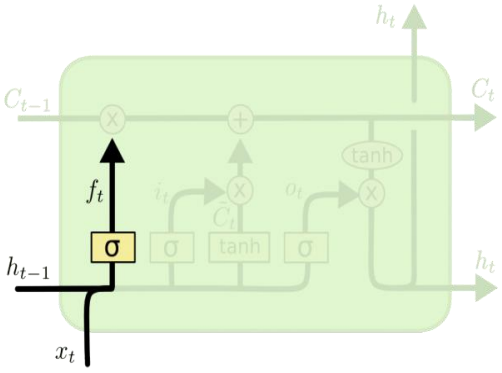
Виходом sigmoid-шару є число від 0 до 1, яке визначає рівень пропускання. Нуль означає "не пропустити нічого", а одиниця - "пропустити все".

Комірка має три гейти, що керують її станом.

Поетапне пояснення процесів, що відбуваються в LSTM-мережі

На першому етапі необхідно вирішити, яку інформацію слід видалити зі стану комірки. Це рішення приймає sigmoid-шар, що називається «гейтом забування» (forget gate). Він приймає на вході h_{t-1} та xt і дає на виході число від 0 до 1 для кожного значення у стані комірки C_{t-1}. Одиниця означає «повністю зберегти», а нуль – «цілком видалити».

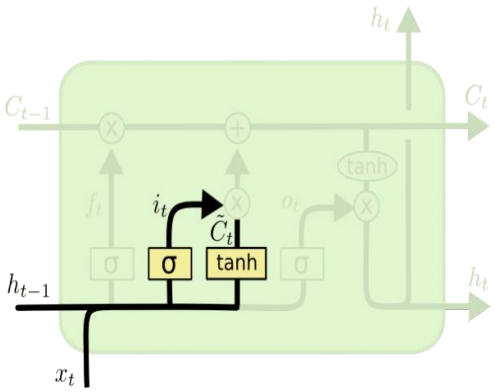
Повернімося до нашої мовної моделі, яка намагається передбачити наступне слово на підставі попередніх слів. У разі стан комірки може зберігати стать суб’єкта, про який йдеться, що дозволить використовувати правильні займенники. Коли з’являється новий суб'єкт, необхідно «забути» стать попереднього.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

На наступному етапі необхідно вирішити, яку нову інформацію слід записати у стан комірки. Цей етап поділяється на дві частини. Спочатку sigmoid-шар, званий вхідним гейтом (input gate), вирішує, які значення необхідно оновити. Потім tanh-шар створює вектор нових значень-кандидатів C_t, які можуть бути додані в стан комірки.

У разі мовної моделі ми хочемо записати у стан комірки стать нового суб’єкта замість статі попереднього суб’єкта, яку необхідно «забути».



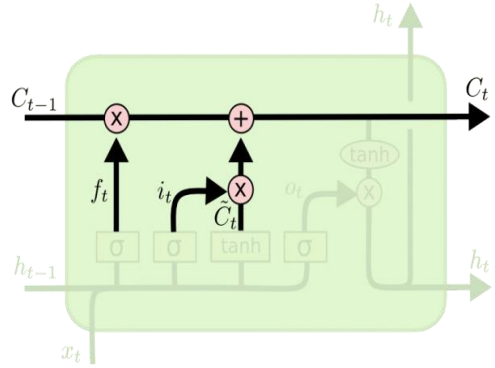
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Настав час оновити попередній стан комірки C_{t-1} до поточного стану C_t. На попередніх етапах вже було вирішено, що потрібно зробити, тепер залишилося лише виконати це.

Ми множимо попередній стан комірки на f_t, «забуваючи» таким чином те, що раніше було вирішено «забути». Потім додаємо i_t * C_t' – нові значення-кандидати, відмасштабовані відповідним чином.

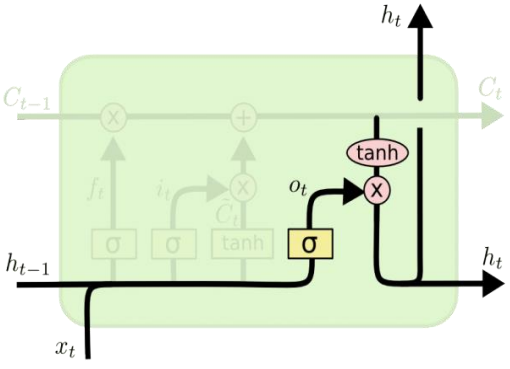
У разі мовної моделі на цьому етапі ми безпосередньо видаляємо інформацію про поле попереднього суб'єкта та додаємо нову інформацію згідно з рішеннями, прийнятими на попередніх етапах.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Нарешті необхідно вирішити, що слід відправити на вихід. Виходом буде відфільтрований стан комірки. Спочатку sigmoid-шар вирішує, які елементи стану комірки необхідно передати на вихід. Потім стан комірки перетворюється за допомогою tanh-шару до інтервалу від -1 до 1 і множиться на вихід sigmoid-шару, щоб вивести тільки те, що було вирішено вивести.

У разі мовної моделі, оскільки в її полі зору щойно з'явився новий суб'єкт, логічно вивести, наприклад, інформацію пов'язану з дієсловом, якщо наступним словом має бути дієслово. Зокрема, вона могла б вивести граматичне число, що б визначити правильну форму дієслова.



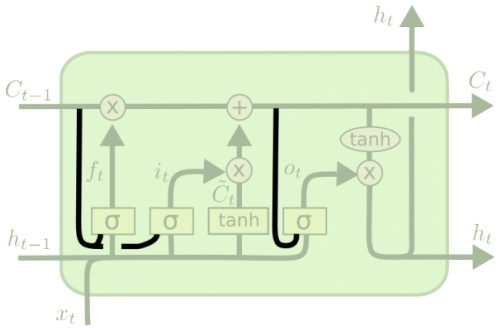
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Модифікації LSTM-архітектури

Ми розглянули стандартну LSTM-архітектуру. Крім неї також є різні модифікації. Насправді, майже у всіх роботах, що стосуються теми LSTM-мереж, застосовуються версії з певними відмінностями. Розглянемо деякі з них.

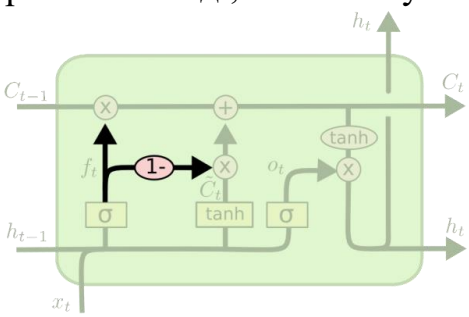
Один з популярних варіантів LSTM-мережі, представлений у роботі Gers and Schmidhuber, 2000, містить спеціальні зв'язки, які реалізують свого роду оглядові отвори. Завдяки цьому гейти отримують можливість "бачити" стан комірки.



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

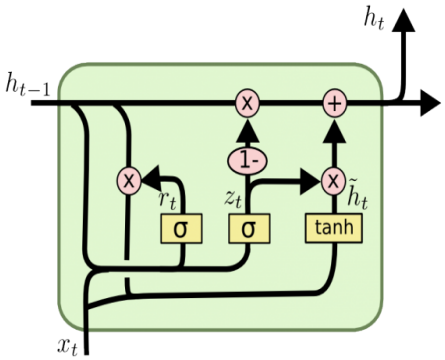
На представленій вище схемі такі зв'язки мають усі гейти, але в багатьох роботах вони мають тільки деякі.

В іншому варіанті гейт забування та вхідний гейт об'єднуються. Замість того, щоб окремо приймати рішення про видалення та додавання інформації, ми приймаємо ці рішення спільно. Тобто ми «забуваємо» щось, тільки тоді, коли збираємося замінити його чимось іншим. Відповідно, ми записуємо щось нове у стан комірки лише тоді, коли «забуваємо» щось старе.



$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

Ще однією варіацією на тему LSTM-мереж є архітектура на основі GRU (gated recurrent unit, керований рекурентний нейрон),. У цьому варіанті гейт забування та вхідний гейт об'днані в один «гейт оновлення» (update gate). Крім того, об'єднані разом стан комірки та прихований стан, а також присутні інші зміни. Отримана в результаті модель є більш простою, ніж стандартні LSTM-моделі, і, як наслідок, останнім часом набирає все більшої популярності.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Ми розповіли лише про деякі найцікавіші модифікації LSTM-архітектури. Існує безліч інших варіантів, наприклад, depth-gated RNN (PHC з керуванням по глибині). Також були запропоновані та принципово інші підходи до вирішення проблеми довготривалих залежностей, наприклад, clockwork RNN.

Стиснення даних (encoders)

На відміну від традиційних методів стиснення – математичного обчислення і видалення надмірності – нейронна мережа при вирішенні задачі стиснення виходить з міркувань нестачі ресурсів. Топологія мережі і її алгоритм навчання такі, що дані великої розмірності потрібно передати з входу нейронної мережі на її виходи через порівняно невеликих розмірів канал. Для реалізації стиснення такого роду може використовуватися багатошаровий перцептрон наступної архітектури: кількість нейронів у вхідному і вихідному шарі однаково одно розмірності стискаються даних; між цими шарами розташовуються один або більше проміжних шарів меншого розміру рис.7.8.

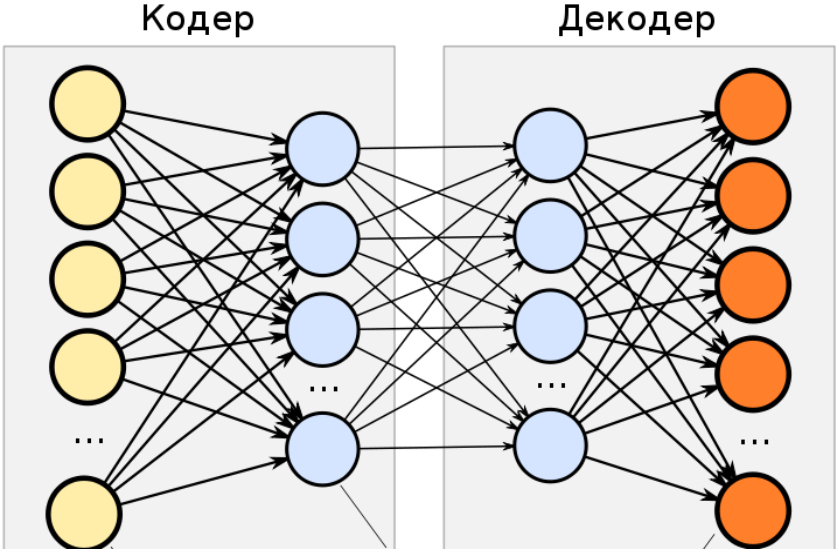


Рисунок 7.8 – Структура мережі для стиснення даних

8. ЕВОЛЮЦІЙНІ ОБЧИСЛЕННЯ

Серед безлічі проблем, які постають перед дослідниками як у галузі теорії, так і в численних практичних додатках значну частку становлять так звані оптимізаційні проблеми. Поняття оптимальності, мабуть, знайоме майже кожному й увійшло в практику більшості предметних галузей. З оптимізаційною проблемою ми стикаємося кожного разу, коли виникає необхідність вибору з деякої множини можливих рішень найкращого за певними критеріями і, як правило, задовольняє заданим умовам та обмеженням. Саме поняття оптимальності отримує цілком чітке тлумачення у математичних теоріях, проте у інших галузях може інтерпретуватися швидше змістовно.

Існують такі класи оптимізаційних задач (так звані NP-повні задачі), вирішення яких неможливо знайти без повного перебору варіантів. Відомо, що з великої розмірності цих задач реалізація перебору варіантів практично неможлива через надзвичайно великі часові витрати. У цій ситуації альтернативним походом до вирішення згаданих задач є застосування методів, що базуються на методології еволюційних обчислень.

Цей термін зазвичай використовується для загального опису алгоритмів пошуку, оптимізації або навчання, що базуються на деяких формалізованих принципах природного еволюційного відбору. Особливості ідей еволюції та самоорганізації полягають у тому, що вони є плідними та корисними не тільки для біологічних систем, що постійно підтверджується. Ці ідеї нині успішно використовуються для розробки багатьох технічних і, особливо, програмних систем.

8.1. Еволюційні теорії

Розробники алгоритмів еволюційних обчислень надихаються відомими еволюційними теоріями.

Модель еволюції Ч. Дарвіна (Дарвінізм)

З моделі Дарвіна розробники беруть положення про природний відбір, що «виживає найсильніший». Механізм природного відбору це – процес, за допомогою якого індивідіви деякої популяції, що мають більш високе функціональне значення (з сильними ознаками), отримують більшу можливість для відтворення нащадків, ніж «слабкі» індивідіви.

Модель еволюції Ж. Ламарка (Ламаркізм)

Заснована на припущенні, що характеристики, придбані індивідівою (організмом) протягом життя, успадковуються його нащадками. Модель є найбільш ефективною, коли популяція має тенденцію збіжності в точку локального оптимуму.

Модель еволюції де Фріза (Сальтаціонізм)

В основі моделі лежить моделювання соціальних і географічних катастроф, що призводять до різкої зміни видів і популяцій. Еволюція, таким чином, є послідовність стрибків у розвитку популяції без попереднього накопичення кількісних змін в еволюційних процесах.

Модель К. Поппера

Еволюція розглядається як розвиток ієрархічної системи гнучких

механізмів управління, в яких мутація інтерпретується як метод випадкових проб і помилок, а відбір – як один із способів управління за допомогою усунення помилок при взаємодії з зовнішнім середовищем.

8.2. Класифікація еволюційних обчислень

Еволюційні обчислення – це розділ еволюційного моделювання, який використовує та моделює процеси природного відбору. Простіше кажучи, це комп'ютерні програми, які намагаються вирішити складні проблеми, імітуючи процеси еволюції природи.

Іншими словами, це еволюційна теорія на мові ІТ. Завдяки еволюційній теорії Дарвіна ми знаємо, що спадковість, мінливість та природний відбір у природі призводять до невпорядкованої, спонтанної появи нових рішень проблеми виживання та розмноження. Отже, можна спробувати вирішувати існуючі обчислювальні завдання не класичним програмуванням, а тими ж непередбачуваними та випадковими принципами мінливості та відбору.

Еволюційні обчислення складають три великі напрями:

- генетичні алгоритми
- еволюційна стратегія
- еволюційне програмування

Генетичні алгоритми (ГА). Основна особливість – використання оператора рекомбінації (схрещення) в якості основного механізму пошуку. Це ґрунтується на припущенні, що частини оптимального розв'язку можуть бути знайдені незалежно та рекомбіновані для отримання кращого розв'язку.

Еволюційні стратегії (ЕС). Особливість – використання само-адаптивних механізмів для контролю процесу мутації. Ці механізми зосереджені на еволюції шуканих розв'язків й на еволюції параметрів мутації.

Еволюційне програмування (ЕП). Фокусується більше на адаптації індивідів, аніж на еволюції генетичної інформації. Зазвичай застосовує безстатеве розмноження та мутації, тобто, внесення невеликих змін в поточний розв'язок та методи селекції прямої конкуренції.

8.3. Генетичний алгоритм

Із досліджень в галузі генетики відомо, що вся інформація про організм закодована в молекулах ДНК, що знаходяться в ядрі клітини. Ці молекули називають хромосомами. ДНК складаються з послідовності генів (біологічних кодів). Кожен ген визначає окрему властивість організму. Новий організм при розмноженні успадковує половину хромосом від батька і половину від матері і, таким чином, він буде мати нові властивості, які для його виживання в середовищі можуть бути як позитивними, так і негативними. Після цього починають діяти закони еволюції і, в результаті залишаються тільки ті організми, які найбільш пристосовані до зовнішніх умов тобто «оптимальні». Крім цього окремий організм не здатний вижити в живій природі. Виживає разом якась мінімальна кількість організмів, або популяція.

Ідею генетичних алгоритмів висловив Holland J.N (Дж. Холланд) у 1975 році []. Він зацікавився властивостями процесів природної еволюції (в тому числі фактом, що еволюціонують хромосоми, а не самі живі істоти). Холланд був упевнений у можливості скласти і реалізувати у вигляді комп'ютерної програми алгоритм, який буде вирішувати складні задачі так, як це робить природа – шляхом еволюції.

Генетичні алгоритми відрізняються від традиційних методів оптимізації декількома базовими елементами:

1. обробляють не значення параметрів самої задачі, а їх закодовану форму;
2. здійснюють пошук рішення виходячи не з єдиної точки, а з їх деякої популяції;
3. використовують тільки цільову функцію, а не її похідні або іншу додаткову інформацію;
4. застосовують імовірнісні, а не детерміновані правила вибору.

Перераховані чотири властивості, які можна сформулювати також як кодування параметрів, операції на популяціях, використання мінімуму інформації про завдання і рандомізація операцій приводять у результаті до стійкості генетичних алгоритмів і до їх переваги над іншими широко вживаними технологіями.

Терміни генетичного алгоритму

При описі генетичних алгоритмів використовуються визначення, запозичені з генетики. Наприклад, мова йде про популяцію індивідів, а в якості базових понять застосовуються ген, хромосома, генотип, фенотип, алель. Також використовуються відповідні цим термінам визначення з технічного лексикону, зокрема, ланцюг, двійкова послідовність, структура.

Популяція – це кінцева множина індивідів.

Індивід, що входять в популяцію, у генетичних алгоритмах представляються хромосомами з закодованими в них множинами параметрів задачі, тобто рішень, які інакше називаються точками в просторі пошуку (search points). У деяких роботах індивідів називаються особинами, організмами.

Хромосоми (інші назви – ланцюги або кодові послідовності) – це впорядковані послідовності генів.

Ген (який також називається властивістю, знаком чи детектором) – це атомарний елемент генотипу, зокрема, хромосоми.

Генотип або структура – це набір хромосом окремого індивіду. Отже, індивідами популяції можуть бути генотипи або одиничні хромосоми (в досить поширеному випадку, коли генотип складається з однієї хромосоми).

Фенотип – це набір значень, які відповідає даному генотипу, тобто декодована структура або множина параметрів задачі (розв'язок задачі, точка простору пошуку).

Алель – це значення конкретного гена, також визначається як значення властивості або варіант властивості.

Локус або позиція вказує місце розміщення даного гена в хромосомі (ланцюгу). Множина позицій генів – це локи.

Функція пристосованості (**fitness function**) або функція оцінки. Вона оцінює міру пристосованості даного індивіду в популяції. Ця функція відіграє найважливішу роль, оскільки дозволяє оцінити ступінь пристосованості конкретних індивідів у популяції і вибрати з них згідно з еволюційним принципом виживання «найсильніших» найбільш пристосованих (тобто тих, що мають найбільші значення функції пристосованості).

Функція пристосованості також отримала свою назву безпосередньо із генетики. Вона має сильний вплив на функціонування генетичних алгоритмів і повинна мати точне і коректне визначення. У задачах оптимізації функція пристосованості, як правило, оптимізується (точніше кажучи, максимізується) і називається цільовою функцією. У задачах мінімізації максимізується від'ємна цільова функція, і проблема все одно зводиться до максимізації. У теорії управління функція пристосованості може приймати вигляд функції похибки, а в теорії ігор – вартісної функції.

На кожній ітерації генетичного алгоритму пристосованість кожного індивіда даної популяції оцінюється за допомогою функції пристосованості, і на цій основі створюється наступна популяція індивідів, що складають нову множину потенційних рішень проблеми оптимізації. Чергова популяція в генетичному алгоритмі називається поколінням, а до новостворюваної популяції індивідів застосовується термін «нове покоління» або «покоління нащадків».

Генетичні оператори

При пошуку рішень генетичні оператори моделюють головні процеси еволюції: спадковість, мінливість, природний відбір.

Оператор схрещування (crossover operator) – обмін генетичним матеріалом між індивідами (хромосомами) для створення нового індивіда. Оператор моделює мінливість і спадковість.

Оператор мутації (mutation operator)

Моделює випадкову зміну гену, що може призвести до стрибка у розвитку популяції. Для двійкових хромосом це інверсія випадково обраного біта.

На рис. 8.1 подані приклади оператора мутації і схрещування.

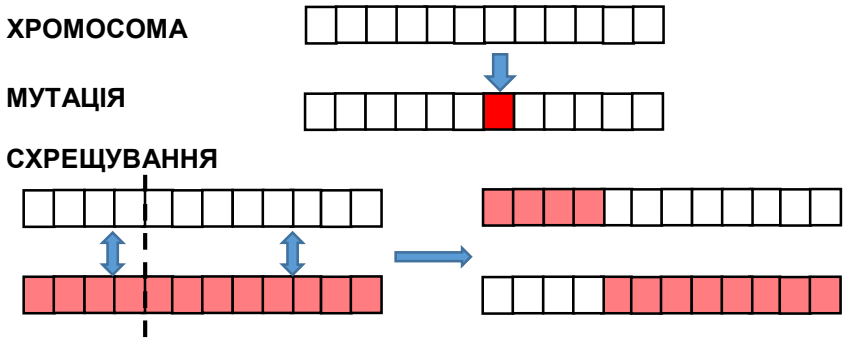


Рисунок 8.1 – Схема операторів мутації і схрещування

Оператор селекції – правило відбору хромосом для нового покоління. Моделює процеси природного відбору. Налічує декілька способів.

Схема пропорційної селекції:

$$\bar{F} = \frac{\sum F_i}{N} \qquad r_i = \frac{F_i}{\bar{F}}$$

Обирається хромосома з найкращою функцією пристосованості серед хромосом, що утворені оператором схрещування.

Схема рулетки (roulette-wheel selection):

$$a_i = 2\pi \frac{F_i}{\bar{F}}$$

Значення функції пристосованості індивідів популяції подається, як сектор кола, чим більше це значення, тим більше площа сектора.

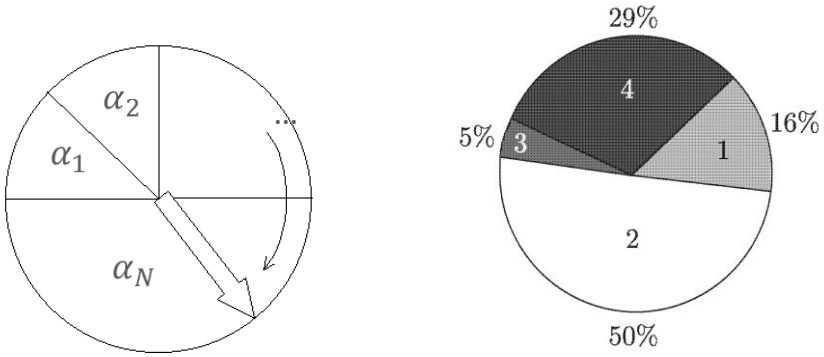


Рисунок 8.2 – Схема рулетки

Розіграш за допомогою колеса рулетки зводиться до випадкового вибору числа з інтервалу [0, 100], що вказує на відповідний сектор на колесі, тобто на конкретну хромосому. Таким чином моделюється відомий факт, що і від «слабких» батьків може народитись сильна дитина, яка потім прийме участь у подальшому відборі, тобто є невелика ймовірність, що й такі хромосоми будуть розглядатись, а не тільки найкращі з точки зору фітнес функції

Турнірна селекція

При турнірній селекції все індивіди популяції розбиваються на підгрупи з подальшим вибором в кожній з них індивідів з найкращою пристосованістю. Розрізняються два типи такого вибору: детермінований вибір і випадковий вибір. Детермінований вибір здійснюється з імовірністю, що дорівнює 1, а випадковий вибір – з ймовірністю, меншою 1. Підгрупи можуть мати довільний розмір, але найчастіше популяція поділяється на підгрупи по 2-3 індивіди в кожній.

Рангова селекція

При ранговій селекції індивіди популяції ранжуються за значенням їх функції пристосованості. Це можна уявити собі як відсортований список індивідів, упорядкованих у напрямку від найбільш пристосованих до найменш пристосованим (або навпаки), в якому кожному індивіду приписується ранг (число), що визначає його місце в списку. Кількість копій кожного індивіду, введених в батьківську популяцію, розраховується по апріорно заданій функції в залежності від рангу індивіду.

Елітарна стратегія (elitist strategy)

Полягає в захисті найкращих хромосом на наступних ітераціях. У класичному генетичному алгоритмі найбільш пристосовані індивіди не завжди переходять в наступне покоління. Це означає, що нова популяція не завжди містить хромосому з найбільшим значенням функції пристосованості з попередньої популяції. Елітарна стратегія застосовується для запобігання втрати такого індивіду. Цей індивіда гарантовано включається в нову популяцію.

Умовно елітизм можна розділити на два класи-підходи:

1. ***Конкурентний підхід –*** батьківські індивіду "змагаються" з нащадками і переможці (або переможець) переходять у наступну популяцію (покоління);

2. ***Неконкурентний підхід*** – частина батьківської підпопуляції (випадкова або визначена за правилом) переходить у нову популяцію без будь-яких заперечень з боку електорату.

8.4. Класичний генетичний алгоритм (Standard Genetic Algorithm, SGA)

Ідея генетичного алгоритму припускає, що діючи ітеративно, знаходячи все більш кращі хромосоми і популяції ми за декілька кроків поступово наблизимся до найкращого результату. Але ми можемо проскочити максимум, або він може не існувати, тому важливо вчасно зупинитись, щоб не потрапити в нескінченний цикл. Звісно найкращою умовою є досягнення заданого наперед значення функції пристосованості. Взагалі розглядаються такі критерії зупинки ГА:

- виконання заданої кількості ітерацій;
- виконання заданої кількості ітерацій без поліпшення;
- досягнення заданого значення функції пристосованості (fitness function).

Таким чином, (рис. 8.3) генетичний алгоритм складається з таких етапів:

1. Створення початкової популяції.
2. Обчислення функції пристосованості для індивідів популяції (оцінювання).
3. Повторювання до виконання критерію зупинки алгоритму.
4. Вибір індивідів із поточної популяції (селекція).
5. Схрещення і/або мутація.
6. Обчислення функції пристосовуваності для всіх індивідів.
7. Формування нового популяції.

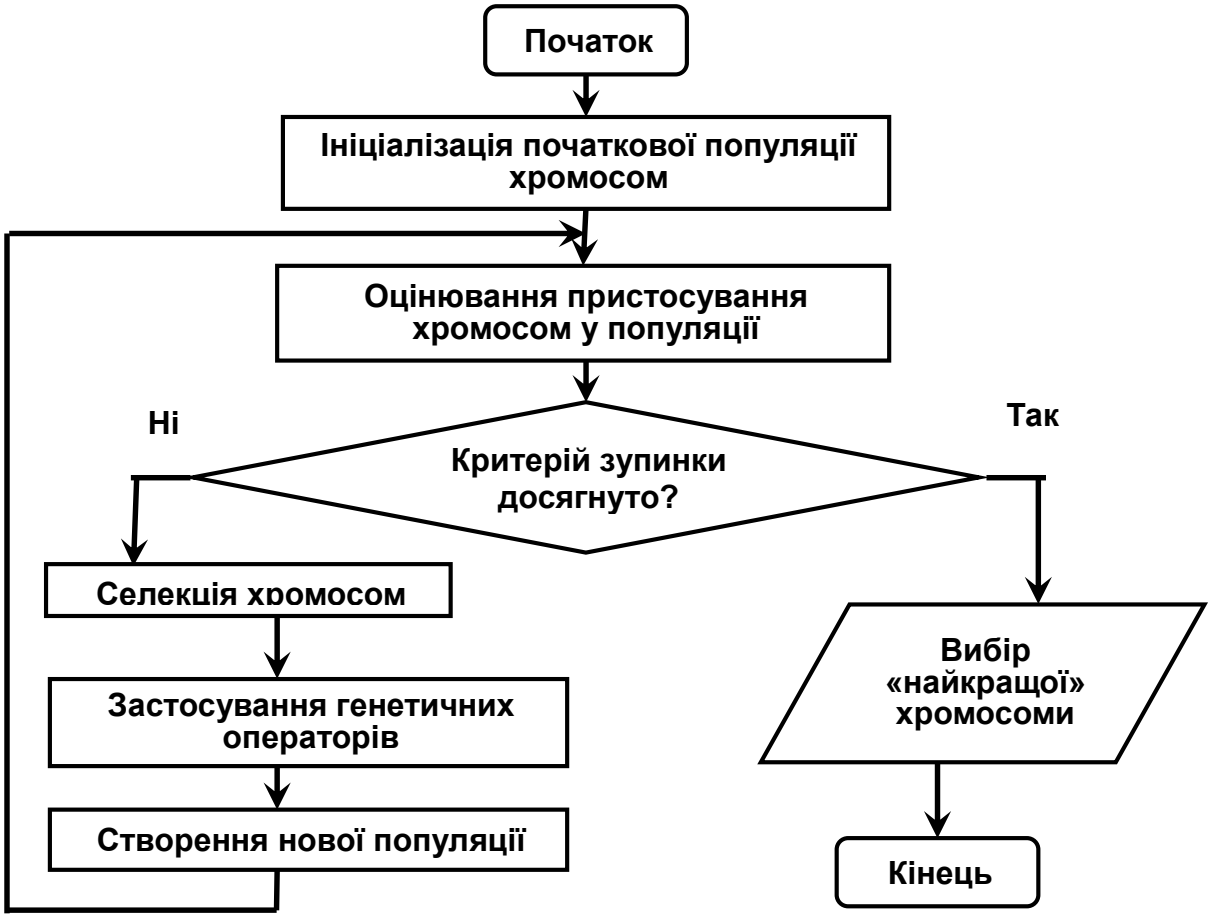


Рисунок 8.3 – Блок-схема генетичного алгоритму

Основна теорема про генетичні алгоритми

Для того щоб краще зрозуміти функціонування генетичного алгоритму, будемо використовувати поняття схеми і сформулюємо основну теорему, що відноситься до генетичних алгоритмів і називається **теоремою про схеми**. Поняття схема було введено Холландом і використовується для аналізу роботи ГА. Зокрема розглядаються процеси конструювання і руйнування визначеної схеми протягом розвитку популяції (schema dynamics).

Схемою називається рядок виду

$$(a_1, a_2, \ldots, a_i, \ldots, a_l),\ a_i \in \{0, 1, *\}$$

Символом "*" у деякому розряді позначається те, що там може бути як 1, так і 0.

Наприклад, для двох бінарних рядків

"111000111000" і

"110011001100" схема буде виглядати в такий спосіб:

"11*0****1*00". Тобто за допомогою схем можна як би виділяти загальні ділянки двійкових рядків і маскувати розходження. Маючи в складі схем m символів "*" можна закодувати (узагальнити) 2^m двійкових рядків. Так, наприклад, схема "01*0*1" описує набір рядків

{"010001", "010011", "011001", "011011"}.

Генетичний алгоритм базується на принципі трансформації найбільш пристосованих індивідів (хромосом). Нехай *P(0)* означає вихідну популяцію індивідів, а *P(k)* – поточну популяцію (на *k*-й ітерації алгоритму). З кожної популяції *P(k)*, *k* = 0, 1, ... методом селекції вибираються хромосоми з найбільшою пристосованістю, які включаються в так званий батьківський пул (mating pool) *M(k)*. Далі, у результаті об'єднання індивідів з популяції *M(k)* у батьківські пари і виконання операції схрещування з ймовірністю p_c, а також операції мутації з ймовірністю $p_т$ формується нова популяція *P(k+1)*, у яку входять нащадки індивідів з популяції *M(k)*. Отже, для будь-якої схеми, що представляє гарне рішення, було б бажаним, щоб кількість хромосом, що відповідають цій схемі, зростало зі збільшенням кількості ітерацій *k*.

На відповідне перетворення схем у генетичному алгоритмі впливають 3 фактори: *селекція хромосом, схрещування і мутація*. Проаналізуємо вплив кожного з них з погляду збільшення очікуваної кількості представників окремо узятої схеми. Позначимо розглянуту схему символом S, а кількість хромосом популяції P(k), що відповідають цій схемі – *c(S, k)*.

Отже, *c(S, k)* можна вважати кількістю елементів (тобто потужністю) множини *P(k)∩S*.

Почнемо з *дослідження впливу селекції*. При виконанні цієї операції хромосоми з популяції P(k) копіюються в батьківський пул M(k) з ймовірністю p_c. Нехай F(S, k) позначає середнє значення функції пристосованості хромосом з популяції P(k), що відповідають схемі S. Якщо

$$P(k)S=\{ch_1 \ldots ch_{c(S,k)}\}$$

то

$$c(S,k)=\frac{\sum_{i=1}^{c(S,k)} F(ch_i)}{F(S,k)}$$

Величина F(S, k) також називається пристосованістю схеми S на k-й ітерації. Очікувана кількість хромосом з множини P(k)∩ S, відібраних для включення в батьківський пул M(k), буде дорівнювати

$$\frac{\sum_{i=1}^{c(S,k)} F(ch_i)}{\bar{F}(k)}=c(S,k)\frac{F(S,k)}{\bar{F}(k)}$$

де

$\bar{F}(k)$ – середнє значення функції пристосованості хромосом цієї популяції.

Оскільки кожна хромосома з батьківського пулу M(k) одночасно належить популяції P(k), то хромосоми, що складають множину M(k)∩S - це ті ж самі індивіди, що були відібрані з множини P(k)∩S для включення в популяцію M(k). Якщо кількість хромосом батьківського пулу M(k), що відповідають схемі S (тобто кількість елементів множини M(k)∩S), позначити b(S, k), то з приведених міркувань можна зробити наступний висновок:

Висновок 3.1

Очікуване значення b(S, k), тобто очікуване значення кількості хромосом

батьківського пулу M(k), що відповідають схемі S, визначається виразом

$$E[b(S,k)] = c(S,k)\frac{F(S,k)}{\bar{F}(k)}$$

З цього випливає, що якщо схема S містить хромосоми зі значенням функції пристосованості, що перевищує середнє значення (тобто пристосованість схеми S на k-й ітерації виявляється більшою, ніж середнє значення функції пристосованості хромосом з популяції P(k), і тому F(S,k)/F(k)>1, то очікувана кількість хромосом з батьківського пулу M(k), що відповідають схемі S, буде більше кількості хромосом з популяції P(k), що відповідають схемі S. Тому можна стверджувати, що селекція викликає поширення схем із пристосованістю «краще середньої» і зникнення схем з «гіршою» пристосованістю.

Перш ніж приступити до аналізу впливу генетичних операторів схрещування і мутації на хромосоми з батьківського пулу, визначимо необхідні для подальших міркувань поняття *порядку й охоплення схеми*. Нехай L позначає довжину хромосом, що відповідають схемі S.

Означення 3.1

Порядок (order) схеми S, інакше називаний рахунковістю схеми і що позначається o(S) – це кількість сталих позицій у схемі, тобто нулів і одиниць у випадку алфавіту {0, 1, *}.

Наприклад,

o(10*1) = 3 o(*01*10) = 4 o(**0*Г) = 2 o(*101**) = 3

Порядок схеми o(S) дорівнює довжині L за винятком кількості символів *, що легко перевірити на представлених прикладах (для L = 4 з одним символом * і для L = 6 із двома, чотирма і трьома символами *). Легко помітити, що порядок схеми, що складається винятково із символів *, дорівнює нулеві, тобто o(****) = 0, а порядок схеми без єдиного символу * дорівнює L; наприклад, o(10011010) = 8. Порядок схеми o(S) - це завжди ціле число з інтервалу [0, L].

Означення 3.2

Охоплення (defining length) схеми S, яке називається також довжиною схеми (не плутати з довжиною L) і що позначається d(S) - це відстань між першим і останнім сталим символом (тобто різниця між правими і лівої крайніми позиціями, що містять сталі символи). Наприклад,

d(10*1) = 4-1 =3 d(**0*1*) = 5-3 = 2 d(*01*10) = 6-2 = 4 d(*101**) = 4-2 = 2

Охоплення схеми d(S) - це ціле число з інтервалу [0, L - 1]. Відзначимо, що охоплення схеми зі сталими символами на першій і останній позиції дорівнює L-1 (як у першому з приведених прикладів). Охоплення схеми з єдиною сталою позицією дорівнює нулеві, зокрема, d(**1*)=0. Охоплення характеризує змістовність інформації, яка міститься в схемі.

Перейдемо до міркувань про вплив операції схрещування на обробку схем у генетичному алгоритмі. Насамперед відзначимо, що одні схеми виявляються більш підданими знищенню в процесі схрещування, ніж інші. Наприклад, розглянемо схеми

S1 = 1****0* і S2 = **01***, а також хромосому ch=[1001101], що відповідає обом схемам. Видно, що схема S2 має більше шансів «пережити» операцію схрещування, ніж схема S1, що більше піддана «розщепленню» у точках схрещування 1, 2, 3, 4 і 5. Схему S2 можна розділити тільки при виборі точки схрещування, рівної 3. Звертаємо увагу на охоплення обох схем, що - це очевидно - виявляється істотним у процесі схрещування.

У ході аналізу впливу операції схрещування на батьківський пул M(k) розглянемо деяку хромосому з множини M(k)∩S, тобто хромосому з батьківського пулу, що відповідає схемі S.

Ймовірність того, що ця хромосома буде відібрана для схрещування, дорівнює p_c. Якщо жоден з нащадків цієї хромосоми не буде належати до схеми S, то це означає, що точка схрещування повинна знаходитися між першим і останнім сталим символом даної схеми. Ймовірність цього дорівнює d(S)/(L-1). З цього можна зробити наступні висновки:

Висновок 3.2 (вплив схрещування)

Для деякої хромосоми з M(k)∩S ймовірність того, що вона буде відібрана для схрещування і жоден з її нащадків не буде належати до схеми S, обмежена зверху величиною p_c

$$p_c \frac{d(S)}{L-1}$$

Ця величина називається *ймовірністю знищення схеми S.*

Висновок 3.3

Для деякої хромосоми з M(k)∩S ймовірність того, що вона не буде відібрана для схрещування або, що хоча б один з її нащадків після схрещування буде належати до схеми S, обмежена знизу величиною

$$1- p_c \frac{d(S)}{L-1}$$

Ця величина називається *ймовірністю виживання схеми S.*

Легко показати, що якщо дана хромосома належить до схеми S і відбирається для схрещування, а друга батьківська хромосома також належить до схеми S, то обидва їх нащадка теж будуть належати до схеми S. Висновки 3.2 і 3.3 підтверджують значимість показника охоплення схеми d(S) для оцінки ймовірності знищення або виживання схеми.

Розглянемо тепер *вплив мутації на батьківський пул* M(k). Оператор мутації з ймовірністю p_m випадковим чином змінює значення в конкретній позиції з 0 на 1 і зворотно. Очевидно, що схема переживе мутацію тільки в тому випадку, коли всі її сталі позиції залишаться після виконання цієї операції незмінними.

Хромосома з батьківського пулу, що належить до схеми S (тобто хромосома з множини M(k)∩S) залишиться в цій схемі тоді і тільки тоді, коли жоден символ цієї хромосоми, що відповідає сталим символам схеми S, не зміниться в процесі мутації.

Ймовірність такої події дорівнює $(1-p_m)^{o(S)}$.

Цей результат можна представити у формі наступного висновку:

Висновок 3.4 (вплив мутації)

Ймовірність того, що деяка хромосома з M(k)∩S буде належати до схеми S після операції мутації, визначається виразом

$$(1-p_m)^{o(S)}$$

Ця величина називається **ймовірністю виживання схеми S після мутації**.

Висновок 3.5

Якщо ймовірність мутації p_m мала ($p_m << 1$), то можна вважати, що ймовірність виживання схеми S після мутації, яка визначена у висновку 3.4, приблизно дорівнює $1 - p_m o(S)$

Ефект спільного впливу селекції, схрещування і мутації (висновки 3.1 - 3.4) з урахуванням факту, що якщо хромосома з множини M(k)∩S дає нащадка, що відповідає схемі S, то він буде належати до P(k+1)∩S, веде до побудови наступної схеми репродукції:

$$E[c(S,k+1)] \geq c(S,k)\frac{F(S,k)}{\bar{F}(k)}\left(1-p_c\frac{d(S)}{L-1}(1-p_m)^{o(S)}\right)$$

Ця залежність показує, як змінюється від популяції до популяції кількість хромосом, що відповідають даній схемі. Ця зміна викликається трьома факторами, представленими в правій частині виразу, зокрема: F(S,k)/F(k) відбиває роль середнього значення функції пристосованості, $1 - p_c d(S)/(L-1)$ показує вплив схрещування і $(1 - p_m)^{o(S)}$ – вплив мутації. Чим більше значення кожного з цих факторів, тим більшим виявляється очікувана кількість відповідностей схемі S у наступній популяції.

Теорема

Схеми малого порядку, з малим охопленням і з пристосованістю вище середньої формують показниково зростаючу кількість своїх представників у наступних поколіннях генетичного алгоритму.

Відповідно до приведеної теореми важливим питанням стає кодування, що повинне забезпечувати побудову схем малого порядку, з малим охопленням і з пристосованістю вище середньої. Непрямим результатом теореми (про схеми) можна вважати наступну гіпотезу, названу *гіпотезою про цеглинки (або про будівельні блоки)*.

Гіпотеза

Генетичний алгоритм прагне досягти близького до оптимального результату за рахунок комбінування гарних схем (із пристосованістю вище середньої) малого порядку і малого охоплення. Такі схеми називаються *цеглинками (або будівельними блоками)*.

Гіпотеза про будівельні блоки висунута на підставі теореми про схеми з урахуванням того, що генетичні алгоритми досліджують простір пошуку за допомогою схем малого порядку і малого охоплення, які згодом беруть участь в обміні інформацією при схрещуванні. Незважаючи на те, що для доведення цієї гіпотези виконувалися значні дослідження, однак у більшості нетривіальних доданків приходиться спиратися на емпіричні результати. Протягом останніх років опубліковано численні роботи, присвячені застосуванням генетичних алгоритмів, що підтверджують цю гіпотезу. Якщо вона вважається правильною, то проблема кодування здобуває критичне значення для генетичного алгоритму;

кодування повинне реалізувати концепцію малих будівельних блоків. Якість, яка забезпечує генетичним алгоритмам явну перевагу перед іншими традиційними методами, безсумнівно полягає в обробці великої кількості різних схем.

Будівельний блок (ББ) (building block (BB)) – це одне з ключових понять у теорії генетичних алгоритмів, поряд зі схемою і теоремою схем. До будівельних блоків (як правило) прийнято відносити схеми з малою визначальною довжиною, малим порядком і високою пристосованістю. Пристосованість ББ найчастіше визначається як середня пристосованість особин, хромосоми яких містять даний будівельний блок. У гіпотезі про будівельні блоки (building blocks hypothesis) вважається, що в процесі роботи генетичного алгоритму, у міру наближення до глобального оптимуму, порядок і пристосованість будівельного блоку збільшуються. Тобто на початку еволюції відносно високою пристосованістю володіють будівельні блоки малої визначальної довжини, потім, у результаті добору і рекомбінації, з'являються більш пристосовані і "довгі" будівельні блоки. Таким чином, говорять про обробку будівельних блоків (building blocks processing) у результаті роботи генетичного алгоритму. Виділяють наступні проблеми в обробці будівельних блоків:

1. Ідентифікація.
2. Змішування.

Під ***ідентифікацією*** розуміють проблему знаходження (локалізації) будівельного блоку. Іншими словами, яку саме ділянку хромосоми можна вважати будівельним блоком.

Очевидно, що мінімальний по розмірах ББ представляє один розряд, а максимальний усю хромосому. Але це в найпростішому варіанті, а у випадку "не граничних умов" кількість "претендентів" на звання будівельного блоку дуже велика. Зокрема, для хромосоми довжиною n може існувати C(n, 2) різних позицій для будівельних блоків 2-го порядку, C(n, 3) різних позицій для будівельних блоків третього порядку і т.д., де C(n, m) - кількість сполучень з n по m (обчислюється як n!/(m!(n-m)!), де "!" - факторіал, тобто n! = 1*2*...*(n-1)*n, де "*" - позначає операцію множення).

Проблема ***змішування*** полягає в тому, що, навіть якщо будівельні блоки знайдені, важко визначити, які з них варто змішувати (грубо говорячи, поміщати в одну хромосому), а які – ні, так як висока пристосованість різних будівельних блоків не гарантує високої пристосованості хромосом, отриманих у результаті їхньої комбінації.

У силу позначених проблем приділяється досить багато уваги розробці операторів і підходів, що дозволили б забезпечити ефективну (BB-wise) обробку будівельних блоків. Також існує думка, що ефективна ідентифікація і змішування є критичними умовами для забезпечення успішної роботи генетичного алгоритму.

8.5 Типи генетичних алгоритмів

Існують різні типи генетичного алгоритму (класичний, простий генетичний алгоритм, гібридний, CHC генетичний алгоритм та ін.) Вони

розрізняються по стратегіям відбору та формування нового покоління індивідів, операторами генетичного алгоритму, кодуванням генів і т.і.

CHC-алгоритм

CHC (Cross generational elitist selection, Heterogenous recombination, Cataclysmic mutation) був запропонований Есхелманом і характеризується наступними параметрами:

1. Для нового покоління вибираються N кращих різних індивідів серед батьків і дітей. Дублювання рядків не допускається.
2. Для схрещування вибирається випадкова пара, але не допускається, щоб між батьками була малою Хеммінгова відстань або малою відстань між крайніми бітами.
3. Для схрещування використовується різновид однорідного кросовера HUX (Half Uniform Crossover): дитині переходить рівно половина бітів кожного з батьків.
4. Розмір популяції невеликий, близько 50 індивідів. Цим виправдано використання однорідного кросовера.

Genitor

Цей алгоритм був створений Д. Уитли. Genitor-подібні алгоритми відрізняються від класичного ГА наступними трьома властивостями:

1. На кожному кроці тільки одна пара випадкових батьків створює тільки одного нащадка.
2. Цей нащадок замінює не батьків, а одного з найгірших індивідів популяції (в первісному Genitor – найгіршого).
3. Відбір індивідів для заміни проводиться по їх рангу (рейтингу), а не по пристосованості.

У Genitor пошук гіперплощин відбувається краще, а збіжність швидше, ніж у класичному генетичного алгоритму, запропонованого Холландом.

Гібридні алгоритми

Ідея гібридних алгоритмів (hybrid algorithms) полягає в поєднанні генетичного алгоритму з деяким іншим методом пошуку. В кожному поколінні кожен отриманий нащадок оптимізується цим методом, після чого здійснюються звичайні для генетичного алгоритму дії.

Такий тип пошуку називається Ламарковою еволюцією, при якій індивід здатен навчатися, а потім отримані навички записувати в свій генотип, щоб передати їх нащадкам. І хоча такий метод погіршує здатність алгоритму шукати рішення за допомогою відбору гіперплощин, однак на практиці гібридні алгоритми виявляються дуже вдалими. Це пов’язано з тим, що зазвичай велика ймовірність того, що один з індивідів потрапить в оласть глобального максимуму і після оптимізації виявиться рішенням задачі.

Паралельні генетичні алгоритми

Генетичні алгоритми можна організувати як кілька паралельно виконуються процесів, це збільшує їх продуктивність. Розглянемо перехід від класичного генетичного алгоритму до паралельного. Для цього будемо використовувати турнірний відбір. Заведемо N/2 процесів (тут і далі процес мається на увазі як деяка машина, процесор, який може працювати незалежно).

Кожен з них буде вибирати випадково з популяції 4 індивіду, проводити 2 турніру і схрещувати переможців. Отримані діти будуть записуватися в нове покоління. Таким чином, за один цикл роботи одного процесу буде змінюватися ціле покоління.

8.6 Застосування генетичних алгоритмів:

Задача комівояжера (Travelling Salesman Problem, TSP)

Задача комівояжера є класичною оптимізаційною задачею, на якій можна оцінювати нові алгоритми оптимізації. Суть її полягає в наступному. Дано множину з m міст і матриця відстаней між ними або вартостей переїзду (залежно від інтерпретації). Мета комівояжера – об'їхати всі ці міста по найкоротшому шляху або з найменшими витратами на поїздку. Причому в кожному місті він повинен побувати один раз і свій шлях закінчити в тому ж місті, звідки почав. Відстань між містами відома, шляхи між містами орієнтовані (рис. 8.4).

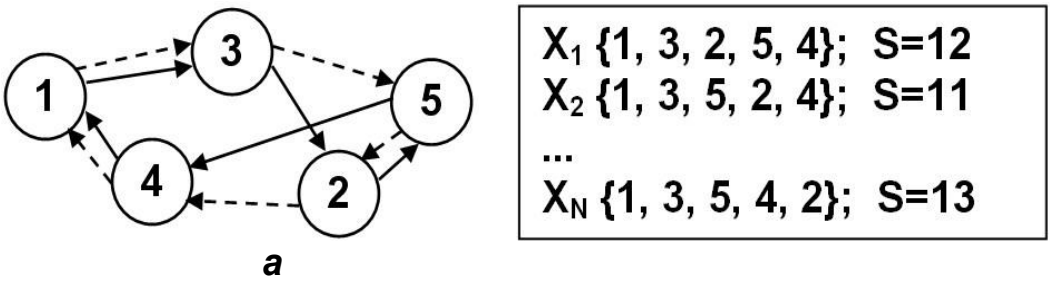


Рисунок 8.4 – Вирішення задачі комівояжера за допомогою ГА:
а) – фенотип (маршрути); *б*) – генотип (набір хромосом популяції);
S – довжина шляху (хромосома).

Кожен маршрут (фенотип) описується рядком з *M* десяткових чисел, тобто хромосома *X* індивіда є впорядкованою послідовністю з *M* генів, де на *j*-му місці стоїть номер *j*-го за порядком обходу міста. Якщо на початку розв'язання задачі комівояжера ми маємо випадковий набір маршрутів (індивідів), то в процесі еволюції виживуть тільки маршрути з мінімальною довжиною (кращим набором генів).

В загальному випадку ГА застосовуються для таких задач.

1. Оптимізація функцій.
2. Оптимізація запитів в базах даних.
3. Різноманітні задачі на графах (задача комівояжера, розфарбування).
4. Налаштування і навчання штучної нейронної мережі.
5. Задачі компоновки.
6. Створення розкладів.
7. Ігрові стратегії.
8. Апроксимація функцій
9. Штучне життя
10. Біоінформатика (згортання білків)
11. Створення дизайну за допомогою комп'ютера
12. Складання порядку рішення завдань.

8.7. Еволюційні алгоритми

Генетичні алгоритми разом з еволюційною стратегією й еволюційним програмуванням представляють три головних напрямки розвитку так називаного еволюційного моделювання (simulated evolution).

Незважаючи на те, що кожен з цих методів виник незалежно від інших, вони характеризуються рядом важливих загальних властивостей. Для кожного з них формується вихідна популяція індивидів, що у наступному піддається селекції і впливові різних генетичних операторів (найчастіше схрещуванню і/або мутації), що дозволяє знаходити більш гарні рішення.

Еволюційні стратегії – це алгоритми, створені як методи рішення оптимізаційних задач і засновані на принципах природної еволюції.

Еволюційне програмування являє собою підхід, запропонований американськими вченими спочатку в рамках теорії кінцевих автоматів і узагальнений пізніше для додатків до проблем оптимізації. Обидва напрямки виникли в шістдесятих роках XX століття.

Сконцентруємо увагу на найважливіших подібностях і розходженнях між еволюційними стратегіями і генетичними алгоритмами. Очевидно, що **головна подібність** полягає в тім, що обидва методи використовують популяції потенційних рішень і реалізують принцип селекції і перетворення найбільш пристосованих індивідів. Однак обговорювані підходи сильно відрізняються один від одного.

Перше розходження полягає в способі представлення індивідів. ***Еволюційні стратегії оперують векторами дійсних чисел, тоді як генетичні алгоритми – двійковими векторами***.

Друге розходження між еволюційними стратегіями і генетичними алгоритмами криється *в організації процесу селекції*. При реалізації еволюційної стратегії формується проміжна популяція, що складається з усіх батьків і деякої кількості нащадків, створених у результаті застосування генетичних операторів. За допомогою селекції розмір цієї проміжної популяції зменшується до величини батьківської популяції за рахунок виключення найменш пристосованих індивідів. Сформована в такий спосіб популяція утворює чергове покоління. Навпроти, у генетичних алгоритмах передбачається, що в результаті селекції з популяції батьків відбирається кількість індивідів, рівна розмірності вихідної популяції, при цьому деякі (найбільш пристосовані) особи можуть відбиратися багаторазово. У той же час, менш пристосовані особи також мають можливість з’явитися в новій популяції. Однак шанси їхнього відбору пропорційні величині пристосованості індивідів. Незалежно від застосовуваного в генетичному алгоритмі методу селекції (наприклад, рулетки або рангового) більш пристосовані особи можуть відбиратися багаторазово. *При реалізації еволюційних стратегій особи відбираються без повторень*. В еволюційних стратегіях застосовується детермінована процедура селекції, тоді як у генетичних алгоритмах вона має випадковий характер.

Третє розходження між еволюційними стратегіями і генетичними алгоритмами стосується послідовності виконання процедур селекції і

рекомбінації (тобто зміни генів у результаті застосування генетичних операторів). При реалізації еволюційних стратегій *спочатку виконується рекомбінація, а потім селекція*. У випадку виконання генетичних алгоритмів ця послідовність інвертується. При здійсненні застосування еволюційних стратегій нащадок утворюється в результаті схрещування двох батьків і мутації. Формована в такий спосіб проміжна популяція, що складається з усіх батьків і отриманих від них нащадків, надалі піддається селекції, що зменшує розмір цієї популяції до розміру вихідної популяції. При виконанні генетичних алгоритмів спочатку виконується селекція, що приводить до утворення перехідної популяції, після чого генетичні оператори схрещування і мутації застосовуються до індивідів (обираних з ймовірністю схрещування) і до генів (обираних з ймовірністю мутації).

Наступне розходження між еволюційними стратегіями і генетичними алгоритмами полягає в тім, що параметри генетичних алгоритмів (такі, як ймовірності схрещування і мутації) залишаються сталими протягом усього процесу еволюції, тоді як при реалізації еволюційних стратегій ці параметри піддаються безперервним змінам (так називана самоадаптація параметрів).

Ще одне розходження між еволюційними стратегіями і генетичними алгоритмами стосується трактування обмежень, що накладаються на розв'язувані оптимізаційні задачі. Якщо при реалізації еволюційних стратегій на деякій ітерації нащадок не задовольняє всім обмеженням, то він відкидається і включається в нову популяцію. Якщо таких нащадків виявляється багато, то еволюційна стратегія запускає процес адаптації параметрів, наприклад, шляхом збільшення ймовірності схрещування. У генетичних алгоритмах такі параметри не змінюються. У них може застосовуватися штрафна функція для тих індивідів, що не задовольняють накладеним обмеженням, однак ця технологія має багато недоліків.

В міру розвитку еволюційних стратегій і генетичних алгоритмів протягом останніх років істотні розходження між ними поступово зменшуються. Наприклад, при реалізації генетичних алгоритмів для рішення оптимізаційних задач усе частіше застосовується представлення хромосом дійсними числами і різні модифікації «генетичних» операторів, що має на меті підвищити ефективність цих алгоритмів. Подібні методи, що значно відрізняються від класичного генетичного алгоритму, за традицією зберігають колишню назву, хоча більш коректно було б називати їх «еволюційними алгоритмами».

Еволюційне програмування, також як і еволюційні стратегії, робить основний упор на адаптацію і розмаїтість способів передачі властивостей від батька до нащадків у наступних поколіннях. Познайомимося зі стандартним алгоритмом еволюційного програмування.

Початкова популяція рішень вибирається випадковим чином. У задачах оптимізації значень дійсних чисел (прикладом яких може служити навчання нейронних мереж) особа (хромосома) представляється ланцюгом значень дійсних чисел. Ця популяція оцінюється щодо заданої функції (функції пристосованості). Нащадки утворюються від вхідних у цю популяцію батьків у результаті випадкової мутації. Селекція заснована на ймовірнісному виборі

(турнірний метод), при якому кожне рішення змагається з хромосомами, випадковим чином обираними з популяції. Рішення-переможці (які є найкращими) стають батьками для наступного покоління. Описана процедура повторюється так довго, поки не буде знайдено шукане рішення або не буде вичерпаний ліміт машинного часу.

Еволюційне програмування застосовується для оптимізації функціонування нейронних мереж. Так, як і інші еволюційні методи, воно не вимагає градієнтної інформації і тому може використовуватися для рішення задач, у яких ця інформація недоступна, або для її одержання потрібні значні обсяги обчислень. Одними з перших додатків еволюційного програмування вважаються задачі теорії штучного інтелекту, а самі ранні роботи стосувалися теорії кінцевих автоматів.

Спостерігається велика подібність між еволюційними стратегіями й еволюційним програмуванням у їхніх додатках до задач оптимізації безперервних функцій з дійсними значеннями. Деякі дослідники стверджують, що ці процедури, по суті, однакові, хоча вони і розвивалися незалежно одна від одної. Дійсно, обидва методи схожі на генетичні алгоритми. ***Принципове розходження між ними полягає в тому, що еволюційне програмування не зв'язане з конкретною формою представлення індивідів, оскільки оператор мутації не вимагає застосування якого-небудь спеціального способу кодування***.

Усі три представлених методи, тобто *генетичні алгоритми, еволюційні стратегії й еволюційне програмування поєднуються під загальною назвою еволюційні алгоритми (evolutionary algorithms)*. Також застосовується термін *еволюційні методи (evolutionary methods)*.

Еволюційними алгоритмами називаються й інші методи, що реалізують еволюційний підхід, зокрема, генетичне програмування (genetic programming), що представляє собою модифікацію генетичного алгоритму з урахуванням можливостей комп'ютерних програм. При використанні цього методу популяція складається з закодованих відповідним чином програм, що піддаються впливові генетичних операторів схрещування і мутації для знаходження оптимального рішення, яким вважається програма, що щонайкраще вирішує поставлену задачу. Програми оцінюються щодо визначеної спеціальним чином функції пристосованості. Цікавою представляється модифікація еволюційних алгоритмів для рішення оптимізаційних задач методом так називаної м’якої селекції, що запропонована Р. Галаром.

Для позначення різноманітних алгоритмів, заснованих на еволюційному підході, також застосовується *поняття еволюційних програм* (evolution programs). Цей термін поєднує як генетичні алгоритми, еволюційні стратегії й еволюційне програмування, так і генетичне програмування, а також інші аналогічні методи.

Еволюційні програми можна вважати узагальненням генетичних алгоритмів. Класичний генетичний алгоритм виконується при фіксованій довжині двійкових послідовностей і в ньому застосовуються оператори схрещування і мутації. Еволюційні програми обробляють більш складні

структури (не тільки двійкові коди) і можуть виконувати інші «генетичні» операції. Наприклад, еволюційні стратегії можуть трактуватися як еволюційні програми, у яких хромосоми представляються дійсними (не двійковими) числами, а мутація використовується як єдина генетична операція.

Розглянемо *узагальнений приклад еволюційної програми*. Припустимо, що шукається граф, що задовольняє певним обмеженням (наприклад, виконується пошук топології комунікаційної мережі, оптимальної за конкретними критеріями, наприклад, по вартості передачі і т.п.). Кожна особа в еволюційній програмі представляє одне з потенційних рішень, тобто в даному випадку деякий граф. Вихідна популяція графів P(0), сформована випадковим чином або створювана при реалізації якого-небудь евристичного процесу, вважається відправною точкою (k=0) еволюційної програми. Функція пристосованості, що зазвичай задається, зв'язана із системою обмежень розв'язуваної задачі. Ця функція визначає «пристосованість» кожного графа шляхом виявлення «кращих» і «гірших» індивідів. Можна запропонувати кілька різних операторів мутації, призначених для трансформації окремих графів, і трохи операторів схрещування, що будуть створювати новий граф у результаті рекомбінації структур двох або більш графів. Дуже часто такі оператори обумовлюються характером розв'язуваної задачі. Наприклад, якщо шукається граф типу «дерево», то можна запропонувати оператор мутації, що видаляє галузь з одного графа і додає нову галузь, що поєднує два окремих підграфи. Інші можливості полягають у проектуванні мутації незалежно від семантики задачі, але з включенням у функцію пристосованості додаткових обмежень - «штрафів» для тих графів, що не є деревами.

Принципову різницю між класичним генетичним алгоритмом і еволюційною програмою, тобто еволюційним алгоритмом у широкому змісті, показано на рис. 8.5.

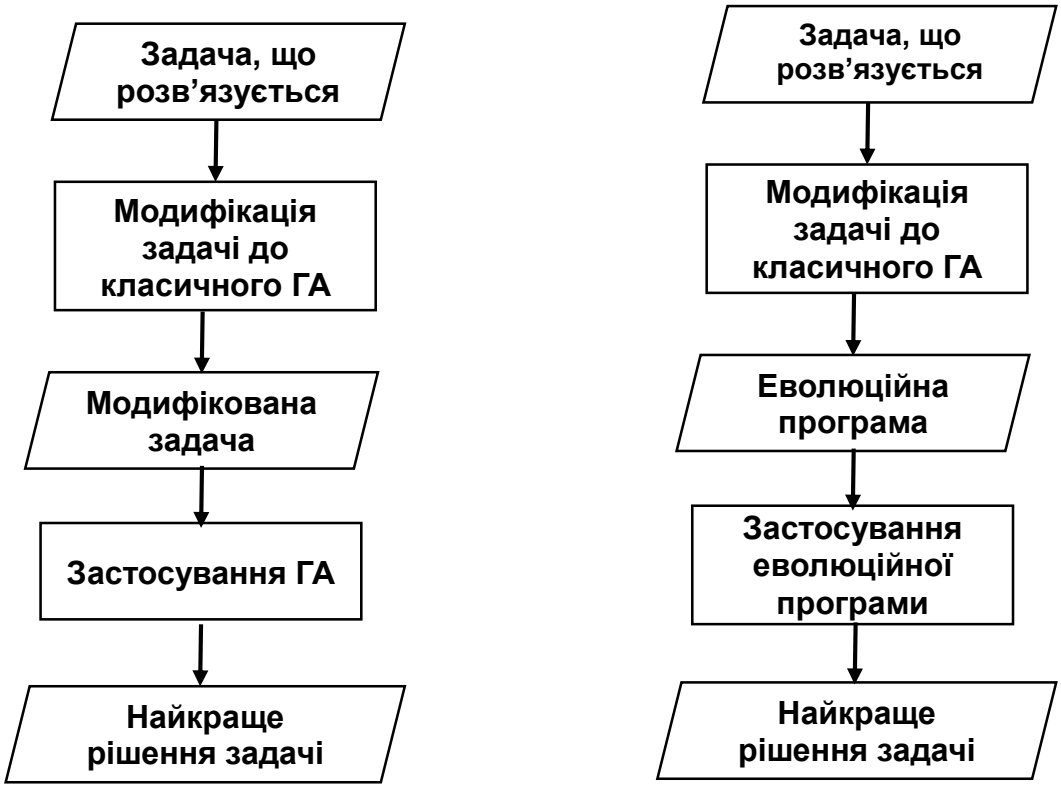


Рисунок 8.5 – Рішення задачі за допомогою класичного генетичного алгоритму a) і еволюційної програми b)

Класичний генетичний алгоритм, що оперує двійковими послідовностями, вимагає представити розв'язувану задачу в строго визначеному виді (відповідність між потенційними рішеннями і двійковими кодами, декодування і т.п.). Зробити це не завжди просто.

Еволюційні програми можуть залишити постановку задачі в незмінному виді за рахунок модифікації хромосом, що представляють потенційні рішення (з використанням «природних» структур даних), і застосування відповідних «генетичних» операторів. Іншими словами, для рішення нетривіальної задачі можна або перетворити її до виду, необхідному для використання генетичного алгоритму, або модифікувати генетичний алгоритм так, щоб він задовольняв задачі.

При реалізації першого підходу застосовується класичний генетичний алгоритм, а при реалізації другого підходу - еволюційна програма Таким чином, модифіковані генетичні алгоритми можна в загальному випадку називати еволюційними програмами. Однак найчастіше зустрічається термін еволюційні алгоритми. Еволюційні програми також можуть розглядатися як еволюційні алгоритми, підготовлені програмістом для виконання на комп'ютері. Основна задача програміста полягає при цьому у виборі відповідних структур даних і «генетичних» операторів. Саме таке трактування поняття еволюційна програма представляється найбільш обґрунтованою.

Усі поняття, застосовувані в цьому розділі і стосовні головним чином до методів, заснованим на еволюційному підході, можна зіставити головному напрямкові досліджень – комп'ютерному моделюванню еволюційних процесів.

Ця область інформатики називається Evolutionary Computation, що можна перевести як еволюційні обчислення.

До еволюційних алгоритмів також застосовується поняття технологія еволюційних обчислень. Можна додати, що назва генетичні алгоритми використовується як у вузькому змісті, тобто для позначення класичних генетичних алгоритмів і їхніх несуттєвих модифікацій, так і в широкому змісті - припускаючи будь-які еволюційні алгоритми, що значно відрізняються від «класики».

Останнім часом з`явилося багато нових методів, заснованих на еволюційному моделюванні, але базові технології, що використовуються – головним чином класичний генетичний алгоритм, еволюційні стратегії й еволюційне програмування.

8.8. Колективний інтелект (*Swarm intelligence*)

Swarm intelligence – термін, що описує комплексну колективну поведінку децентралізованої системи із самоорганізацією і розглядається в теорії штучного інтелекту як метод оптимізації. У природі таку поведінку демонструють різні тварини. Прикладом можуть служити зграя птахів, колонія мурах, рій бджіл, косяк риб і ін. Інколи колективний інтелект ще називають **ройовим інтелектом**.

Системи колективного інтелекту, як правило, складаються із множини агентів (мультигентна система), що локально взаємодіють як між собою так із навколишнім середовищем. Самі агенти зазвичай досить прості, але всі разом, локально взаємодіючи, створюють так званий колективний інтелект.

Алгоритм пташиної зграї

У 1986 році Craig Reynolds створив комп’ютерну модель Boids (bird-like-object), яка моделює поведінку рою (зграї птахів), який дотримується деяких правил. Крейг Рейнольдс визначив поведінку кожного з птахів окремо, а також їх загальну взаємодію в зграї. Він вивів три елементарних правила поведінки.

- *Правило поділу*: кожен птах повинен прагнути уникнути зіткнень з іншими птахами зграї.
- *Правило вирівнювання*: кожен птах зграї повинен рухатися в тому ж напрямку, що і птахи, які знаходяться неподалік.
- *Правило згуртованості*: птахи повинні прагнути рухатися на рівній відстані один від одного, прагнучі до середньої позиції (центру мас) зграї.

Багато моделей, розроблених пізніше, використовують варіації цих правил, та визначають зони їх дії.

- В *зоні відштовхування*, що обмежує окіл поряд з твариною, кожна тварина прагне дистанціюватися від своїх сусідів, щоб уникнути зіткнення.
- Трохи далі, в *зоні вирівнювання*, кожна тварина прагне вирівняти напрямок руху зі своїми сусідами.
- В крайній *зоні тяжіння*, що сягає відстані, на яких тварина здатна відчути своїх сусідів, тварини прагнуть рухатися в бік сусіда.

Алгоритм рою частинок (Particle swarm optimization)

Нехай f: Rn→R – цільова функція, яку потрібно мінімізувати, S – кількість частинок в рої, кожній з яких приписана координата $x_i \in Rn$ і швидкість $v_i \in Rn$ в просторі рішень. Нехай також p_i – найкраще з відомих положень частинки i, а g – найкращий відомий стан рою в цілому. Тоді загальний вигляд методу рою частинок такий.

1. Для кожної частинки i = 1, ..., S зробити:

1.1. Згенерувати початкове положення частинки за допомогою випадкового вектора $x_i \sim U$ (blo, bup), що має багатовимірний рівномірний розподіл (blo і bup – нижня і верхня межі простору рішень відповідно).

1.2. Присвоїти кращому відомому положенню частинки його початкове значення: $p_i \leftarrow x_i$.

1.3 Якщо (f (p_i) <f (g)), то оновити найкращий відомий стан рою: g ←p_i.

1.4 Присвоїти значення швидкості частинки:

$$v_i \sim U (- (bup-blo), (bup-blo)).$$

2. Поки не досягнутий критерій зупинки (наприклад, досягнення заданої кількості ітерацій або значення цільової функції), повторювати для кожної частинки i=1, ..., S:

2.1. Згенерувати випадкові вектори r_p, $r_g \sim U (0,1)$.

2.2. Оновити швидкість частинки:

$$v_i \leftarrow \omega v_i + \varphi_p r_p \times (p_i - x_i) + \varphi_g r_g \times (g - x_i),$$

де операція × означає покомпонентне множення.

2.3. Оновити положення частинки перенесенням x_i на вектор швидкості: $x_i \leftarrow x_i + v_i$. Зауважимо, що цей крок виконується незалежно від поліпшення значення цільової функції.

2.4. Якщо (f (x_i) <f (p_i)), то робити:

2.4.1. Оновити краще відоме положення частинки: $p_i \leftarrow x_i$.

2.4.2. Якщо (f (p_i) <f (g)), то оновити стан рою в цілому: g←p_i.

3. Тепер g містить найкраще з знайдених рішень.

Параметри ω, φ_p, і φ_g вибираються користувачем і визначають поведінку і ефективність методу в цілому.

Мурашиний алгоритм

Ідея мурашиного алгоритму – моделювання поведінки мурах, пов’язаної з їх здатністю швидко знаходити найкоротший шлях від мурашника до джерела їжі і адаптуватися до умов, що змінюються, знаходячи новий найкоротший шлях.

При своєму русі мураха мітить шлях феромоном, ця інформація використовується іншими мурахами для вибору шляху. Це елементарне правило поведінки і визначає здатність мурашок знаходити новий шлях, якщо старий виявляється недоступним. На рис. 8.6. Показаний приклад зміни шляху при появі перешкоди.

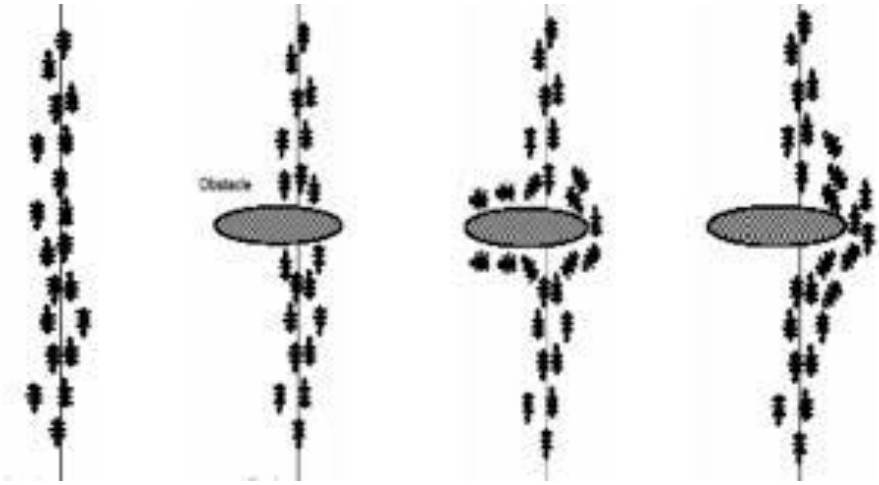


Рисунок 8.6 – Поведінка мурах при виникненні перешкоди на шляху

Дійшовши до перепони, мурахи з однаковою ймовірністю будуть обходити її справа і зліва. Те ж саме буде відбуватися і на зворотному боці перепони. Однак, ті мурахи, які випадково виберуть найкоротший шлях, будуть швидше його проходити, і за кілька пересувань він буде більш збагачений феромоном. Оскільки рух мурах визначається концентрацією феромону, то наступні мурахи будуть віддавати перевагу саме цьому шляху, продовжуючи збагачувати його феромоном до тих пір, поки цей шлях з якої-небудь причини не стане недоступним.

Моделювання випаровування феромону – негативний зворотній зв'язок – гарантує нам, що знайдене локально оптимальне рішення не буде єдиним – мурашки будуть шукати й інші шляхи. Якщо ми моделюємо процес такої поведінки на деякому графі, ребра якого є можливі шляхи переміщення мурашок, протягом певного часу, то найбільш збагачений феромоном шлях по ребрах цього графа і буде рішенням задачі, отриманим за допомогою мурашиного алгоритму.

Застосування мурашиного алгоритму (задача комівояжера)

Імовірність переходу з вершини *i* в вершину *j* визначається за формулою:

$$P_{ij}(t) = \frac{\tau_{ij}(t)^{\alpha}\left(\frac{1}{d_{ij}}\right)^{\beta}}{\sum_{j \in allowd\ nodes} \tau_{ij}(t)^{\alpha}\left(\frac{1}{d_{ij}}\right)^{\beta}}$$

де:

$\tau_{ij}(t)$ – рівень феромону;

d_{ij} – евристична відстань,

α, β – константи

При $\alpha = 0$ вибір найближчого міста найбільш ймовірний

При $\beta = 0$ вибір відбувається тільки на основі феромону

Рівень феромону оновлюється відповідно до формули:

$$\tau_{ij} = (t+1) = (1-\rho)\tau_{ij}(t) + \sum_{\substack{k \in Colony\ that \\ used\ edge(i,j)}} \frac{Q}{L_k}$$

де:

ρ – інтенсивність випаровування;

$L_k(t)$ – вартість поточного рішення рівень для *k*-ої мурахи,

Q – порядок вартості оптимального рішення,

$\frac{Q}{L_k(t)}$ – феромон, що залишає *k*-та мураха на ребрі *(i, j)*.

Млин (карусель) смерті (Death mill)

Природне явище, коли одна або невелика група мурах починає без видимої причини бігати по колу, поступово залучаючи в свій нескінчений цикл все більше і більше інших мурах рис.8.7.

В 1921 році американський мандрівник Уільям Біб описав бачене ним в Гайяні коло мурах-ецитонів діаметром близько 365 метрів, в якому кожна мураха виконувала повний цикл за 2,5 години.

Ця карусель, усіяна мертвими тілами, тривала 2 дні, поки невелика група робочих мурах випадково не відокремилась від загального руху і не повела тих, що залишилися в живих з а собою в ліс.

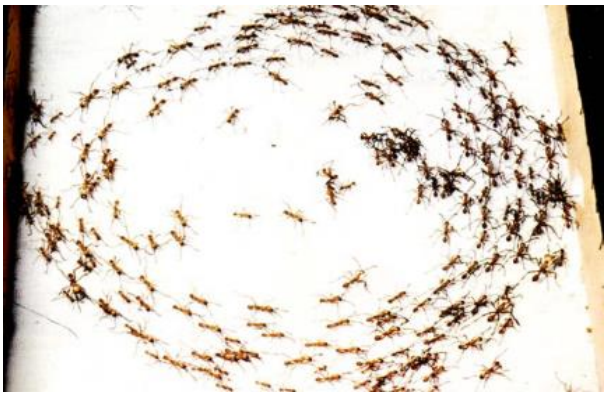


Рисунок 8.7 – Приклад руху мурах по колу

Алгоритм бджолиного рою (Bees Algorithm)

Початкові дані. У бджолиній сім'ї розрізняються три типи бджіл: розвідники не зайняті і фуражири та існує певне джерело нектару, яке характеризується своєю корисністю:

- віддаленість від вулика,
- концентрація нектару,
- зручність його видобутку.

Бджоли-розвідники здійснюють обліт території в пошуку джерел нектару.

Після повернення у вулик, бджола-розвідник виконує «вербування» незайнятих бджіл за допомогою танцю на спеціальному майданчику вулика – галузі танців. Одночасно в межах простору танців різні бджоли можуть «рекламувати» різні джерела нектару.

Логічним є припущення, що ймовірність вербування визначається корисністю відповідного джерела нектару.

Завербована бджола слідує за відповідною бджолою-розвідником до галузі з нектаром і стає, таким чином, зайнятим фуражиром.

Зайнятий фуражир після видобутку нектару повертається у вулик. Після цього даний фуражир може виконати одну з наступних дій:

- залишити «своє» джерело нектару і стати незайнятим фуражиром;
- продовжити заготівлю нектару з колишнього джерела, не вербуючи інших бджіл;
- виконати вербування.

Бджола вибирає одну із зазначених дій по

9 ЗАСТОСУВАННЯ ШІ В ЗАДАЧАХ КІБЕРБЕЗПЕКИ

9.1 Загальні питання застосування ШІ в кібербезпеці

Сучасні загрози кібербезпеці вимагають набагато більшої швидкості реагування, ніж це дозволяє людина. Враховуючи швидке зростання обсягу та частоти атак зловмисного програмного забезпечення, системи кіберзахисту ШІ все частіше впроваджуються для проактивного виявлення та пом'якшення загроз. У той час як традиційні антивірусні методи покладаються на «чорний список» історичних загроз на основі сигнатур вірусів, антивірус на основі штучного інтелекту може розпізнавати аспекти програмного забезпечення, яке може бути шкідливим, без необхідності покладатися на заздалегідь визначений список. Як підсумовано в нещодавньому звіті Darktrace, компанії з кібербезпеки штучного інтелекту:

У той час як застарілі інструменти безпеки часто можуть ідентифікувати відомі загрози, які вже були виявлені «в дикій природі», штучний інтелект може однозначно помітити слабкі та непомітні сигнали кіберзагрози, яку раніше не бачили. Ця можливість стала необхідною в останні роки, оскільки просунуті кіберзлочинці продовжують розробляти нові тактики, методи та процедури, спеціально розроблені для ухилення від засобів контролю, які були попередньо запрограмовані на основі сигнатур минулих атак.

Подібним чином можна навчити системи виявлення мережі на основі штучного інтелекту вивчати, що є «нормальною» активністю в мережі організації, виявляти ненормальний мережевий трафік на основі аналізу даних журналу та реагувати в режимі реального часу. Відповідно, ці методи можна використовувати для виявлення та позначення аномальної активності системи, яка може свідчити про внутрішню загрозу. У звіті Cybersecurity Insiders за 2018 рік було виявлено, що «86% організацій уже мають або створюють програму інсайдерських загроз», в основному базуючись на платформах «виявлення та запобігання вторгнень (IDS), керування журналами та SIEM [інформація про безпеку та керування подіями]».

Автентифікація користувачів є ще однією потенційною цінністю для UKIC. Останні дослідження зосереджені на використанні так званої «поведінкової біометрії» для ідентифікації користувачів на основі унікальних аспектів їхньої цифрової діяльності, наприклад, як вони користуються мишею або складають речення в документі. Такі системи активної автентифікації можуть посилити кібербезпеку, забезпечуючи постійну автентифікацію користувача після початкового входу в сеанс.

Чотирьохетапна модель кібербезпеки

Найвідомішою схемою концептуалізації кібербезпеки є кібербезпека Framework, розроблена Національним інститутом стандартів і технологій (NIST) рис. 9.1.

NIST FUNCTIONS	OUR MODEL
Identify	Prevention
Protect	
Detect	Detection
Respond	Response and Recovery
Recover	
N/A	Active Defense

Рисунок 9.1 – Напрями застосування ШІ в кібербезпеці

Хронологія розвитку машинного навчання для трьох основних завдань кібербезпеки подана на рис. 9.2.

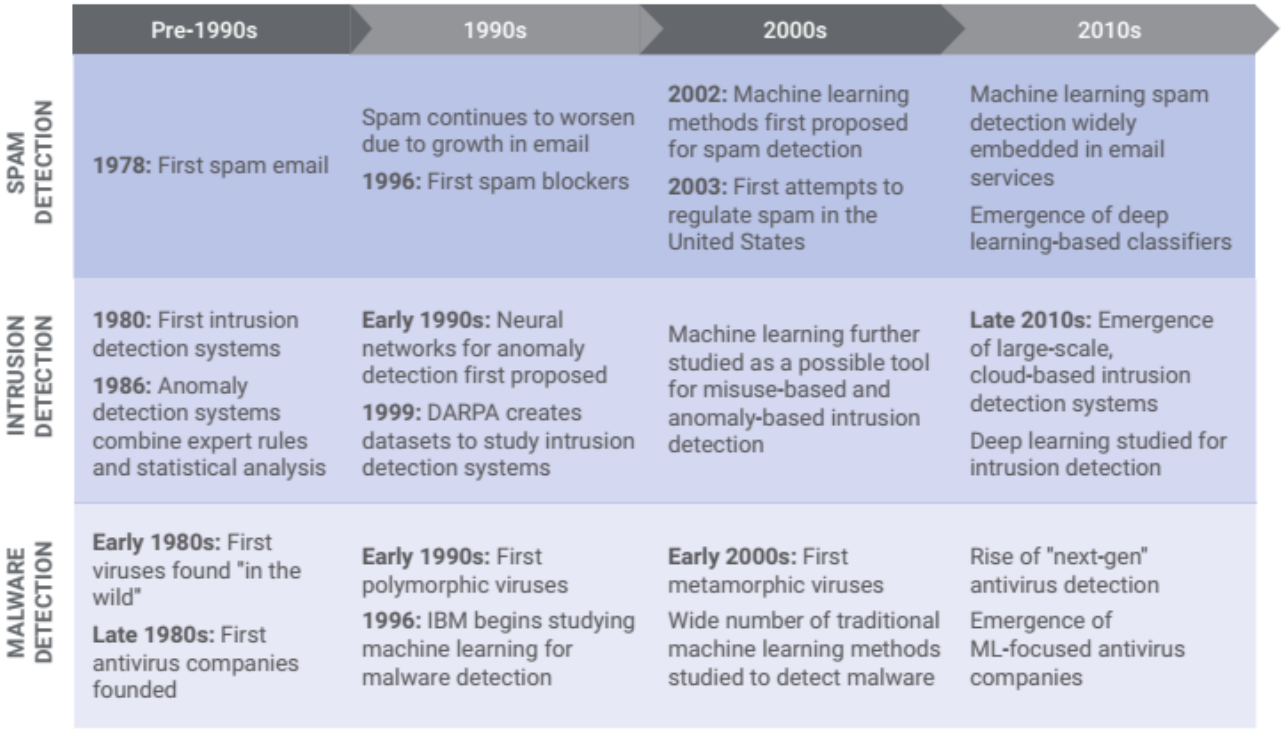


Рисунок 9.2 – Етапи застосування ШІ в кібербезпеці

9.2. Практичне застосування нечіткої логіки в задачах кібербезпеки

Бездротові мережі на сьогоднішній день використовуються практично у всіх сферах діяльності. Широке використання бездротових мереж обумовлено тим, що вони можуть використовуватися не тільки на персональних комп'ютерах, а й в телефонах, планшетах і ноутбуках, їх зручністю і порівняно невисокою вартістю.

В наш час методи впливу на конкурентів переходять від фізичного впливу до інтелектуального. При цьому використовуються новітні способи і засоби

несанкціонованого отримання інформації. Саме тому актуальним є необхідність оцінки ризиків інформаційної безпеки для побудови ефективних систем захисту інформації . Існуючі методи оцінки ризиків в більшості своїй засновані на теоріях ймовірності і класичних множин. Ці методи не дозволяють врахувати той факт, що будь-яка складна система є динамічною системою з набором невизначених даних. Системи оцінки ризиків, побудовані на застосуванні нечіткої логіки характеризуються логічністю і високою стійкістю в тому випадку, коли аналіз ризиків здійснюється в умовах нестачі даних і знань. Тому зраз є розроблені моделі нечіткі оцінки рівня захисту інформації та програмні засоби на її основі. Розглянемо одну з методик виявлення однієї з найпоширеніших кібератак – JS (HTML)/ScrInject на основі застосування математичного апарату теорії нечітких множин та нечіткого логічного виводу.

Методика виявлення кібератак типу JS(HTML)/SCRINJECT на основі апарату теорії нечітких множин

Розробка методики базується на алгоритмі дій, який включає в себе етапи підготовки вхідних даних, фазіфікації значень досліджуваних параметрів та здійснення процедури нечіткого логічного виводу.

Шкідливе програмне забезпечення (ШПЗ) JS (HTML)/ScrInject – троянська програма, зазвичай завантажується з веб-сайтів, які пропонують користувачам завантажувати та/або запускати оновлення, наприклад, для таких програмних додатків, як Flash Player або Java Virtual Machine. Після запуску, ШПЗ ScrInject налаштовується на автоматичний старт та за допомогою JavaScript-сценаріїв перенаправляє користувача на небезпечні ресурси, що дозволяє кіберзлочинцям у подальшому здійснювати інформаційно-руйнівні впливи на інфіковану систему.

Послідовність характерних дій даного ШПЗ зводиться до:

- спроб звернення до Інтернет-ресурсів із шкідливим вмістом з погодженням
- користувача або без такого, при чому користувачеві виводиться повідомлення із запитом зовсім іншого контексту;
- автоматичного запуску сценаріїв на завантаження іншого ПЗ без згоди користувача;
- ініціювання запису ПЗ до автоматичного запуску із завантаженням системи;
- помітного для користувача додаткового навантаження на систему (ресурси оперативної пам’яті, завантаження процесора, повільний відклик системи на дії);
- ініціювання завантаженими сценаріями досить великої кількості запитів до різноманітних Інтернет-ресурсів (сервісів) під виглядом оновлень.

Оскільки модель даної кібератаки є не параметричною, а поведінковою, то доцільно на етапі її виявлення опиратися саме на шаблон (pattern) поведінки.

Методика виявлення передбачає визначення функцій системи, а також потоків інформації, які пов’язують між собою вказані функції. Відповідно до методології функціонального моделювання IDEF0, на рис. 9.3. представлено

контекстну діаграму рівня А-0, на якій згідно аналізу цілей та функцій СВА визначено вхідні та вихідні дані, управляючі компоненти та механізми (суб'єкти), що впливають на результат. Контекст системи обмежено однією ітерацією процесу визначення належності кібератаки до типу JS (HTML)/ScrInject.

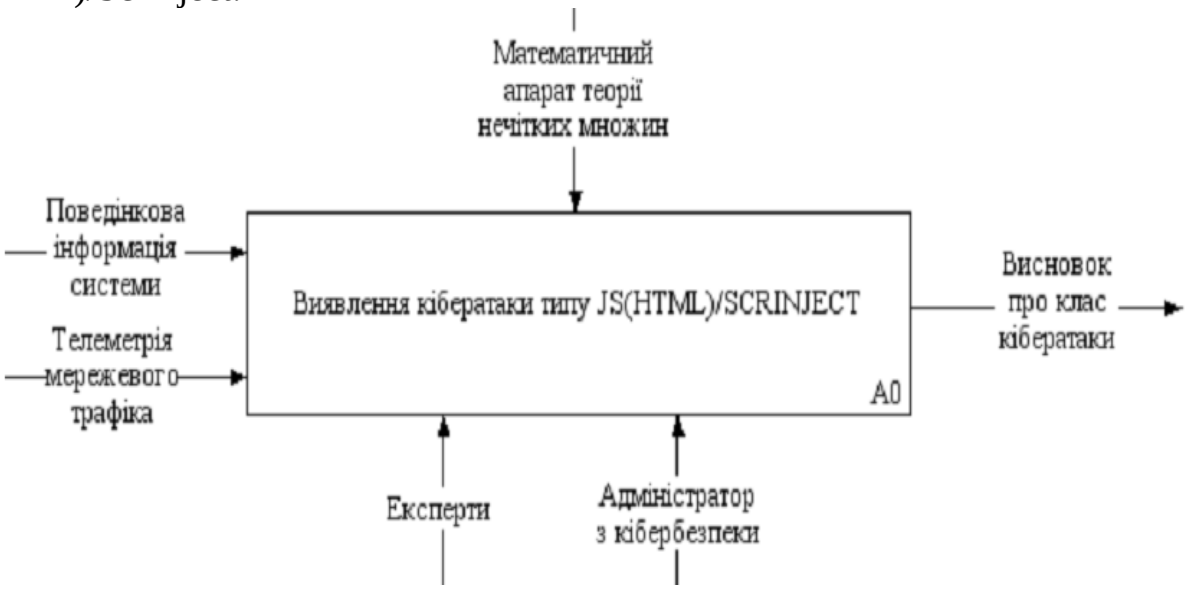


Рисунок 9.3 – Контекстна А-0 діаграма процесу визначення класу кібератаки

Вхідними даними є поведінкова інформація в інформаційно-телекомунікаційній мережі (ІТМ), на основі аналізу якої приймається рішення щодо виявлення кібератаки та телеметрія мережевого трафіку – статистичні дані для уточнення (адаптації) функцій належності з метою врахування особливостей та умов функціонування системи – об'єкта захисту.

Управляючим компонентом є математичний апарат теорії нечітких множин.

Суб'єкти, за допомогою яких відбувається даний процес: експерти – на етапі налаштування системи виявлення атаки (СВА), зокрема заповнення бази знань (БЗ) правилами та адміністратор з кібербезпеки – особа, якій надаються управлінські рішення системою про стан ІТМ для здійснення подальших превентивних заходів у разі необхідності.

Вихідними даними є перевірена на відповідність описаному шаблону поведінки кібератаки інформація у вигляді логічного висновку, який характеризує поточний стан безпеки ІТМ щодо наявності досліджуваної кібератаки у режимі реального часу RTC (Real-Time Clock).

Отже методика виявлення кібератаки типу JS (HTML)/ScrInject складається з наступних етапів, що визначають порядок дій для своєчасного та достовірного її виявлення, які представлено у вигляді функціональної схеми на рис. 9.4.

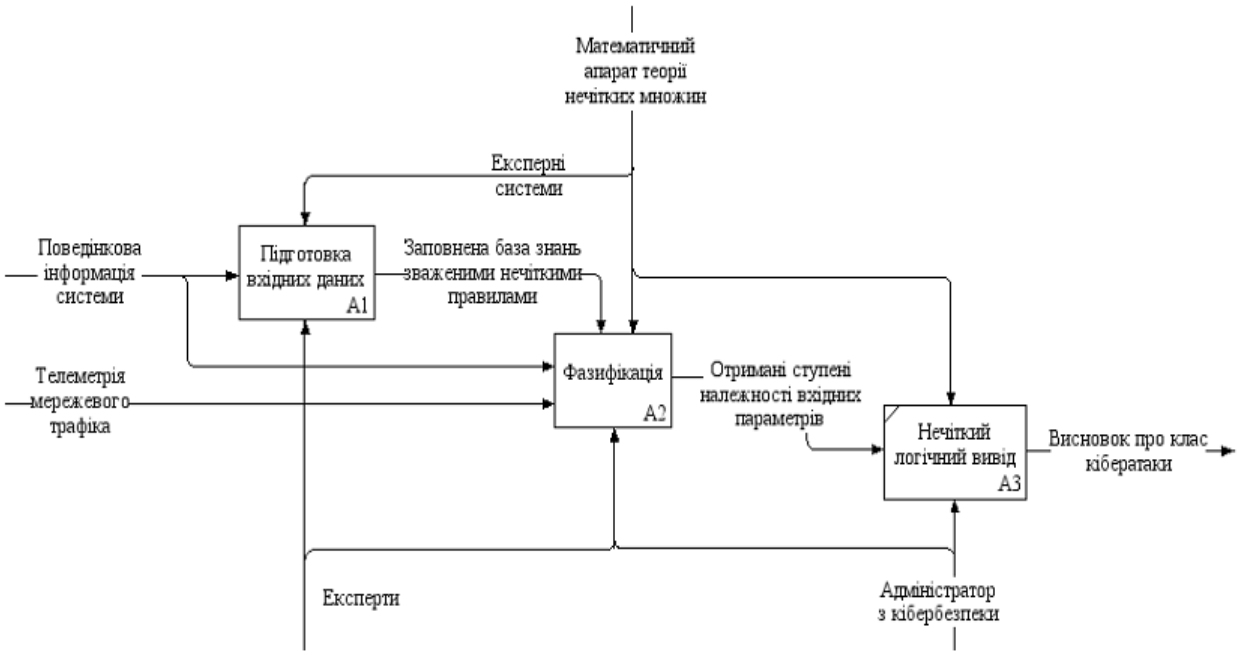


Рисунок 9.4 – Функціональна схема процесу визначення класу кібератаки типу JS (HTML)/ScrInject

1 етап – підготовка вхідних даних. Виконання цього етапу характеризуються послідовністю дій, які представлені наступною функціональною діаграмою на рис. 9.5.

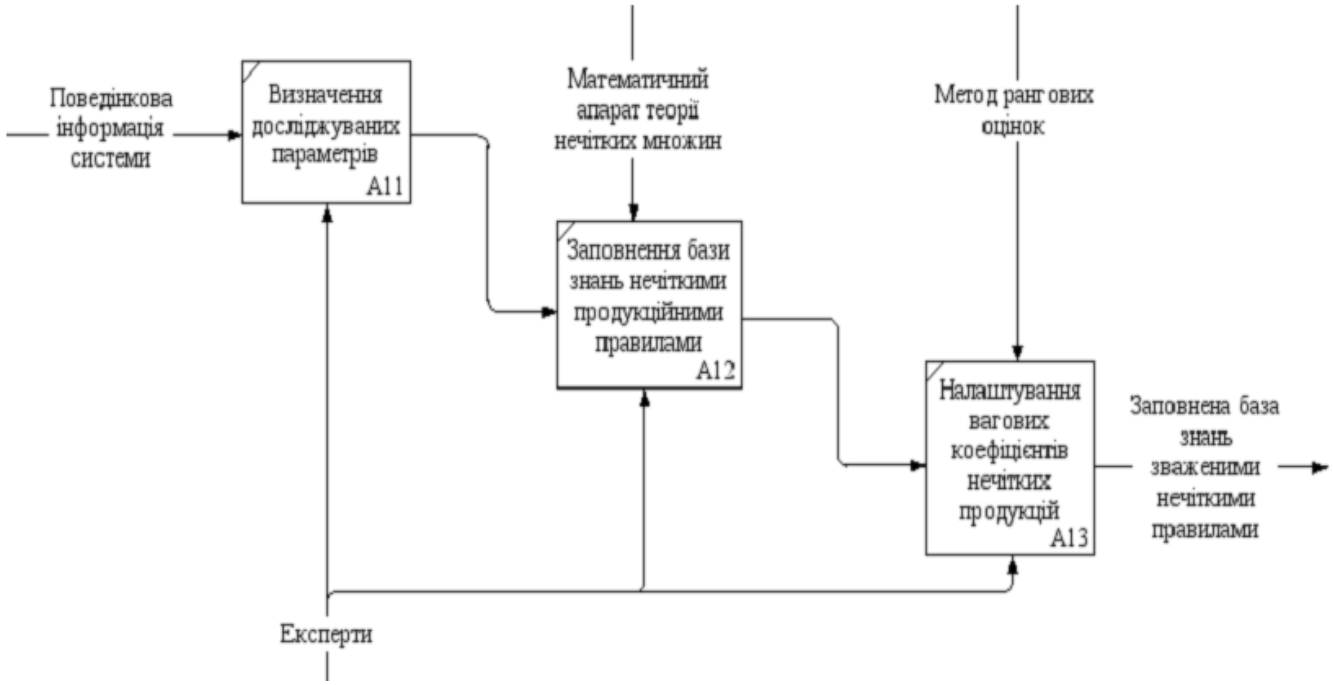


Рисунок 9.5 – Функціональна схема підготовки вхідних даних для класу кібератаки типу JS (HTML)/ScrInject

Вхідними даними на даному етапі є поведінкова інформація процесів в ІТМ, яка представлена наступними ***лінгвістичними змінними*** на основі аналізу характерних дій кібератаки JS (HTML)/ScrInject:

GET_NET – звернення до Інтернет-ресурсів із шкідливим вмістом з погодженням користувача або без такого;

AUTOSTART_BOOT – автоматичний запуск сценаріїв на завантаження іншого ПЗ без згоди користувача;

AUTOSTART – ініціювання запису ПЗ до автоматичного запуску із завантаженням операційної системи;

BOOT_LEVEL – рівень завантаженості операційної системи у відсотках;

REQUEST – ініціювання досить великої кількості запитів до різноманітних Інтернет-ресурсів (сервісів) під виглядом оновлень.

Наступним процесом є заповнення БЗ нечіткими правилами, що описують стани поведінки в ІТМ з наступними лінгвістичними термами та відповідними значеннями:

GET_NET, AUTOSTART_BOOT, AUTOSTART, REQUEST – {„Н – низька [до 5]”, „С – середня [5,10,15,20]”, „В – велика [від 20 і вище]”} на універсумі [0,50];

BOOT_LEVEL – {„Н – низький [до 25]”, „нС – нижче середнього [20, 30, 40]”, „С – середній [40, 45, 55, 65]”, „вС – вище середнього [60, 70, 80]”, „В – високий [від 80 і вище]”} на універсумі [0, 100].

На рис. 9.6. представлено графічне зображення описаних лінгвістичних термів функцій належності.

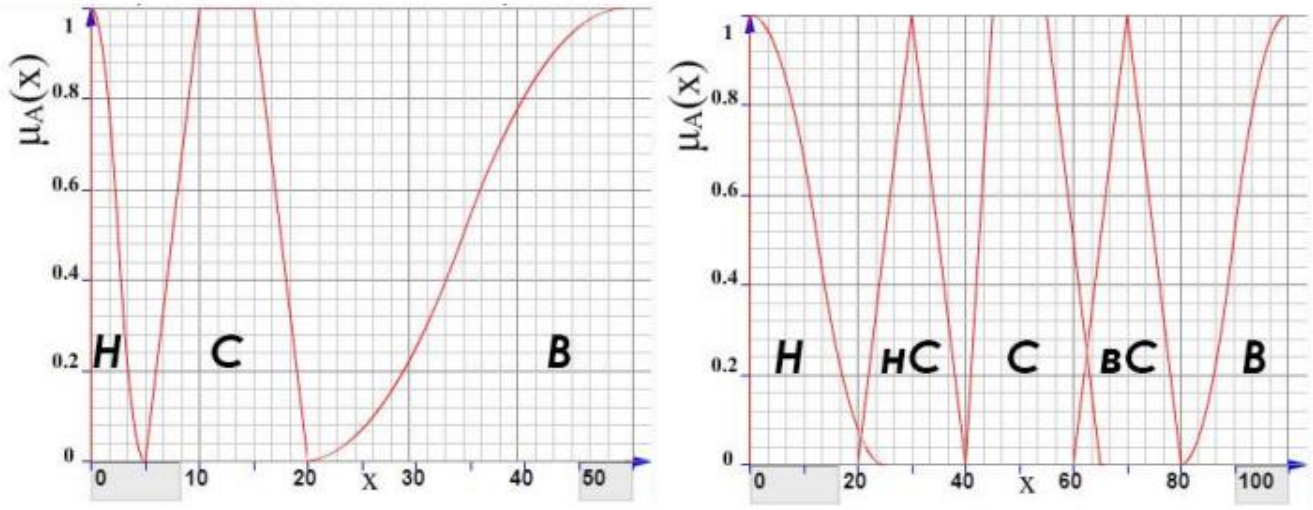


Рисунок 9.6 – Графічне зображення лінгвістичних термів функцій належності

У зведеній табл. 9.1 представлені правила з БЗ та відповідні їм експертні рішення.

Таблиця 9.1 – Стани ІТМ описані нечіткими зваженими правилами

Вхідні лінгвістичні змінні (ознаки атаки „*back*”)					Ваговий коефіцієнт	Висновок
GET_NET	*AUTOSTART_BOOT*	*AUTOSTART*	*BOOT_LEVEL*	*REQUEST*		*y*
H	H	H	H	H	w_1	d_1
H	H	H	B	H	w_2	d_2
B	C	C	вC	B	w_3	d_3
B	B	B	вC	B	w_4	d_4
B	B	B	B	B	w_5	d_5

Де d_1 – d_5 – варіанти експертних рішень щодо станів ІТМ:

d_1 – d_2 – нормальний стан ІТМ;

d_3 – d_5 – наявність кібератаки в ІТМ класифікованої як JS (HTML)/ScrInject.

Управляючими компонентами є математичний апарат теорії нечітких множин та нечіткого логічного виводу, а також вагові коефіцієнти правил *W*, отримані на основі застосування методу рангових оцінок з метою ранжування (впорядкування) нечітких правил, визначених експертами з кібербезпеки, що характеризують їх впевненість у кожному прийнятому рішенні. Так, - w_1 – w_5– вагові коефіцієнти нечітких правил у БЗ. Такий підхід дозволяє підвищити ефективність нечіткого логічного виводу, оскільки враховується відносна значущість (перевага) нечітких правил.

Суб’єкти, за допомогою яких відбувається даний процес – експерти з кібербезпеки.

Вихідні дані – заповнена нечітка база зважених правил про стани ІТМ.

2 етап – фазифікація. Виконання даного етапу характеризуються послідовністю дій, які представлені наступною функціональною діаграмою на рис. 9.5:

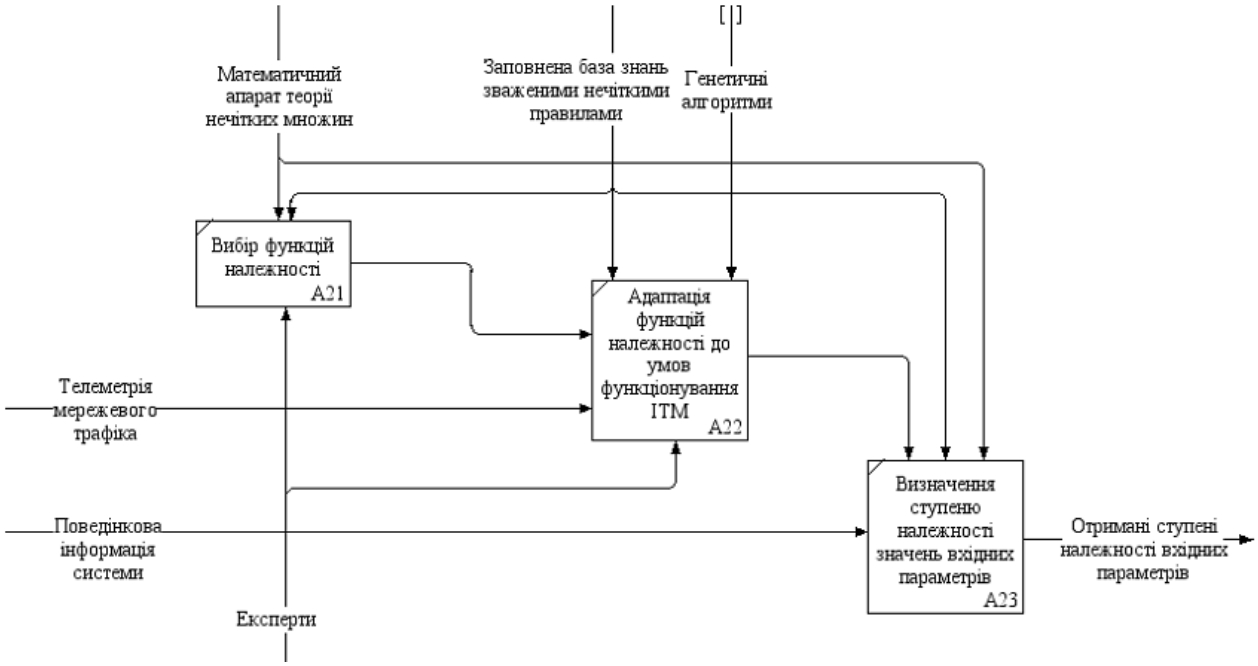


Рисунок 9.5 – Функціональна діаграма другого етапу методики

На даному етапі запропонованої методики визначається ступінь належності значень досліджуваних параметрів терм-множинам вищевказаних лінгвістичних змінних.

Дослідження ознак даної кібератаки проводилось на основі Z,S – подібного (початок/кінець діапазону значень), трикутного (Δ) та трапецеїдального (T) (проміжні значення діапазону значень) типу функцій належності з наступними значеннями (1, 2):

GET_NET, AUTOSTART_BOOT, AUTOSTART, REQUEST – {„Н – низька [до 5]”, „С – середня [5, 10, 15, 20]”, „В – велика [від 20 і вище]”} на універсумі [0, 50];

BOOT_LEVEL – {„Н – низький [до 25]”, „нС – нижче середнього [20, 30, 40]”, „С – середній [40, 45, 55, 65]”, „в С – вище середнього [60, 70, 80]”, „В – високий [від 80 і вище]”} на універсумі [0,100].

З метою забезпечення ефективного виявлення кібератак типу JS (HTML)/ScrInject у різних ІТМ за призначенням, топологією та специфікою доцільно враховувати умови та особливості їх функціонування. Так, налаштовані експертами значення термів функцій належності, від яких багато в чому залежить точність та достовірність отриманого логічного висновку потребують уточнення (адаптації) на основі статистичних даних телеметрії мережевого трафіку. Ця задача може бути вирішена шляхом застосування математичного апарату генетичних алгоритмів, на кожній ітерації якого діапазони терм-множин функцій належності підлягають оцінці функцією відповідності, яка в свою чергу приймає участь у формуванні нової їх популяції. В результаті розвитку такої популяції алгоритм зводиться до вибору оптимальних або субоптимальних значень термів.

Визначення ступеню належності значень досліджуваних параметрів, що описують поведінку процесів у ІТМ здійснюється на основі моделей:

$$\mu_Z(x;a,b)=\begin{cases}1, x\le a\\ \frac{1}{2}+\frac{1}{2}\cos\left(\frac{x-a}{b-a}\pi\right), a\le x\le b\\ 0, x>b\end{cases}\quad \mu_S(x;a,b)=\begin{cases}0, x<a\\ \frac{1}{2}+\frac{1}{2}\cos\left(\frac{x-b}{b-a}\pi\right), a\le x\le b\\ 1, x>b\end{cases}$$

$$\mu_\Delta(x;a,b,c)=\begin{cases}0, x\le a\\ \frac{x-a}{b-a}, a\le x<b\\ \frac{c-x}{c-b}, b\le x\le c\\ 0, c\le x\end{cases}\qquad \mu_T(x;a,b,c,d)=\begin{cases}\frac{x-a}{b-a}, a\le x<b\\ 1, b\le x<c\\ \frac{d-x}{d-c}, c\le x\le d\\ 0, x\notin(a,d)\end{cases}$$

Управляючими компонентами є математичний апарат теорії нечітких множин та нечіткого логічного виводу, БЗ та штучні генетичні алгоритми.

Суб’єкти, за допомогою яких відбувається даний процес – експерти з кібербезпеки та адміністратор, який має змогу приймати участь в налаштуванні СВА у зв’язку з безпосереднім виконанням посадових інструкцій на конкретній ІТМ.

Вихідні дані – розраховані ступені належності значень досліджуваних параметрів щодо стану ІТМ.

3 етап – нечіткий логічний вивід. Виконання даного етапу передбачає проведення розрахунків над отриманими ступенями належності на попередньому етапі для всіх експертних рішень у нечіткій БЗ з метою визначення найбільш відповідного висновку на основі застосування нечітких максимінних операцій. Описаний процес можливо представити у вигляді наступної системи нечітких логічних рівнянь:

$$\mu^{d_j}(x_1, x_2, ..., x_n) = \max_{p=1,k_j} \left\{ w_{jp} \min_{p=1,k_j} \left[\bigwedge_{i=1}^{n} \mu^{jp}(x_i) \right] \right\}, \quad j = \overline{1,m}$$

Так, при отриманні значень у режимі RTC, що відповідають експертним висновкам адміністратору з кібербезпеки буде виведено рішення про наявність в ІТМ кібератаки, класифікованої як JS (HTML)/ScrInject.

Вхідними даними є ступені належності значень вхідних параметрів терм-множинам обраних функцій належності. Управляючим компонентом є математичний апарат теорії нечітких множин та нечіткого логічного виводу.

Суб'єкти, за допомогою яких відбувається даний процес: адміністратор з кібербезпеки – особа, якій надаються управлінські рішення системою про стан ІТМ для здійснення подальших превентивних заходів у разі необхідності.

Вихідними даними є перевірена на відповідність описаному шаблону поведінки кібератаки інформація у вигляді логічного висновку, який характеризує поточний стан безпеки ІТМ щодо наявності досліджуваної кібератаки у режимі реального часу RTC (Real-Time Clock).

Таким чином, представлено методику виявлення кібератак типу JS (HTML)/ScrInject, яка на відміну від існуючих, забезпечує їх виявлення в умовах режиму реального часу функціонування ІТМ на основі дослідження параметрів, якими характеризується кібератака. Експериментальна перевірка застосування запропонованої методики на практиці дозволяє зробити висновок про підвищення точності та достовірності виявлення розглянутої кібератаки СВА. Практична цінність методики полягає у можливості виявлення кібератак як в умовах невизначеності та нечіткості управляючої інформації, так і з врахуванням умов та особливостей функціонування об'єкта захисту.

9.2 Виявлення аномалій у поведінці мережі

Багато наборів даних (журналів, конфігурацій, мобільного трафіку тощо) безперервно надходять із телекомунікаційних мереж і називаються «великими даними», терміном, який описує великий і розподілений характер наборів даних.

Існує багато статистичних визначень великих даних. Більшість із них описує їх як масивні, високошвидкісні та різноманітні набори даних, які вимагають економічно ефективної аналітики нових даних для прийняття рішень і отримання цінних висновків. В останні роки основні проблеми великих даних були широко визначені. Вони містяться в п'яти V великих даних – Value, Veracity, Variety, Velocity та Volume.

Цінність стосується переваг, пов'язаних з аналізом даних; правдивість відноситься до точності даних; а різноманіття відноситься до багатьох типів даних, таких як структуровані, напівструктуровані або неструктуровані. Обсяг – це кількість накопичених даних (тобто розмір даних). Що більша розмірність даних, то більший обсяг.

Розмірність стосується кількості ознак, атрибутів або змінних у даних, доступних для аналізу. Навпаки, швидкість означає «швидкість», з якою генеруються дані, і може містити багато вимірів. Ці елементи поточного визначення великих даних стосуються фундаментальних проблем.

Виявлення аномалій спрямоване на виявлення аномальних моделей, що відрізняються від решти великих даних, які називаються аномаліями або викидами. Висока розмірність створює труднощі для виявлення мережевих аномалій. Коли кількість атрибутів або ознак збільшується, кількість даних, необхідних для точного узагальнення, також зростає, що призводить до розрідженості даних, у якій точки даних більш розкидані та ізольовані.

Що таке виявлення аномалій?

Це питання отримало багато назв, таких як «виявлення аномалій», «виявлення викидів», «виявлення новизни», «виявлення вторгнень» і «виявлення піків». Не існує загальноприйнятого визначення аномалії чи викиду. Загалом, виявлення аномалій (AD) визначає неочікувані елементи або події в наборах даних. З точки зору статистики, припускаючи розподіл подій, аномалії вважаються малоймовірними подіями щодо визначеного порогу. Існує також подієво-орієнтований погляд на аномалії, оскільки аномальні події мають деякі несподівані та зазвичай несприятливі наслідки для процесів, які нас цікавлять.

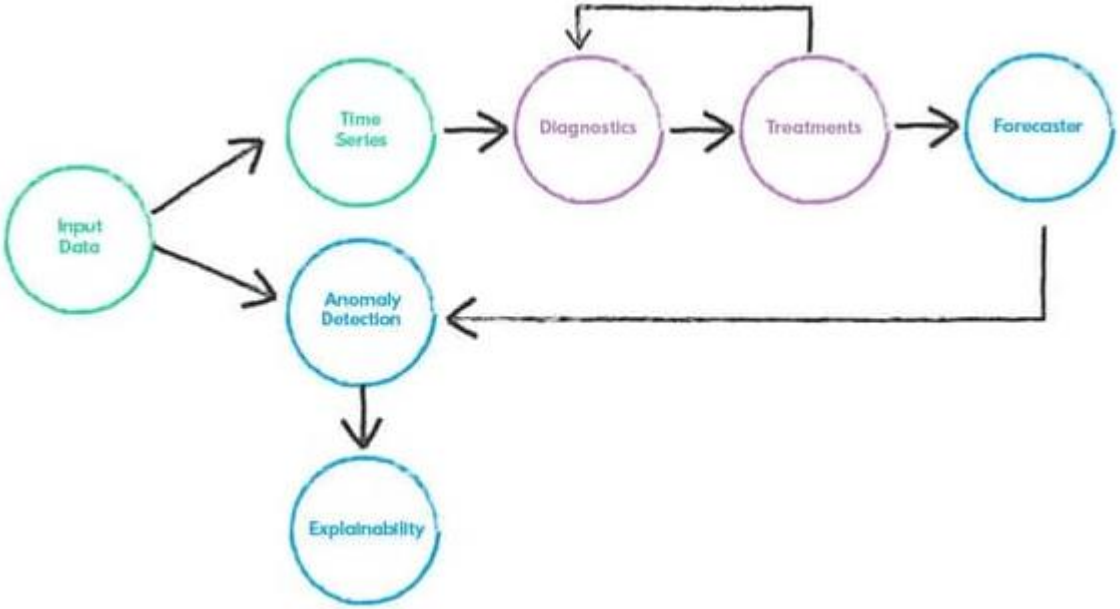


Рисунок 9.8 – Типовий робочий процес для виявлення мережевих аномалій

Виявлення аномалій можна використовувати для досягнення покращень у важливих сферах, таких як:

- Зменшення часу обороту для помилок застосування
- Забезпечення високої продуктивності рішення
- Виявлення загроз безпеці до їх виникнення

У галузі мобільних мереж визначення аномалій тісно пов’язане з бізнес-міркуваннями. Аномалія – це вибірка даних, яка вказує на подію або несправний (програмний/апаратний) компонент, що призвело до фінансових втрат, прямо чи опосередковано.

Це стосується багатьох підприємств, і це важливо у сфері телекомунікацій. Оскільки ми переходимо до 5G та Industry 4.0, зростає потреба в управлінні та моніторингу великої кількості даних, що надходять із складніших систем. Оператор не може отримати інформацію з коливань тисяч різних показників продуктивності, а виявлення аномалій у таких взаємопов’язаних процесах стає неможливим для візуалізації неозброєним оком навіть для експертів у галузі. Типовим прикладом аномалії може бути раптовий сплеск навантаження на базову станцію або збільшення кількості розривів викликів у мережі.

Завдяки останнім розробкам машинного навчання та штучного інтелекту зростає потреба в демократизації фреймворків виявлення аномалій для широкого кола користувачів. Підприємствам знадобиться, щоб їхні співробітники стали більш обізнаними з даними в різних аспектах їхньої роботи, починаючи від експертів у галузі, бізнес-аналітиків і розробників програмного забезпечення.

Для чого використовуються детектори аномалій?

Перш ніж розгортати систему виявлення аномалій, важливо встановити реалістичні очікування. Цього можна досягти шляхом визначення рамки, яка:

• Автоматично виявляти незвичні зміни в поведінці мережі;

• Прогнозувати великі збої зі 100% точністю;

• Забезпечте легкий для розуміння аналіз першопричини та розширений інтелектуальний аналіз безпеки, щоб постачальники послуг точно знали, як вирішити наявні проблеми.

Насправді аналіз поведінки мережі не такий простий:

• Жоден детектор аномалій не може надати 100% правильних відповідей так/ні. Хибнопозитивні та хибнонегативні результати завжди існуватимуть, і між ними є компроміси. Проте невідомі загрози безпеці в обхід традиційної безпеки можуть залишитися непомітними;

• Жоден детектор аномалій не може забезпечити 100% правильний аналіз першопричини, можливо, через низьке співвідношення сигнал/шум і кореляцію між показниками ефективності. Постачальникам послуг часто доводиться робити висновок про причинно-наслідковий зв’язок, поєднуючи результати виявлення аномалій мережевого трафіку зі своїми доменними або інституційними знаннями.

Додаткові проблеми ускладнюють це завдання:

• Обсяг даних для навчання та тестування моделі може бути обмеженим, і вони можуть бути не позначені (тобто ми не знаємо, які точки даних є аномаліями). Машинне навчання, глибоке навчання та інші складні алгоритми зазвичай вимагають великих обсягів даних, оскільки мережеві аномалії за визначенням статистично малоймовірні під час звичайного трафіку (тобто аномальна поведінка менш імовірна, ніж звичайна поведінка), набори даних часто незбалансовані (тобто існують більше випадків нормальної поведінки, ніж

аномальної поведінки або мережевих атак), що створює додаткові проблеми в моделях навчання, які точно визначають або передбачають підозрілу поведінку.

• Детектори аномалій можуть бути побудовані на динамічних системах із швидко зростаючою базою користувачів. У результаті детектори аномалій повинні адаптувати свою поведінку з часом, навчитися виявляти невідомі загрози безпеці, які неможливо виявити, перш ніж стати передбачуваними в міру розвитку базового IT-середовища.

Оскільки аномалії в інформаційних системах найчастіше свідчать про певні проломи або порушення безпеки, виявлення аномалій застосовувалося в різних галузях для вдосконалення ландшафту IT-безпеки та виявлення зловмисної поведінки, невідомого зловмисного програмного забезпечення, внутрішніх загроз, зловживань коло безпеки або мережевих атак для забезпечення профілактичних заходів. системи безпеки. Одним із випадків використання автоматичного виявлення в цьому контексті є сучасні кіберзагрози та інші загрози, що обходять традиційну безпеку.

Кібербезпека в цьому випадку гарантується за допомогою технології виявлення аномалій поведінки мережі (NBAD). Система аналізує підписи пакетів, щоб виявити підозрілу поведінку, виявити ще невідомі загрози безпеці, автоматично визначити аномалії в мережі та заблокувати вхідні/вихідні дані, які підривають внутрішні системи. NBAD також проводить безперервний моніторинг, щоб виявити зловмисну поведінку або тенденції у важливій інфраструктурі та критичних мережах.

Аналіз поведінки мережі з великими даними

Виявлення аномалій є важливою технікою для розпізнавання шахрайства, мережевих аномалій, підозрілих дій, зловмисної поведінки, мережевих вторгнень, сучасних кіберзагроз та інших незвичайних подій, які можуть мати велике значення, але їх важко виявити. Важливість виявлення аномалій полягає в тому, що цей процес перетворює дані в критично важливу інформацію та вказує на інформацію про проактивне виявлення в різних сферах застосування.

Виявлення аномалій, яке вирішує проблеми високого затемнення (ensionality) можна застосовувати в режимах онлайн або офлайн. В автономному режимі аномалії мережі виявляються в історичних наборах даних, відомих як «пакетна обробка». Це стосується функції «обсяг» великих даних. Навпаки, нові точки даних, відомі як «набір даних», постійно вводяться в онлайн-режимі, поки виявляються аномалії. Це стосується функції «швидкості» великих даних. Кілька існуючих опитувань і оглядів підкреслюють проблему високої розмірності для різних сфер, таких як машинне навчання та інтелектуальний аналіз даних.

Виявлення аномалій загалом представляє багато проблем у багатьох аспектах:

• Поняття стандартних даних дуже залежить від домену. Не існує універсальної процедури моделювання середніх даних. Концепція нормальності все ще суб’єктивна і її важко перевірити. Крім того, класи даних (нормальний/ненормальний) загалом є незбалансованими. А оскільки статистичні інструменти не застосовуються до кількох зразків, моделювати аномалії непросто через їхню рідкість.

• Межа між нормальними та ненормальними даними нечітка. Немає жодного правила для визначення балів у цій сірій зоні, окрім суб'єктивного судження. Крім того, дані можуть містити шум, що ускладнює розрізнення нормальних і ненормальних даних.

• Концепція нормальності розвивається з часом. Те, що очікується в певний проміжок часу, може стати аномальним у майбутньому. Також можуть з'явитися нові аномалії, які не містяться в моделі.

• Нечасто мати позначені дані для навчання моделі в багатьох областях.

Системи виявлення аномалій поведінки мережі (ADS)

Вибір підходу до виявлення аномалії залежить від використовуваних даних навчання та тестових даних. В даний час відомо три його види:

• Контрольоване виявлення використовується з повністю позначеними навчальними та тестовими наборами даних. Цей метод добре працює з незбалансованими класами та виконує маркування даних, оскільки аномалії добре відомі та вже позначені. Таким чином, він не застосовується у випадках, коли викиди ще не визначені.

• Напівконтрольоване виявлення використовує навчальні та тестові набори даних, але навчальні дані позбавлені аномалій. Цей підхід передбачає, що система визначить аномалію, коли дізнається стандартний набір даних і побачить його відхилення.

• Модель неконтрольованого машинного навчання для AD є найбільш гнучкою з трьох з точки зору надання системі міток і розмежування між навчальним і тестовим набором даних. Таким чином, система оцінює дані в наборі даних лише на основі характеристик одиниць, без будь-яких попередньо визначених значень нормальності.

Розширені системи безпеки на основі аналітичних даних

Хоча виявлення аномалій є суб'єктивним, було розроблено багато систем виявлення аномалій (ADS).

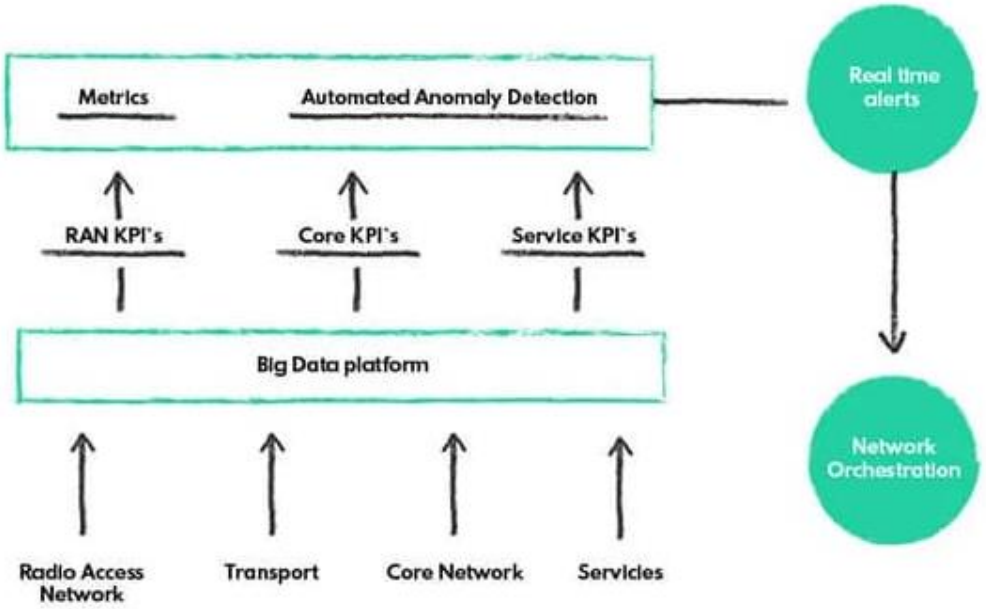


Рисунок 9.9 – Системи виявлення аномалій (ADS), методи та техніка

Деякі ADS дуже специфічні для домену і не можуть бути застосовані до інших доменів через обмеження (схеми вводу/виводу, використання ресурсів тощо). Інші ADS є дуже гнучкими і можуть використовуватися в багатьох різних дисциплінах. На відміну від традиційних рішень, нещодавно запропоновані рішення на основі штучного інтелекту (AI) є достатньо загальними для інтеграції в різні IT-середовища (тобто критичні мережі).

Загалом ADS збирає велику кількість даних. Він визначає аномальні точки на основі попередніх знань (експертні системи) або знань, виведених із даних (наприклад, рішень машинного навчання). Залежно від сфери застосування та надійності ADS, про виявлені аномалії можна або повідомити експерту для їх аналізу, або передати в іншу систему, яка автоматично виконує заходи пом'якшення. Ось найпопулярніші методи, які використовуються в галузі телекомунікацій:

Методи машинного навчання

На основі знань. Цей підхід базується на знаннях людини про домен. Він припускає, що шаблони аномалій відомі та що їх можна закодувати таким чином, щоб їх зрозуміла машина. Виходячи з цих припущень, створення та функціонування системи, що базується на знаннях (також називається експертною системою), передбачає три етапи:

- Експерт повинен вручну проаналізувати велику кількість даних і виявити аномалії.
- Шаблони аномалій реалізовані в автоматизованій системі.
- Система працює автоматично та виявляє аномалії в нових даних.

Регресія. Цей підхід заснований на принципі прогнозування. Аномалію можна визначити як розрив між реальністю та тим, що очікувалося. Система AD на основі регресії працює в такому режимі:

• По-перше, він оцінює дані, що надходять.

• Він кількісно визначає розрив між фактичним значенням і прогнозованим. Якщо розрив достатньо великий (вищий за попередньо визначене порогове значення), нова точка даних вважається аномалією.

Нейронні мережі можна використовувати для виконання регресії.

Класифікація. Цей підхід передбачає, що точки даних можна згрупувати в класи. Є два можливі способи перегляду аномалій:

- Як розкидані точки далеко від щільної центральної групи точок даних (двокласова класифікація) або
- Щільні групи даних відрізняються від груп стандартних даних (багатокласова класифікація).
- Методи класифікації припускають, що ми маємо навчальний набір даних із позначеними даними. У цьому підході процес виявлення аномалії можна розбити на такі етапи:
- Ми класифікуємо навчальні дані та визначаємо атрибути класифікації.
- Вивчаємо модель, використовуючи навчальні дані.
- Ми можемо класифікувати нові дані за допомогою моделі навчання.

Кластеризація. Кластеризація – це неконтрольована техніка групування схожих даних. Цей підхід передбачає, що аномалії є або точками, що не належать жодному кластеру, або належать до малих кластерів порівняно з великими та щільними кластерами нормальних даних. Алгоритми кластеризації мають дві фази:

1. Фаза навчання. Під час фази навчання точки даних групуються в кластери протягом великої кількості ітерацій до досягнення конвергенції або досягнення попередньо визначених критеріїв (максимальна кількість ітерацій тощо).

2. Фаза тестування. На фазі тестування нова точка даних призначається найближчим кластерам на основі вимірювання відстані/подібності, що використовується під час навчання.

Тематичні дослідження виявлення мережевих аномалій

1. Програма виявлення аномалій на хмарній платформі Google (GCP)

Існує багато програм, розроблених постачальниками телекомунікаційних компаній і CSP, які фіксують захоплюючі інциденти в мобільних мережах, і одна з найважливіших стосується виявлення аномалій. Ось один із прикладів послуги, запущеної технологією Nokia Bell Labs і розгорнутої в пан'європейській мережі Vodafone.

Продукт швидко виявляє та усуває несправності, такі як перевантаження мобільного сайту та перешкоди, а також неочікувана затримка, що впливає на якість обслуговування клієнтів. Vodafone очікує, що близько 80 відсотків аномальних проблем мобільної мережі та вимог до пропускної здатності будуть автоматично виявлені та усунені за допомогою служби виявлення аномалій.

Програма виявлення аномалій

Очікується, що розроблений додаток Anomaly Detection забезпечить Vodafone значні переваги в галузі радіозв'язку та є частиною ширшої мети Vodafone щодо ефективного використання мережі та витрат.

Vodafone використовує варіант виявлення аномалій для підтримки мережевого планування та оптимізації з планом розширення до мережевих операцій. Це перший крок до досягнення повної автоматизації управління життєвим циклом мережі. Платформа Neuron працює в GCP для відтворення та зберігання даних RAN, а алгоритми Nucleus використовуються для розпізнавання шаблонів, кластеризації та класифікації на основі машинного навчання.

Nokia розробила програму виявлення аномалій, яка буде використовуватися як послуга. Він поділив своє бачення з Vodafone і продемонстрував гнучкість і відкритість для постійної співпраці (деякі основні вимоги Vodafone). Nokia працювала на рівноправних засадах і надала вихідний код алгоритму, що підвищило довіру в рамках партнерства.

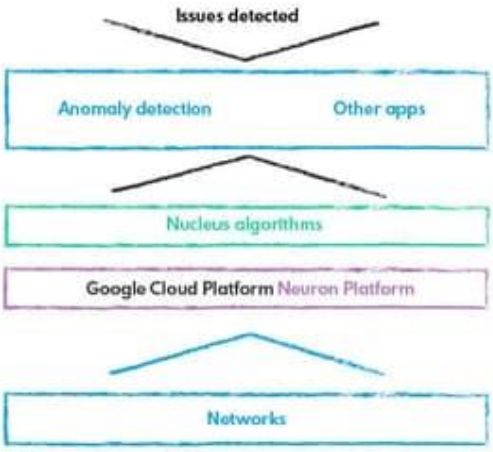


Рисунок 9.10 – Система виявлення аномалій на базі Google Cloud Platform

Ключові переваги

Просте розгортання нових додатків серед операційних компаній. GCP дозволяє Vodafone зосередитися на варіантах використання та бізнес-результатах. Це дозволило Vodafone прийняти метод «розробити один раз, розгорнути багато разів», що призвело до зменшення зусиль на 60–70%, що дозволило Vodafone швидко розгортати програми на різних ринках.

Спільне інноваційне партнерство та глобальні практики доступу. Рівноправне партнерство з Nokia разом із спільним баченням і співпрацею з GCP також забезпечує міцну основу для майбутньої співпраці. Між Vodafone і Nokia існує чіткий розподіл обов'язків: Vodafone надає мережеві дані, а Nokia створює моделі та програми.

Виявлення аномалій – це інструмент для ефективного повсякденного планування мережі, оптимізації та операцій. Виявлення аномалій забезпечує найбільш значні переваги з точки зору автоматизації аналізу першопричини (покращення ефективності роботи на 25–30%). Він використовує машинне навчання для автоматичного виявлення таких проблем, як помилки встановлення виклику.

2. Система виявлення аномалій для створення прототипів

Один із проектів Глобального прискорювача штучного інтелекту Global Artificial Intelligence Accelerator (GAIA) компанії Ericsson має на меті зробити інструменти ШІ більш доступними для виявлення аномалій. Він представляє структуру E-ADF.

Ідея, що лежить в основі E-ADF, виникла на основі багатьох проектів виявлення аномалій, які надходять у конвеєр GAIA, багато з яких потрібно починати з нуля. Співпраця між дослідниками даних, які працюють над цими проектами, призвела до використання багаторазових компонентів із коду та створення ADF.

Проект отримав користь від внеску найкращих методів виявлення аномалій у GAIA та Ericsson Research. Початкова мета створити ресурс багаторазового використання для науковців з обробки даних перетворилася на

план створення простої у використанні платформи штучного інтелекту як для науковців із обробки даних, так і для інших спеціалістів.

E-ADF сприяє швидшому створенню прототипів для випадків використання виявлення аномалій, пропонуючи свою бібліотеку алгоритмів для виявлення аномалій і часових рядів із такими функціями, як візуалізація, лікування та діагностика. Включені алгоритми дають змогу користувачам обробляти як одновимірні, так і багатовимірні дані, такі функції, як рухоме вікно та сегментація, пояснювач детектора, який допомагає знайти першопричину певних аномалій, і конвеєри.

Мережі майбутнього зазнають і продовжуватимуть зазнавати інцидентів. Для цього є багато згаданих причин, але відповідей також стає все більше, оскільки в гру входять рішення на основі ШІ. Незважаючи на те, що багато операторів мобільних мереж досі використовують традиційні методи пошуку та відстеження інцидентів у мережі, настав час для штучного інтелекту допомогти у вирішенні критичних проблем.

Однак перед розгортанням будь-яких систем виявлення аномалій із штучним інтелектом важливо встановити реалістичні очікування для системи:

• виявлення незвичайних змін, мережевих атак або внутрішніх загроз,

• прогнозувати великі невдачі зі 100%

• забезпечити легкий для розуміння аналіз першопричини, щоб CSP точно знав, що і як вирішити проблему.

Після розробки моделей виявлення аномалій наступним кроком буде їх інтеграція у виробничу систему. Це може спричинити проблеми з розробкою даних, оскільки аномалії слід виявляти в режимі реального часу, постійно, з потенційно великим обсягом потокових даних.

9.3. Кібербезпека і медицина

Напрями застосування ШІ у кібербезпеці і медиціні

1. Безпека мережі рис. 9.11.

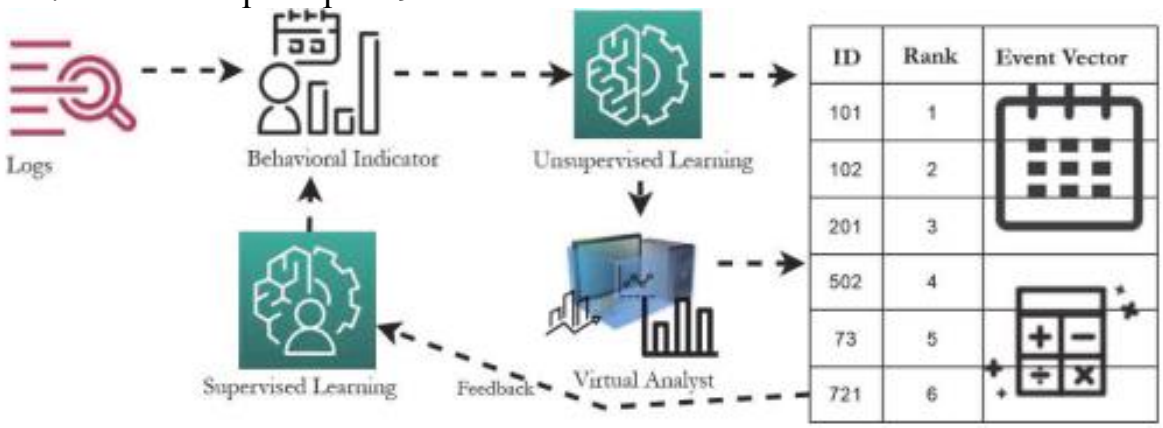


Рисунок 9.11 – Система кібербезпеки на основі ШІ

2. Швидший час відповіді

Ключова перевага штучного інтелекту в кібербезпеці полягає в тому, що штучний інтелект може негайно виявляти будь-яку аномальну поведінку та підозрювані проблеми та запобігати потенційній кіберзагрозі для систем

охорони здоров’я. Здатність виявляти загрозу та швидко реагувати на неї може покращити систему безпеки будь-якої організації, яка вимагає ресурсів та репутації. Три важливі стратегії покращення виявлення та реагування до того, як загрози завдадуть шкоди критично важливій системі охорони здоров’я, – це керована служба безпеки, просування вперед за допомогою ШІ та централізація реагування.

3. Виявлення та запобігання фішингу.

Фішингові атаки є однією з найпоширеніших проблем безпеки для окремих осіб і компаній у захисті своєї інформації, коли зловмисники намагаються передати свої корисні дані за допомогою фішингової атаки. Штучний інтелект і машинне навчання можуть взяти на себе важливу роль у запобіганні фішинговим атакам і відверненні від них. Комп’ютерне інтелектуальне машинне навчання може розпізнавати та стежити за понад 10 000 динамічних джерел фішингу. Крім того, AI-машинне навчання працює над фільтрацією небезпек фішингу з усього світу. Фішингові атаки можуть мати кілька різних цілей, включаючи доставку зловмисного програмного забезпечення, крадіжку грошей і викрадення облікових даних. Більшість фішингових шахраїв спрямовані на викрадення особистої інформації. Немає обмежень у розумінні спроб фішингу на певній геологічній території. Комп’ютерний інтелект зробив можливим швидко відокремити фальшивий сайт від справжнього.

4. Безпечна автентифікація.

Забезпечення безпеки або автентифікація стала ключовим питанням у бездротових мережах через їх життєво важливу роль у підтримці численних послуг. Фізично впізнаваний доказ, у якому штучний інтелект досліджує різні елементи безпеки, щоб відрізнити користувача, може бути основним способом перевірки безпеки. Смартфон може використовувати сканер для отримання унікальних відбитків пальців і виразу обличчя, щоб забезпечити безпечний вхід користувача. Додаток для смартфона перевіряє відбиток пальця та вираз обличчя, щоб визначити, чи правильний вхід. Крім того, техніка штучного інтелекту може досліджувати різні функції, щоб перевірити автентифікацію користувача та дозволити користувачеві отримати доступ до інформації з будь-якого пристрою.

5. Аналітика поведінки.

Одним із важливих застосувань штучного інтелекту в кібербезпеці є його здатність аналізувати поведінку. Це означає, що обчислення машинного навчання можуть навчитися та стати прикладом вашої поведінки, аналізуючи, як ви використовуєте свій гаджет і онлайн-етапи. Читати про використання штучного інтелекту в охороні здоров’я, як-от дослідження ДНК/генома, справді захоплює. Люди залучені до поведінкової частини кібербезпеки. Крім того, поведінка машини відіграє значну роль у кіберподіях. ШІ змінює наш спосіб життя, в тому числі те, як ми живемо, працюємо та граємо. З огляду на те, що все більше даних про охорону здоров’я збираються за допомогою мультисенсорних систем і медичних інструментів і обробляються, передбачення та поведінкова аналітика дозволяють генерувати розуміння та вживати необхідних заходів.

10 ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В ТЕЛЕКОМУНІКАЦІЯХ

Штучний інтелект (ШІ) помітно розвивається приблизно з 2010 року та застосовувався на телекомунікаційному ринку вже більше десяти років. Крім того факту, що телекомунікаційні інфраструктури є життєво важливими для суспільства, все більше і більше програм залежать від добре функціонуючих, надійних і завжди доступних телекомунікаційних послуг. Напрями застосування:

- Штучний інтелект використовується в телекомунікаційному секторі вже більше десятиліття, причому програми зосереджені на оптимізації радіосигналів, управлінні живленням і оцінці передачі.
- ШІ в телекомунікаціях може забезпечувати швидше та ефективніше прийняття рішень, ефективніше використовувати експертні знання та ефективно вирішувати повторювані завдання.
- Програми штучного інтелекту можуть працювати автономно завдяки автономному навчанню та діям із сценаріями, починаючи від замкнутих систем до взаємодії людини в циклі.
- Проблеми впровадження штучного інтелекту в телекомунікаційному секторі включають технічну інтеграцію, брак технічного досвіду та роботу з неструктурованими даними.

10.1. Напрями застосування ШІ в телекомунікаціях

Більшість сучасних додатків штучного інтелекту зосереджені на покращенні конкретних параметрів. Це строго визначені програми, такі як:

1. Оптимізація параметрів радіосигналу

Зараз машинне навчання (ML) використовується для оптимізації потоку даних до та з базової станції (BTS) у мобільній мережі. Відстань до користувачів, підключені користувачі та певні фактори навколишнього середовища визначають параметри радіозв’язку. Вони, у свою чергу, визначають максимальну кількість даних, які можуть бути передані на кількість спектру за одиницю часу. Перешкоди також відіграють певну роль: радіоресурси можуть координуватися між мікро- та макросотами. Щоб максимізувати ефективність, алгоритми використовуються для динамічного визначення того, яка частина спектра повинна використовуватися для якого користувача та з якими параметрами. Параметри цих алгоритмів можна «налаштувати» за допомогою ШІ.

2. Управління живленням

Методи машинного навчання використовуються для досягнення енергозбереження в живих мобільних мережах. Ґрунтуючись на метеорологічних даних, кількості користувачів і їхньому положенні, антени активно коригують свою діаграму спрямованості, напрямок і потужність відповідно до вимог. Це призводить до економії енергії, наприклад, вночі, коли попит на дані є відносно низьким, і до більш ефективного використання базових

станцій, оскільки більша площа може працювати в точках налаштування, де потреба в потужності неоднакова.

3. Оцінка якості передачі

За допомогою оптичних з'єднань сигнал може порушуватися або перериватися, що може призвести до постійного виходу з ладу обладнання. Машинне навчання використовується, щоб заздалегідь оцінити, наскільки добре передача буде працювати через з'єднання. Він обчислює найкращий шлях на основі таких речей, як довжина кабелю, інші сигнали всередині кабелю та вік обладнання. Трафік маршрутизується на основі цієї оцінки. Також можливо, що такі алгоритми використовуються в бездротових мережах, наприклад, для визначення того, наскільки використовується корекція помилок або надмірність (наприклад, повторна передача). Експертні системи та алгоритми машинного навчання є двома методами ШІ, які широко використовуються в телекомунікаційному секторі, тоді як ML і розподілений штучний інтелект є двома методами ШІ, які є найбільш перспективними на майбутнє.

10.2. ШІ та машинне навчання

AI і ML руйнують і трансформують бізнес. Телекомунікаційні компанії можуть використовувати ці технології, щоб покращити утримання клієнтів, увімкнути самообслуговування, покращити технічне обслуговування обладнання та одночасно знизити операційні витрати.

Телекомунікаційна галузь пливе на хвилі технологічної революції та цифрової трансформації, щоб пропонувати своїм споживачам більший спектр послуг. Однак споживачі в сучасному цифровому світі не будуть задоволені звичайними продуктами та послугами – вони також вимагають кращої якості послуг і більш чуйних постачальників послуг. Статистика на основі даних, заснована на рішеннях на основі штучного інтелекту та машинного навчання, може допомогти провайдерам зв'язку виправдати ці очікування.

Автономне навчання та дія

Програми штучного інтелекту часто беруть на себе завдання, які зазвичай виконують люди, або підтримують людей. Це означає, що ці системи мають певний ступінь автономності. Існує дві форми незалежності: автономне навчання та автономна дія.

Автономне навчання

Сучасну модель ШІ було розроблено на основі великої кількості (історичних) даних. Алгоритм «вивчає» бажані результати з цих даних за допомогою певних вхідних даних. Є кілька способів сформувати це навчання або «навчання»:

Офлайн-навчання – модель навчається один раз або час від часу на основі статичного набору даних. І модель, і дані, що використовуються, можуть бути протестовані та підтверджені до того, як модель буде запущена у виробництво;

Онлайн-навчання – модель навчається як і офлайн-навчання, а потім періодично перенавчається на основі нових даних;

Безперервне навчання – модель постійно оновлюється за допомогою вхідних даних. На відміну від онлайн-навчання, більше немає різних «версій»

моделі: кожен запит на висновок потенційно безпосередньо впливає на наступне рішення ШІ.

Автономна дія

Програми штучного інтелекту в телекомунікаційній галузі можна застосовувати різними способами. До найбільш використовуваних відносяться:

Closed-loop scenario (Сценарій замкнутого циклу) – у цьому сценарії система ШІ виконує дії безпосередньо. Єдина дія, якої люди можуть досягти, це вимкнути систему, наприклад, програмне забезпечення для розпізнавання мовлення.

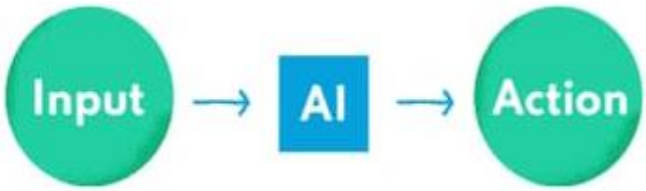


Рисунок 10.1 – Сценарій замкнутого циклу

Open-loop scenario (Сценарій відкритого циклу) – роль системи AI полягає в наданні підтримки. ШІ пропонує людині результат, і на основі цього результату людина може діяти. У цьому випадку люди можуть відхилитися від поради та/або перевірити цю пораду на основі іншої інформації. Приклад – експертні системи, які допомагають лікарям сформувати діагноз.

Рисунок 10.2 – Сценарій відкритого циклу

У **rule-constrained closed-loop scenario** (сценарії замкнутого циклу), обмеженого правилами, система ШІ може виконувати прямі дії, але обмежена певними «жорсткими» правилами. Порушення правил призводить до вимкнення системи або бездіяльності. Прикладом цієї стратегії є автономні транспортні засоби. Вони часто оснащені різними правилами «відмовостійкості», які забезпечують аварійну зупинку автомобіля в небезпечних ситуаціях;

Рисунок 10.3 – Сценарій, обмежений правилами

У **human in the loop scenario** (сценарії «Людина в циклі») ШІ може виконувати дії безпосередньо, але люди можуть зупиняти або коригувати ці дії, якщо це необхідно. Приклад: автономні транспортні засоби, де людям доводиться тримати руки на кермі;

Multiple AI systems in the loop (Кілька систем штучного інтелекту в циклі) – одна або кілька інших систем штучного інтелекту контролюють систему штучного інтелекту, яка виконує дії. Керуюча модель ШІ може переглядати початкові вхідні дані та рішення ШІ та оцінювати, чи правильне це рішення.

Додана вартість ШІ в телекомунікаційній галузі

Використання штучного інтелекту в телекомунікаціях може допомогти вирішити кілька складних і іноді тривалих проблемних питань і в той же час принести масу доданої вартості як споживачам, так і операторам. Останній завжди збирав значні обсяги телеметрії та статистичних даних про використання послуг, але більшість з них ніколи не використовувалися значущим чином через відсутність потрібного програмного забезпечення.

За допомогою штучного інтелекту цей величезний масив даних, які раніше не використовувалися, можна перетворити на благодатний ґрунт для розвитку нових послуг, покращення якості існуючих, виведення клієнтського досвіду на новий рівень та оптимізації бізнес-операцій. Згідно з порівняно недавніми дослідженнями, до 2025 року штучний інтелект у телекомунікаційних компаніях принесе майже 11 мільярдів доларів – приголомшлива сума, яка, ймовірно, продовжуватиме зростати в міру того, як буде розширюватися сфера застосування ШІ. На основі оцінених звітів – Глобальний штучний інтелект на телекомунікаційному ринку. Розмір, стан і прогноз на 2021–2027 рр. Прогнозоване зростання світового ринку штучного інтелекту в галузі телекомунікацій Рис.1.

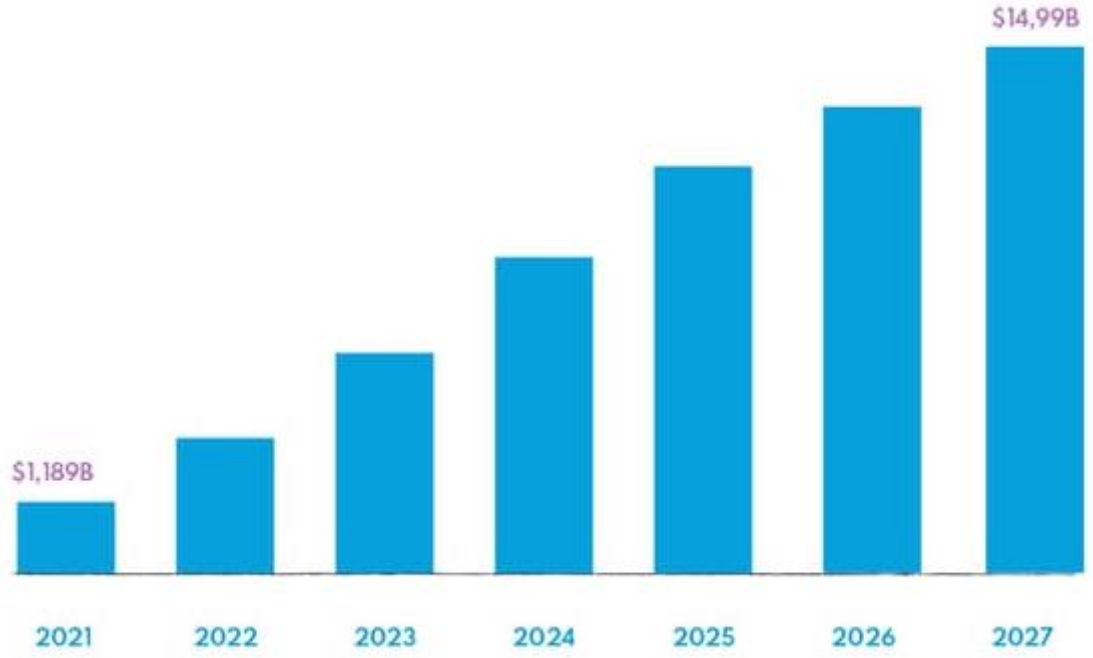


Рисунок 10.4 – – Прогнозоване зростання світового ринку штучного інтелекту в телекомунікаціях.

ШІ зазвичай надає кілька переваг і можливостей:

ШІ може приймати швидше, а часом і кращі рішення, ніж люди

У той час як людині іноді потрібно від кількох секунд до хвилин, щоб прийняти рішення, модель машинного навчання часто може обробити тисячі елементів даних за частки секунди.

Наприклад, AI для виявлення шахрайства може відстежувати тисячі транзакцій кредитних карток у режимі реального часу та блокувати потенційно шахрайські транзакції.

Завдяки штучному інтелекту рідкісні експертні знання можна використовувати ефективніше

Люди часто проходять навчання протягом кількох років, перш ніж вийти на ринок праці. Людина ще не буде на вершині своєї гри, навіть через кілька років. За допомогою ML знання експерта можна перевести в модель, і, таким чином, ці знання можна застосовувати ширше.

Наприклад, Google розробив систему штучного інтелекту, яка може ідентифікувати пухлини за допомогою КТ так само добре, як і досвідчений радіаційний онколог. Таке застосування штучного інтелекту може забезпечити недостатні експертні знання в конкретних сценаріях і тим самим звільнити людей-експертів для більш ефективного виконання іншої роботи.

Штучний інтелект добре справляється з повторюваними завданнями

Люди часто сприймають повторювані завдання як нудні та неприємні. Однак, якщо ці завдання добре сформульовані, структура ШІ ідеально підходить для їх виконання. ШІ не потрібно спати, відпочивати чи робити перерви, тому що він не буде нудьгувати чи втомлюватися. Повторювані завдання також добре піддаються штучному інтелекту, тому що (якщо люди зараз виконують цю роботу) можливо, достатньо великих даних для навчання ШІ.

Проблеми використання AI та ML для телекомунікаційних компаній

Незважаючи на те, що глобальний ШІ на телекомунікаційному ринку поширюється, його впровадження все ще може бути складним для багатьох компаній. Окрім неможливості розпізнати потребу в штучному інтелекті чи визначити відповідні бізнес-випадки використання, найпоширенішими проблемами впровадження штучного інтелекту в телекомунікаційному секторі є:

Технічна інтеграція – хоча старі застарілі системи є однією з найпоширеніших причин невдачі багатьох інтеграцій штучного інтелекту, перед початком будь-якого проекту штучного інтелекту можна підготуватися до кількох речей:

Створити єдину базу даних, де будуть зберігатися всі необхідні системі дані;

Використовуйте озера даних, а також крайові або хмарні обчислення, щоб усунути будь-які проблеми, які можуть виникнути під час зберігання великих обсягів даних;

Не соромтеся повністю оновити процес введення та зберігання даних, якщо ви помітили, що зібрані дані розрізнені або неструктуровані;

Переконайтеся, що у вас є необхідне обладнання та програмне забезпечення для роботи з новою системою.

1. Відсутність технічного досвіду – ШІ є відносно новою технологією. З обмеженими місцевими талантами створення внутрішньої команди може зайняти значну кількість часу та дати незначний результат. Кращим варіантом є пошук технічного партнера для впровадження штучного інтелекту в телекомунікації. Однак пошук постачальника, який має достатньо компетенції та досвіду для успішного створення системи ШІ, сам по собі може бути проблемою. Крім того, впровадження штучного інтелекту може бути досить дорогим, тому розпочати свій проект із правильним партнером має вирішальне значення.

2. Неструктуровані дані. Впровадження системи штучного інтелекту без доступу до відповідних даних є марною спробою. Багато організацій стикаються зі збором даних через кілька типових проблем:

Фрагментовані дані – інформація збирається та зберігається різними системами без єдиної уніфікованої бази даних, звідки до неї можна отримати доступ; Неструктуровані дані – величезна маса некатегоризованих даних без будь-якого контексту чи пояснення того, з чим вони пов’язані, не дуже корисна для будь-якого алгоритму ШІ; Неповні дані – використання даних із відсутніми компонентами може призвести до непослідовного або неправильного навчання системою ШІ.

Оскільки алгоритми штучного інтелекту вимагають чистих, добре структурованих даних, близько 80% часу будь-якого проекту ML присвячено ETL (вилучення, перетворення, завантаження) та очищення даних. Тому важливо створити відповідну інженерну екосистему великих даних для збору, інтеграції, зберігання й обробки даних із численних ізольованих джерел даних.

AI for Telco – найпоширеніші випадки використання

ШІ допоміг телекомунікаційному сектору переосмислити задоволеність клієнтів, відкривши нові можливості та ускладнивши бізнес-моделі. Ось деякі способи, якими штучний інтелект вже зробив внесок у телекомунікаційну галузь:

1. Прогнозне технічне обслуговування

AI та ML дозволили телекомунікаційній індустрії отримувати цінні відомості про бізнес. Оскільки телекомунікаційні компанії мають величезну кількість великих даних, штучний інтелект може використовувати їх для прийняття ефективних та ефективних рішень шляхом сегментації клієнтів, прогнозування життєвої цінності споживача та надання рекомендацій щодо купівлі.

Прогностична аналітика, знаходячи закономірності в історичних даних, може точно передбачати й попереджати про можливі апаратні збої. Крім того, створені алгоритми та моделі науки про дані можуть визначити причину кожного збою, що дає змогу боротися з проблемою в її корені. Профілактичне технічне обслуговування дозволяє телекомунікаційним компаніям бути дуже активними в обслуговуванні свого обладнання, вирішувати проблеми до їх виникнення та мінімізувати запити на підтримку. Цей проактивний підхід покращує загальний досвід клієнтів.

2. Покращена оптимізація мережі

Програми штучного інтелекту в галузі телекомунікацій допомагають постачальникам послуг зв'язку (CSP) створювати мережі з самооптимізацією (SON), щоб підвищити задоволеність клієнтів, знизити витрати на обслуговування клієнтів, запобігти збоям і підтримувати певну якість мережі. Такі мережі автоматично контролюються алгоритмами ШІ, які виявляють і точно прогнозують аномалії мережі. Крім того, вони можуть проактивно оптимізувати та переналаштувати мережу для забезпечення вищої якості послуг.

3. Аномалії мережі

Виявлення аномалій на основі штучного інтелекту ефективно доповнює та автоматизує раннє виявлення, прогнози та прийняття рішень в операційних і бізнес-процесах, де люди не можуть мати справу з обсягом або швидкістю даних. Збільшення загального часу виявлення незмінно призводить до швидшого вирішення інцидентів. Таким чином, це призводить до зменшення витрат, пов'язаних із збоями, і допомагає запобігти втраті прибутку та впливу на бренд.

Крім того, рішення для виявлення аномалій на основі AI/ML можуть аналізувати численні виміри джерел даних, розглядаючи KPI на рівні стільника, абонента та пристрою, відстежуючи несправності мережевого обладнання та корелюючи сповіщення між доменами для зменшення шуму та аналізу першопричини . Це дає інженерам прозоре уявлення про продуктивність мережі та послуг, а також про взаємодію з абонентами – у будь-який момент часу, а також дозволяє їм використовувати дані та вчасно прогнозувати аномалії мережі.

4. Роботизована автоматизація процесів (RPA)

Роботизована автоматизація процесів — це форма цифрової трансформації, яка спирається на впровадження ШІ. Сектор телекомунікацій може використовувати RPA та обробку природної мови (NLP) для автоматизації введення даних, обробки замовлень, виставлення рахунків та інших бек-офісних процесів, які потребують багато часу та ручної роботи. Це звільняє час співробітників, дозволяє їм зосередитися на більш важливих завданнях і зменшує кількість помилок, до яких схильна ручна праця. Завдяки роботизованим рішенням автоматизації процесів офіс працює більш гладко, співробітники більш продуктивні, а клієнти отримують безпомилкове обслуговування.

5. Запобігання шахрайству

Одна з речей, які штучний інтелект у сфері телекомунікацій може робити надзвичайно добре, це виявлення та запобігання шахрайству. Аналітичні системи боротьби з шахрайством можуть виявляти підозрілі моделі поведінки та негайно блокувати додаткові служби або облікові записи користувачів, обробляючи журнали викликів і передачу даних у режимі реального часу. Додавання ML дозволяє таким системам бути ще точнішими та швидшими.

Реальні приклади з життя

1. Nokia

Nokia випустила екосистему AVA Telco AI, яка надає рішення штучного інтелекту, які можна доставляти через хмару за допомогою Microsoft Azure або інших загальнодоступних опцій. Рішення дозволяють постачальникам послуг

зв’язку автоматизувати керування мережею, планування пропускної здатності та забезпечення обслуговування, скорочуючи операційні витрати, підвищуючи гнучкість і підвищуючи якість обслуговування абонентів.

Nokia AVA для аналітики, аналітики та портфоліо AI. На основі маркетингових матеріалів Nokia AVA.

2. Vodafone

Британський телекомунікаційний гігант Vodafone Group запустив програму-помічника під назвою TOBi для управління обслуговуванням клієнтів, інтелектуального віртуального помічника, здатного підтримувати користувачів у вирішенні проблем, управлінні підписками та придбанні нового обладнання та послуг.

3. Deutsche Telecom

Deutsche Telekom робить значні інвестиції в ШІ на різних рівнях. Від чат-бота на основі ШІ під назвою Tinka, здатного надавати понад 1500 відповідей на запитання клієнтів за допомогою інтерактивної голосової відповіді, до інтелектуальних інструментів бізнес-планування, цей CSP активно вбудовує елементи ШІ та науки про дані у свою інфраструктуру та портфель послуг.

Майбутнє ШІ для телекомунікаційних компаній

Хмара, 5G та штучний інтелект, когнітивні обчислювальні технології, які взаємодіють із інформацією споживачів, дозволили відповісти на широкий спектр питань мовою клієнта. Однак у майбутньому, коли компаніям буде зручно передавати інформацію про клієнтів машинам, люди-агенти з обслуговування клієнтів можуть відійти в минуле, дозволяючи клієнтам взаємодіяти з віртуальними помічниками та запобіжниками.

Також передбачається, що ШІ перейде від роботи з інформацією до прогнозування поведінки споживачів і впливу на бізнес-рішення. Це має знизити витрати та покращити взаємодію з клієнтами, підвищивши їх життєву цінність.

Спектр потенційних застосувань штучного інтелекту в телекомунікаціях і штучному інтелекті є напрочуд широким. Немає сумніву, що ключові гравці ринку бачитимуть все більш інтелектуальні системи автоматизації, які розгортаються, щоб оптимізувати повсякденні операції та забезпечувати більшу цінність для клієнтів.

Успіх телекомунікаційних компаній, які розпочинають шлях цифрової трансформації, залежатиме від їхньої здатності якомога раніше ефективно використовувати ШІ та розробляти відповідне програмне забезпечення. Завдяки зібраним за допомогою когнітивних технологій даним, достовірним уявленням і ручному досвіду можливо немає обмежень у досягненні ШІ.

10.3. 5G і Штучний інтелект

5G і штучний інтелект – це дві найбільш революційні технології, які бачив світ за останні десятиліття. Хоча кожен окремо революціонізує індустрію та створює нові враження, поєднання цих двох компонентів сприяє технологічним інноваціям завтрашнього дня.

Ми всі знаємо, що штучний інтелект — це не лише захоплююча технологія, яка покращує точність і прогнозування різноманітних проблем, але й, зрештою,

необхідна для вилучення інтелекту з великої кількості даних, створених у сучасних мобільних мережах. покращення якості мережі.

Використовуючи мережеві дані, створені мобільними пристроями, мозок штучного інтелекту можна додатково використовувати для аналізу, мережевого планування, прогнозів і гарантування здорової роботи мережі.

AI також допомагає будувати стратегію керування даними шляхом автоматичної оптимізації продуктивності мережі.

Можливість штучного інтелекту дозволяє мобільним мережам виявляти такі проблеми, як збої в обслуговуванні або поломки на заводі. Потім це діагностується та виправляється автоматично.

Згодом штучний інтелект також зможе передбачати проблеми ще до їх виникнення. Крім того, AI може допомогти телекомунікаційним компаніям розробити нові послуги 5G, аналізуючи дані в режимі реального часу, щоб переконатися, що мережевих ресурсів достатньо, і вказати, де потрібно більше ресурсів.

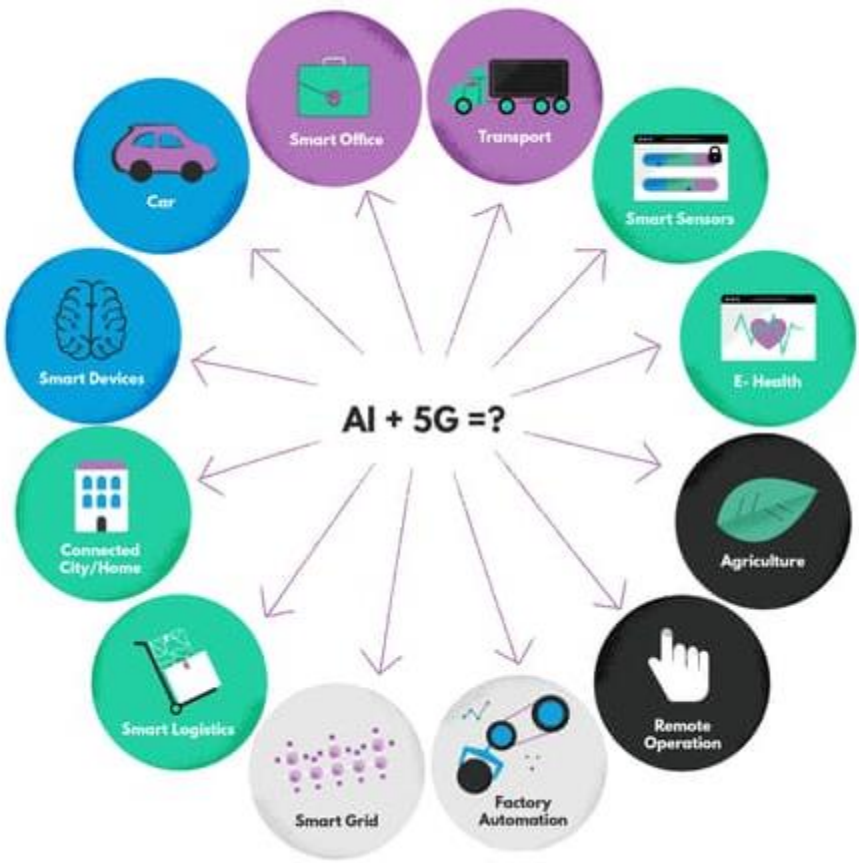


Рисунок 10.5 – Програми ШІ з 5G

Візьмемо для прикладу безпілотні автомобілі. Цим транспортним засобам потрібно буде зрозуміти великі обсяги даних від тисяч датчиків, наприклад, чи є об’єкт попереду людиною чи сміттям, і це потрібно робити безперервно та за лічені частки секунди. У той же час існує багато інших даних, таких як продуктивність або прогнозне обслуговування, які можуть зберігатися в централізованій хмарі.

Оскільки деякі програми вимагатимуть низької затримки, для безперебійної роботи мережевим адміністраторам знадобиться можливість

установлювати пріоритети для потоків трафіку. Нарізка мережі розглядається як одне з рішень. У нарізці мережі одна спільна фізична мережа має декілька віртуалізованих мереж, що працюють поверх неї.

Це дозволить виробнику оплачувати зріз мережі з гарантованою затримкою та надійністю підключення розумних машин і обладнання. Фрагменти необхідно налаштовувати вручну, що стосується поточного рівня техніки. AI може допомогти в цьому, оптимізуючи продуктивність мережі для маршрутизації трафіку на основі потреб пристрою.

AIoT і 5G

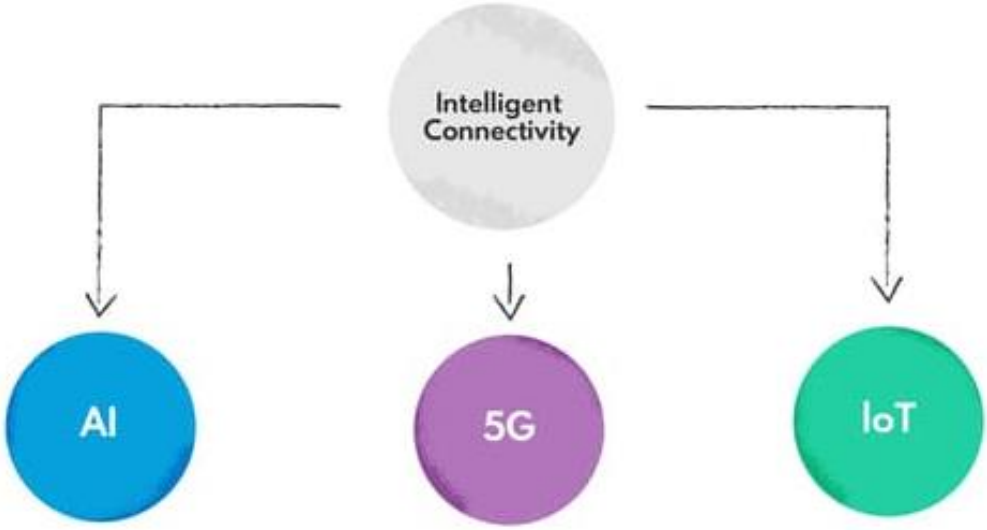


Рисунок 10.6 – 5G + IoT + AI = інтелектуальне підключення

Розвиваючи конвергенцію штучного інтелекту та Інтернету речей ще на один крок вперед, деякі компанії створили термін AIoT5G для позначення конвергенції штучного інтелекту, Інтернету речей і 5G. Перетин цих технологій привабить інновації, які створять подальший прогрес у різних галузевих вертикалях та інших технологіях, таких як робототехніка та віртуальна реальність (VR).

«Розумні міста» є одним із ринків, які можуть надати значні можливості для інтеграції технологій для підтримки високоналаштованих, але масштабованих послуг. Завдяки використанню взаємозалежних можливостей AIoT5G буде створена та підтримувана петля позитивного зворотного зв’язку.

Штучний інтелект працює в поєднанні з Інтернетом речей, щоб істотно покращити ланцюги поставок Smart City. Мегаполісні ланцюги поставок являють собою складні системи організацій, людей, діяльності, інформації та ресурсів, залучених до переміщення продукту чи послуги від постачальника до клієнта.

Розумний Інтернет речей

IoT вже є ключовим прискорювачем для штучного інтелекту. Наразі надає можливості створювати масивні набори даних, які можуть використовувати машинне навчання та алгоритми ШІ, перетворюючи дані на ідеї.

Розумні міста

Сучасні міста стали місцями, де традиційні мережі та послуги стають більш ефективними за допомогою цифрових і телекомунікаційних технологій на благо їхніх жителів і бізнесу.

Розповсюдження та доступність нових технологій необхідні для перетворення міста на Розумне місто (він же Інтелектуальне місто), сприяючи досягненню високого рівня сталого міського розвитку та покращенню якості життя його громадян.

Інтелектуальні міста використовують Інтернет речей (IoT) для збору даних у реальному часі, щоб краще розуміти, як змінюються моделі попиту, і реагувати швидшими та дешевшими рішеннями.

Загалом, екосистеми цифрових міст розроблені для роботи на основі ІКТ, які з'єднують кілька виділених мереж мобільних пристроїв, датчиків, підключених автомобілів, побутової техніки, комунікаційних шлюзів і центрів обробки даних.

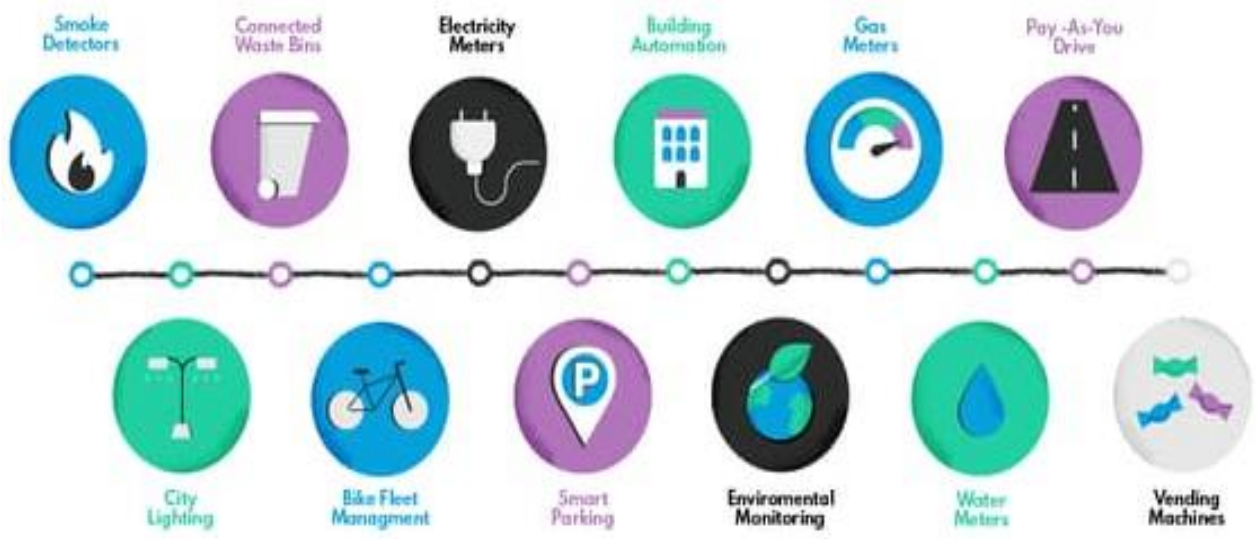


Рисунок 10.7 – Екосистема цифрового міста 5G

Місто як платформа

Міста та муніципалітети прагнуть бути розумними та переходити на цифрові технології як для внутрішніх робочих процесів, так і для нових способів взаємодії зі своїми громадянами.

З тих пір, як вони почали інвестувати в цифрові пристрої, вони покладалися на комп'ютери, щоб обробляти великі обсяги даних про податки, витрати на ремонт доріг тощо, і викидати паперові звіти для інтерпретації менеджерів, таким чином створивши острівці програмного забезпечення, які не можуть спілкуватися з один одного. Поліцейські департаменти мали один вид програмного забезпечення. У бібліотек був інший. У різних міських установах були різні системи.

Сьогодні замість окремих острівців, які не спілкуються, і міста, і муніципалітети повинні уявити собі платформу, яка підтримує та об'єднує всі цифрові функції, необхідні місту для задоволення внутрішніх операційних вимог і взаємодії з громадянами.

Щоб зробити правильний вибір щодо розробки такої платформи, містам потрібно буде застосувати платформне мислення, починаючи з огляду того, що вони мають, куди вони хочуть потрапити та що їм потрібно туди потрапити.

Завдяки взаємодії з багатьма містами під час трансформації розумних міст Nokia помітила, що для повного використання переваг цих ініціатив цим містам потрібно перейти від точкового розгортання окремих послуг і програм до більш пов’язаного, згуртованого підходу.

Стратегія «Місто як платформа» використовує перевірені технології, як-от 5G, промисловий Інтернет речей (IIoT), аналітику та машинне навчання, щоб створити інтелектуальну та інтегровану міську платформу, яка може підтримувати впровадження різноманітних додатків, варіантів використання та бізнес-моделей, що веде до справжніх міських інновацій. Докладніше про цей варіант використання можна знайти в блозі Nokia: Meet the 1000-year-old city of the future | Nokia.

Розумні будинки

Бездротові системи 5G також використовуються в домогосподарствах, перетворюючи їх на розумні будинки.

Завдяки вдосконаленому мобільному широкосмуговому з’єднанню користувачі можуть дистанційно керувати такими функціями, як охоронний доступ до будинку, температура, освітлення та багато інших, залежно від приладів і пристроїв, підключених до мережі. Після підключення всі служби та пристрої стають частиною технології IoT.

Рисунок 10.8 – Автоматизація розумного будинку

Встановлення технологічної системи «розумний дім» забезпечує власникам житла зручність. Замість того, щоб керувати приладами, термостатами, освітленням та іншими функціями за допомогою різних пристроїв, домовласники можуть керувати ними всіма за допомогою одного пристрою — зазвичай смартфона або планшета.

Незважаючи на те, що «розумний дім» пропонує зручність і економію коштів, все ще існують проблеми. Ризики безпеки та помилки продовжують мучити виробників і користувачів технології. Досвідчені хакери, наприклад, можуть отримати доступ до пристроїв розумного дому з підтримкою Інтернету

Розумне виробництво

Розумна фабрика — це оцифроване виробниче підприємство, яке постійно використовує підключені пристрої, обладнання та виробничі системи для збору та обміну даними. Потім ці дані використовуються для прийняття рішень щодо вдосконалення процесів і вирішення будь-яких проблем, які можуть виникнути.

Практики розумного виробництва, які використовуються Smart Factory, забезпечуються різноманітними технологіями, зокрема штучним інтелектом, аналітикою великих даних, хмарними обчисленнями та промисловим Інтернетом речей (IIoT).

Розумні фабрики поєднують цифровий і фізичний світи для моніторингу всього виробничого процесу, від управління ланцюгом поставок до виробничих інструментів і навіть роботи окремих операторів у цехах.

Повністю інтегровані виробничі системи для спільної роботи забезпечують низку переваг для операторів, включаючи можливість адаптації та легкої оптимізації операцій.

Рисунок10.9 – Розумна фабрика

Пошукові звіти та існуючі дослідження визначають чотири рівні розумних фабрик:

Наявність основних даних

На цьому рівні фабрика чи підприємство взагалі не є «розумними». Дані є, але їх нелегко отримати чи проаналізувати. Аналіз даних, якщо він проводиться, займає багато часу та може погіршити ваш виробничий процес.

Проактивний аналіз даних

На цьому рівні можна отримати доступ до даних у більш структурованій та зрозумілій формі. Дані будуть централізовано доступні та впорядковані за допомогою візуалізації та дисплеїв, які допоможуть їх обробити. Усе це дає змогу проводити проактивний аналіз даних, хоча все одно доведеться докладати певних зусиль.

Active Data Insights

На цьому рівні дані можна аналізувати за допомогою машинного навчання та штучного інтелекту, створюючи розуміння без особливого нагляду людини. Система більш автоматизована, ніж на другому рівні, і може завчасно передбачати ключові проблеми чи аномалії, щоб завчасно передбачити можливі збої.

Дані, орієнтовані на дії

Четвертий рівень базується на динамічній природі третього рівня, щоб створювати рішення проблем і, у деяких випадках, вживати заходів для полегшення проблеми або покращення процесу без втручання людини. На цьому рівні дані збираються та аналізуються на наявність проблем, перш ніж генеруються рішення, і, де це можливо, вживаються дії з дуже незначним людським втручанням.

Розумні фабрики використовують підключене обладнання та пристрої для прийняття рішень на основі фактичних даних для оптимізації ефективності та продуктивності протягом усього виробничого процесу. Я можу перерахувати багато переваг, які приносить автоматизація виробництва, але в основному додаткова вартість полягає в:

• Забезпечення гнучкого, ітераційного виробничого процесу може розширити можливості як пристроїв, так і працівників, що призведе до зниження витрат, скорочення простоїв і менше відходів у виробничій промисловості.

• Виявлення, а потім скорочення або усунення недостатньо використовуваних або недоречних виробничих можливостей підвищує ефективність і продуктивність з невеликими інвестиціями в нові ресурси.

• Переваги цифровізації фабрики включають ті, що пов’язані з плануванням, контролем якості, розробкою продукту та логістикою, оскільки кожен оцінюється та оптимізується на основі реальних відгуків.

• Існують також довгострокові переваги, які можна отримати від впровадження машинного навчання в процес. Можна запланувати профілактичне та прогнозне технічне обслуговування на основі точної інформації з реального життя, щоб уникнути зупинки виробничої лінії шляхом збору та аналізу даних.

ЛІТЕРАТУРА

1. What is Artificial Intelligence? [Електронний ресурс]: [Веб-сайт] – Електронні дані: http://www-formal.stanford.edu/jmc/ – Назва з екрана.

2. Congressional Research Service [Електронний ресурс]: [Веб-сайт] – Електронні дані: https://crsreports.congress.gov/product/pdf/R/R45178 – Назва з екрана.

3. SWI-Prolog Wiki facilities [Електронний ресурс]: [Веб-сайт] – Електронні дані. Режим доступу: https://www.swi-prolog.org/wiki/

4. Charu C. Aggarwal Neural Networks and Deep Learning / Springer International Publishing AG, part of Springer Nature, 2018, 497 p.

5. Stuart Russell and Peter Norvig Artificial Intelligence: A Modern Approach Fourth Edition, 2020, Pearson Education, Inc.

6. Разживін О. В. Синтез нечітких регуляторів в системах автоматичного керування: навч. посіб. / О. В. Разживін, О. В. Суботін. – Краматорськ : ЦТРІ «Друкарський дім», 2021. – 212 с.

7. Новожилова М. В. Розробка експертних систем в середовищі CLIPS: навч. посіб. / М. В. Новожилова, О. О. Петрова ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2019. – 130 с.

8. Субботін С. О. Нейронні мережі: теорія та практика: навч. посіб. / С.О. Субботін. – Житомир: Вид.О. О. Євенок, 2020. – 184 с.

9. Sandro Skansi Introduction to Deep Learning From Logical Calculus to Artificial Intelligence / Springer International Publishing AG, part of Springer Nature 2018, 196 p.

10. Aurélien Géron / Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow – Second Edition, 2019, 510 p.

11. Ensemble methods: bagging, boosting and stacking [Електронний ресурс]: [Веб-сайт] – Електронні дані: https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

12. Кононова К. Ю. Машинне навчання: методи та моделі: підручник для бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка» / К. Ю. Кононова. – Харків: ХНУ імені В. Н. Каразіна, 2020. – 301 с.

13. Федоров Є.Є. Моделі та методи комп’ютерних систем розпізнавання зорових образів: монографія / Є. Є. Федоров, О. В. Нечипоренко, Т. Ю. Уткіна, Я. В. Корпань – Черкаси: ЧДТУ, 2021. – 482 с.

14. Литвинов О. А. Об’єктно-орієнтована розробка інформаційних систем. Монографія. / О. А. Литвинов, В. В. Герасимов, Н. В. Карпенко – Д.: Ліра, 2018. – 448с.

15. Троцько В.В. Методи штучного інтелекту: навчально-методичний і практичний посібник. – Київ: Університет економіки та права «КРОК», 2020 – 86с.

16. Дранишников Л.В. Інтелектуальні методи в управлінні: навч. посіб. / Л. В. Дранишников. – Кам'янське: ДДТУ, 2018. — 416 с.

17. Субач І.Ю. Здоренко Ю.М. Фесьоха В.В. Методика виявлення кібератак типу js(html)/scrinject на основі застосування математичного апарату

теорії нечітких множин. Київ. Збірник наукових праць ВІТІ № 4 – 2018, с. 125-131.

18. Ланде Д.В. Основи теорії і практики інтелектуального аналізу даних у сфері кібербезпеки: навч. посіб. / Д.В. Ланде, І.Ю. Субач, Ю.Є. Бояринова – К.: ІСЗЗІ КПІ ім. Ігоря Сікорського», 2018. — 297 с.

19. AI Applications in the Telecommunications Industry: Challenging Telecoms With Machine Learning Solutions [Електронний ресурс]: [Веб-сайт] – Електронні дані. Режим доступу: https://nexocode.com/blog/posts/ai-applications-in-the-telecommunications-industry/

Надруковано в РВЦ Державного університету телекомунікацій
Формат 60х90/16. Папір друкарський.
Наклад 100 прим. Зам. 534.

03110, м. Київ, вул. Солом’янська, 7.
Тел. (044) 249-25-76