МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інформаційних технологій та
комп’ютерної інженерії

Каштан В.Ю.

Методичні вказівки до виконання лабораторних робіт з дисципліни “Інтелектуальні інформаційні технології”. для магістрів галузі знань 12 Інформаційні технології

Дніпро
НТУ “ДП”
2022

УДК 004.4'23+004.42+004.432
Г20

Каштан В.Ю.

Методичні вказівки до виконання лабораторних робіт з дисципліни “Інтелектуальні інформаційні технології” для магістрів галузі знань 12 Інформаційні технології. – Д.: НТУ «ДП», 2022. – 120 с.

Автор:

В.Ю. Каштан, к.т.н., доц., доцент кафедри інформаційних технологій та комп’ютерної інженерії

Погоджено рішенням науково-методичної комісії спеціальності 126 Інформаційні системи та технології (протокол № 6 від 30.06.2022).

Методичні рекомендації містять опис методики виконання лабораторних робіт з дисципліни “Інтелектуальні інформаційні технології” магістрами галузі знань знань 12 Інформаційні технології.

Орієнтовано на активізацію навчальної діяльності магістрів та закріплення практичних знань з даної дисципліни.

ЗМІСТ

ВСТУП

Метою навчальної дисципліни “Інтелектуальні інформаційні технології” є формування умінь та компетенцій та практична підготовка студентів щодо вивчення систем обробки даних та принципів інтелектуального аналізу даних у сферах професійної діяльності

Цілями навчальної дисципліни “Інтелектуальні інформаційні технології” є:

– вивчення сучасних підходів до інтелектуального аналізу даних;

– вивчення теорії еволюційних обчислень, а також розробка та практичне використання методів та алгоритмів, інспірованих природними системами;

– формування навичок самостійної постановки завдань та обґрунтування вибору методу їх вирішення на основі сучасних алгоритмів інтелектуального аналізу даних.

Завданнями дисципліни “Інтелектуальні інформаційні технології” є:

1. Поглиблене вивчення методів оптимізації багатокритеріальних завдань та складання математичних моделей оптимізаційних задач.

2. Формування навичок створення моделей інтелектуального аналізу даних мовою Python.

3. Дослідження методів отримання інформації з текстових даних.

4. Вивчення основних підходів класифікації зображень.

5. Формування навичок побудови нейронних мереж та застосування тих чи інших методів та моделей залежно від розв'язуваного завдання.

Методичні вказівки мають теоретичні відомості та практичні завдання для виконання лабораторних робіт з дисципліни “Інтелектуальні інформаційні технології”. На лабораторних роботах магістранти вивчають основні методи інтелектуального аналізу даних, застосовують їх на практиці за допомогою сучасних програмних середовищ.

Робочою програмою передбачено 7 лабораторних робіт з: визначення інтелектуальної задачі та формування правил предметної області; представлення знань інтелектуальної системи на основі семантичної та фреймової моделей даних; вивчення бібліотек машинного навчання в середовищі Python: Pandas, NumPy, Keras. Це дозволить магістрам глибше зрозуміти суть алгоритмів інтелектуального аналізу даних, інтерпретувати одержані результати.

1 ВИМОГИ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ ТА ОФОРМЛЕННЯ ЗВІТУ

Етапами виконання лабораторних робіт дисципліни “Інтелектуальні інформаційні технології” є:

– уважно ознайомитися з методичними рекомендаціями до конкретної лабораторної роботи (теоретичними відомостями, прикладами, формулюванням завдань);

– виконати індивідуальні завдання за варіантами. Крім того, деякі роботи перебачають вибір рівеня підготовки в залежності від оцінки та складності завдання;

– відповісти на контрольні питання;

– виконати експериментальну частину роботи згідно з отриманим завданням;

– оформити звіт, здати викладачеві (або прикріпити в Moodle) та підготуватись до захисту роботи.

Звіт з будь-якої лабораторної роботи повинен містити

1. Титульний лист, які містить:

– назва дисципліни;

– тема лабораторної роботи;

– дата виконання роботи;

– П.І.Б. студента, курс, номер групи;

– П.І.Б., посада викладача.

Приклад наведено в додатку А.

2. Опис виконаних завдань:

– умову завдання (завдань);

– опис архітектури програми – специфікація програмних вимог (склад, структура модулів, зв'язки між ними, алгоритми): формулювання завдання; специфікація даних; математична модель обробки даних; програмний інтерфейс; план тестування;

– початковий код програми з коментарями для бібліотек (призначення кожної бібліотеки), що підключаються, об'яв, інструкцій, що управляють, і функцій (призначення кожної функції та інструкції, опис параметрів і повертаного значення);

– приклади результатів роботи програми на тестових початкових даних.

3. Висновки за роботою з урахуванням усіх виконаних завдань:

– аналіз отриманих результатів за кожним пунктом завдання;

– аналіз результатів тестування програм;

– ступінь відповідності розроблених програм постановці завдання;

– інша інформація.

4. Вимоги до оформлення звіту

Оформлювання робіт здійснюється за ДСТУ 3008:2015 [13] з урахуванням можливостей текстових комп'ютернихредакторів. В комп'ютерному редакторі виконайте наступні налаштування: поля сторінки: ліве – 2,5 см, праве – 1,5 см, верхнє й нижнє – 2 см; шрифт Times New Roman (висота – 14 пт), міжрядковий інтервал –множник 1,1.

Номери сторінок повинні знаходитися у правому верхньому куті, титульний лист не нумерується.

Листи звіту мають бути з´єднані скріпкою або іншим загальноприйнятим способом**.**

ЛАБОРАТОРНА РОБОТА №1
ВИЗНАЧЕННЯ ІНТЕЛЕКТУАЛЬНОЇ ЗАДАЧІ ТА ФОРМУВАННЯ ПРАВИЛ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Мета роботи

Навчити відрізняти інтелектуальні завдання від не інтелектуальних та складати продукційні моделі для представлення фрагментів знань.

1.2 Завдання до лабораторної роботи

1. Відповідно до таблиці 1.1 визначити які завдання є інтелектуальними. На оцінку «відмінно» аргументуйте відповідь.
2. Результати навести у вигляді табл.1.2.
3. Розробити набір правил відповідно до індивідуального завдання табл.3. Варіант завдання відповідає порядковому номеру академічного журналу групи.
4. Сформувати набір правил, представлених за зразком:

Правило N: *ЯКЩО* <Передумова> *ТО* <Висновок>.

5. Оформити звіт відповідно до стандарту ДСТ ЄСПД.

Таблиця 1.1 – Завдання частина 1

N	Завдання
1.	Прогнозування погоди.
2.	Визначення несправності автомобіля в разі якщо автомобіль не заводиться.
3.	Визначення стратегії гоночної яхти у регаті.
4.	Формування навчального плану університету; розподіл навантаження викладачів
5.	Автоматизований переклад тексту.
6.	Розрахунок оплати комунальних послуг.
7.	Визначення раціону домашніх тварин.
8.	Побудова графіків функцій.
9.	Аналіз концентрації шкідливих речовин в повітрі.
10.	Розрахунок траєкторії руху космічного тіла.
11.	Зарахування абітурієнтів до ЗВО.
12.	Постановка діагнозу пацієнта.
13.	Визначення знака зодіаку.
14.	Визначення відстані між планетами.
15.	Вибір оптимальної моделі мобільного телефону для покупця

Таблиця 1.2 – Представлення результатів

Інтелектуальні задачі	**Не інтелектуальні задачі**

Таблиця 1.3 – Завдання частина 2

N	Завдання
1.	1. Локалізація несправності персонального комп'ютера за зовнішніми симптомами
2.	Визначення конфігурації персонального комп'ютера в залежності від потреб користувача
3.	Визначення конфігурації комп'ютера з точністю до моделей комплектуючих з урахуванням можливих конфліктів обладнання
4.	Вибір комплекту програмного забезпечення для комп'ютера відповідно до завдань, які необхідно вирішувати користувачеві
5.	Вибір оптимальної мови програмування залежно від поставленого завдання
6.	Визначення «прогалин» у знаннях учня з конкретного курсу
7.	Підбір навчальних курсів, які необхідно вивчити для отримання певних навичок (веб-програміст, windows-програміст, unix-програміст, тестувальник, архітектор ПЗ тощо)
8.	Визначення місця відпочинку для майбутньої відпустки
9.	Вибір вищого навчального закладу абітурієнтом
10.	Вибір оптимального способу вивчення іноземної мови в залежності від бюджету, початкового рівня знань та інших особливостей учня
11.	Вибір операційної системи персонального комп'ютера або робочої станції (DOS, Windows, Linux, FreeBSD, MacOS 9, MacOS X, Digital UNIX та ін.)
12.	Вибір оптимального способу підключення до Інтернету (модем, локальна мережа, ADSL, GPRS, супутниковий канал)
13.	Підбір мисливського спорядження
14.	Формування набору документів для вчинення нотаріальної дії (купівля-продаж квартири, оформлення спадщини, заповіту тощо)
15.	Профорієнтація школяра
16.	Формування набору документів для оформлення Шенгенської/Американської/Японської візи

17.	Прогнозування пенсії з урахуванням поточного законодавства
18.	Вибір оптимального способу вкладення грошей відповідно до потреб інвестора
19.	Вибір кафедри для навчання в НТУ «Дніпровська політехніка»(з урахуванням декількох факультетів та безлічі можливих ситуацій та ускладнень)
20.	Вибір оптимальної стратегії дій для студента, який перебуває на межі відрахування на сесію
21.	Діагностика несправності ПК
22.	Вибір книги відповідно до побажань читача
23.	Вибір оптимального способу подорожі літаком
24.	Вибір оптимального способу подорожі машиною
25.	Вибір оптимального способу подорожі потягом
26.	Діагностика несправності смартфона
27.	Вибір оптимального способу подорожі автостопом
28.	Вибір оптимального способу подорожі пароплавом
29.	Вибір продуктів для приготування борщу
30.	Розробити набір правил для підбору краватки

1.3 Теоретичні відомості

Інтелект — якість психіки, що складається із здатності адаптуватися до нових ситуацій, здатності до навчання на основі досвіду, розуміння і застосування абстрактних концепцій і використання своїх знань для управління навколишнім середовищем; Загальна здатність до пізнання і вирішення труднощів, яка об’єднує всі пізнавальні здібності людини: відчуття, сприйняття, пам’ять, уявлення, мислення, уява; здатність мозку вирішувати (інтелектуальні) задачі шляхом надбання, запам’ятовування і цілеспрямованого перетворення знань в процесі навчання на досвід й адаптації до різноманітних обставин.

У цьому визначенні під терміном «знання» мається на увазі не тільки та інформацію, яка надходить у мозок через органи почуттів.

Такого типу знання, отримані через органи чуття надзвичайно важливі, але недостатні для інтелектуальної діяльності. Справа в тому, що об’єкти оточуючого нас середовища мають властивість не тільки впливати на органи почуттів, але і знаходитися один з одним у певних відносинах. Ясно, що для того, щоб здійснювати в навколишньому середовищі інтелектуальну діяльність (або хоча б просто існувати), необхідно мати в системі знань модель цього світу. У цій інформаційній моделі навколишнього середовища реальні об’єкти, їх властивості і відносини між ними не тільки відображаються і запам’ятовуються, але й, як це зазначено в даному визначенні інтелекту, можуть подумки «цілеспрямовано перетворюватися». При цьому важливо те, що формування моделі зовнішнього середовища.

відбувається «у процесі навчання на досвіді й адаптації до різноманітних зовнішнім обставинам».

Для пояснення відмінності інтелектуальної задачі від 23е інтелектуальної, необхідно розглянути термін «алгоритм»

Алгоритм — послідовність, система, набір систематизованих правил виконання процесу, який обов'язково призводить до вирішення певного класу задач після виконання кінцевого числа операцій.

Що ж стосується задач, алгоритми рішення яких уже встановлені, то, як зазначає відомий фахівець у галузі штучного інтелекту М. Мінський,

«надмірно приписувати їм такі містичні властивості, як «інтелектуальність». Справді, після того, як такий алгоритм уже знайдений, процес вирішення відповідних завдань стає таким, що його можуть в точності виконати людина, обчислювальна машина (належним чином запрограмована) або робот, що не мають ні найменшого уявлення про сутність самої задачі. Потрібно тільки, щоб особа, що розв'язує задачу, була здатна виконувати ті елементарні операції, з яких складається процес, і, крім того, щоб вона педантично і акуратно керувалася запропонованим алгоритмом. Така особа, діючи, як кажуть у таких випадках, чисто машинально, може успішно вирішувати будь-яку задачу розглянутого типу.

Таким чином, справедливо стверджувати, що інтелектуальна задача — задача, розв'язання якої неможливо з використанням стандартних методів розв'язання, алгоритмізації. Так до інтелектуальних задач відносяться доведення теорем, розпізнавання образів, гра в шахи.

У свою чергу не інтелектуальна задача — задача, що вимагає для вирішення готових алгоритмів; прикладом не інтелектуальної задачі може бути будь-яка обчислювальна задача (рішення системи рівнянь, переклад чисел з однієї системи числення в іншу і т.д.).

Формування правил продукцій.

Продукції (як і мережевими моделями) є найбільш популярними засобами подання знань в інформаційних системах. Звичайна форма продукції виглядає так: ЯКЩО А, ТО В.

Що розуміється у звичайному логічному сенсі, як знак логічного слідування В з істинного А. Можливі й інші інтерпретації продукції, наприклад, А описуєдеяку умову, необхідну , щоб можна було вчинити дію

В. Продукційна модель або модель, заснована на правилах, дозволяє представити знання у вигляді пропозицій типу

«Якщо (умова), то (дія)». (1.1)

Під умовою розуміється деяка пропозиція — зразок, за яким здійснюється пошук у базі знань, а під дією — дії, що виконуються при успішному результаті пошуку (вони можуть бути проміжними, виступаючими далі як умови, і термінальними або цільовими, такими що завершують роботу системи).

При використанні продукційної моделі база знань складається з набору

правил, Програма, що управляє перебором правил, називається машиною виводу. Найчастіше висновок буває прямий (від даних до пошуку мети) або зворотний (від цілі для її підтвердження — до даних). Дані — це вихідні факти, на аступному ви запускається машина виводу.

Однак не слід ототожнювати правило-продукцію і відношення логічного слідування. Справа в тому, що інтерпретація продукції залежить від того, що знаходиться ліворуч і праворуч від знака логічного слідування. Часто під А розуміється деяка інформаційна структура (наприклад, фрейм), а під В — деяка дія, що полягає в її трансформації (перетворенні). Поняття продукції ширше логічного слідування. Як приклад розглянемо правило, взяте з бази знань експертної системи MYCIN, призначеної для діагностики інфекційних захворювань. (асту.1.4)

Таблиця 1.4 – Фрагмент правил діагностики системи MYCIN

ЯКЩО	ТО
місце виділення культури – кров І реакція мікроорганізму – грам, І форма мікроорганізму – паличка, І пацієнт відноситься до групиризику,	з упевненістю (0,6) назва мікроорганізму — *pseudomoniaaeruginosa.*

Умова правила складається з чотирьох фактів, з’єднаних союзом «І». Якщо всі чотири факти мають місце, то вірним буде слідство правила. З кожним правилом зв’язується деяке число, що приймає значення в діапазоні від -1 до 1, що виражає ступінь достовірності наслідків і називається коефіцієнтом впевненості.

У загальному випадку продукційну модель можна представити в аступному вигляді:

$$N = <A, U, C, I, R> \quad (1.2)$$

N — ім'я продукції;

А — сфера застосування продукції; U — умова застосовності продукції; С — ядро продукції;

І — постумови продукції, актуалізуються при позитивній реалізації продукції;

R— коментар, неформальне пояснення(обґрунтування) продукції, час введення в базу знань та ін.

Переваги і недоліки продукційної моделі.

Переваги продукційної моделі полягають у наступному.

- Переважна частина людських знань може бути записана у вигляді продукцій.

- Простота створення та розуміння окремих правил.
- Простота поповнення та модифікації бази знань (набору продукцій).
- Простота механізму логічного висновку.
- Розбиття системи продукцій на сфери(декомпозиція) дозволяє ефективновикористовувати ресурси і скоротити час пошуку рішення.
- Можливість реалізації немонотонного логічного виведення й обробки суперечливих фактів.
- Можливість паралельної і асинхронної обробки правил.

Недоліки продукційної моделі проявляються в наступному.

- Відсутність теоретичного обґрунтування в побудові продукційних систем. Восновному при їх побудові використовуються евристичні прийоми.
- При великому числі продукцій процедура перевірки несуперечності правил і коректності роботи системи стає вкрай складною. Саме тому число продукцій, з якими працюють реальні інформаційні системи, не перевищує тисячі.
- Можливість легкого внесення серйозних спотворень в базу знань, що призводять до неправильного функціонування системи (якщо в системі немає розвинених засобів перевірки цілісності бази знань).

Приклад.

Нехай заданим є фрагмент бази знань з двох правил:

Правило 1:

ЯКЩО «відпочинок влітку» і «людина активна», *ТО* «їхати в гори».

Правило 2:

ЯКЩО «любить сонце», *ТО* «відпочинок влітку».

Припустимо, що в систему надійшли дані людина активна і любить сонце.

Крок 1: Пробуємо Правило 1 — не працює, так як не вистачає даних «відпочинок влітку».

Крок 2: Пробуємо Правило 2 — працює, в базу надходить новий факт «відпочинок влітку»

Крок 3: Знову пробуємо Правило 1 — працює, активує мету «їхати в гори», яка виступає як порада, що видається системою.

Приклад продукційної моделі

Розглянемо продукційну модель на прикладі інструкції з будівництва моста і його експлуатації, заснованої на правилах. У модельній реалізації у вигляді продукційної моделі вірш буде виглядати наступним чином:

Правило 1. ЯКЩО з колод ТО згниє

Правило 2. ЯКЩО зі сталі ТО заіржавіє

Правило 3. ЯКЩО з каменів ТО зітреться

Правило 4. ЯКЩО із золота ТО вкрадуть

Правило 5. ЯКЩО поставити охорону ТО не вкрадуть Правило 6. ЯКЩО

охорона засне ТО вкрадуть

Правило 7. ЯКЩО охороні дати вина й трубку ТО не засне

Тепер спробуємо за допомогою даної моделі дізнатися, які дії необхідно зробити, щоб побудувати такий міст, який буде стояти досить довго.

Якщо ми говоримо, що збираємося побудувати міст із золота і поставити охорону, дана продукційна модель видасть наступний результат — Правило 6 говорить про те, що існує ймовірність, що охорона засне і міст вкрадуть, тому нам доведеться додати ще одну умову — дати охорони вина і трубку. Тепер, маючи всі достатні умови, ми зможемо побудувати міст, який буде стояти досить довго.

1.4 Приклад виконання роботи:

Розглянемо частину 2 для 30 варіанту «Розробити набір правил для підбору краватки».

Знання, надані консультантом з питань моди (експертом):

1. Краватка потрібна, коли Ви одягаєте офіційний костюм або діловий костюм вробочий день.
2. Якщо Ви збираєтеся поїхати в діловому костюмі на уїк-енд, то Ви можете надіти краватку, але це необов'язково.
3. Якщо Ви носите спортивний піджак, то краватка необхідна.
4. До офіційного костюма підійде чорна краватка-метелик або біла краватка.
5. До блакитного або сірого ділового костюма підійде червона краватка в смужку або однотонна червона.
6. Якщо Ваш костюм зеленого або коричневого кольору, тоді Вам підійде руда смугаста краватка або коричнева смугаста краватка, в крайньому випадку зелена візерунчаста краватка.

Контрольні питання

1. Що називають «інтелектом»?
2. Що називають «інтелектуальним завданням»?
3. Що називають «алгоритм»?
4. Які існують види інтелектуальних завдань?
5. Чи можливий перехід завдання з розряду «інтелектуальних» в розряд «не інтелектуальних»? Наведіть приклади.
6. Що називають «продукційною моделлю»?
7. Що називають «коефіцієнтом впевненості»?
8. Як виглядає продукційна модель в загальному вигляді?
9. У чому переваги і недоліки продукційної моделі?

ЛАБОРАТОРНА РОБОТА №2
ПРЕДСТАВЛЕННЯ ЗНАНЬ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ НА ОСНОВІ СЕМАНТИЧНОЇ МОДЕЛІ

2.1 Мета роботи

Навчити представляти фрагменти знань у вигляді семантичної моделі.

2.2 Завдання до лабораторної роботи:

1. Студенти за варіантом (табл.2.1) представляють знання інтелектуальної системи для конкретної предметної області.

2. Семантична мережа повинна містити щонайменше 20 вершин із різними типами зв'язків (на оцінку «відмінно»), 10 вершин (на оцінку «добре»), 5 вершин (на оцінку «задовільно»).

3. Створити класи та зв’язки в Protégé 5.0, підписати лінії.

4. Представити графічний вид семантичної мережі, виконане в середовищі Protégé 5.0 з покроковим описанням дій, зв’язків.

5. Розробити схему семантичної мережі з визначенням вершин та відношень.

Таблиця 2.1 - Варіанти завдань

N варіанту	Предметна область
16.	Кухня.
17.	Вітальня.
18.	Спальня.
19.	Дитяча
20.	Санітарний вузол
21.	Ванна кімната
22.	Кладовка
23.	Вбиральня
24.	Кімната відпочинку
25.	Ігрова кімната
26.	Спортивна кімната
27.	Тренажерний зал
28.	Комп’ютерний клас
29.	Центральні органи — це президент, уряд і парламент з двох палат (нижня — Палата депутатів, верхня — Сенат).
30.	Органи місцевого самоврядування

2.3 Теоретичні відомості

Термін "семантична" означає "смислова", а сама семантика — це наука, що встановлює відносини між символами і об'єктами, які вони позначають, тобто наука, що визначає сенс знаків. Модель на основі семантичних мереж була запропонована американським психологом Куіллліаном. Основною її

перевагою є те, що вона більше за інших відповідає сучасними уявленнями проорганізацію довготривалої пам'яті людини.

Семантична мережа — це орієнтований граф, вершини якого — поняттяя,а дуги — відносини між ними.

Можна запропонувати кілька класифікацій семантичних мереж, пов'язанихз типами відносин між поняттями:

за кількістю типів відносин:

однорідні (з єдиним типом відносин);

неоднорідні (з різними типами відносин).

за типами відносин:

бінарні (в яких відносини пов'язують два об'єкта);

N-арні (в яких є спеціальні відносини, що зв'язують більше двох понять).

Семантична мережа підтримує 4 типи відносин:

IS (АКО) — «є». A IS B — сутність A відноситься до класу сутностей B.Наприклад, «кішка є домашня тварина».

ISAPART - «є частина». A ISAPART B — сутність A є частиною B, входитьдо B. Наприклад, «голова є частина тулуба».

HAS — «приватна ознака». A HAS B — B є приватним ознакою A. Якщо A входить в яку-небудь іншу сутність C, ознака B не передається C. Наприклад, «волосатість є ознака голови, голова є частина тіла — звідси не випливає, щовсе тіло вкрите волоссям» .

PROP — «загальна властивість». A PROP B — B є загальною властивістю A, яке переноситься на всі сутності, в яких входить A. Наприклад, «голова покрита шкірою, голова є частина тіло, і тіло також покрито шкірою».

Мінімальний склад відносин в семантичної мережі такий: елемент класу або АКО; атрибутивні зв'язку / мати властивість;значення властивості.

Недоліком цієї моделі є складність організації процедури виведення на семантичної мережі. Ця проблема зводиться до нетривіальною задачі пошуку фрагмента мережі, відповідного деякої підмережі, що відбиває поставлений запит до бази.

Переваги семантичних мереж проявляються у наступному:

- універсальність, що досягається за рахунок вибору відповідного набору відносин. В принципі за допомогою семантичної мережі можна описати будь- яку складну ситуацію, факт або предметну область;
- наочність системи знань, представленої графічно;
- близькість структури мережі, що представляє систему знань, до семантичної структурі фраз природною мовою;
- відповідність сучасним уявленням про організацію довготривалої пам'яті людини.

Як приклад на рисунку зображена семантична мережа, що представляє частину знань про тваринний світ.

Дана мережа була розроблена А.М. Коллінзом і М.Р. Куілліаном (1969) і використовувалася, для моделювання механізмів пам’яті людини. Завдяки

відносинам ako мережа являє собою якусь ієрархічну структуру і дозволяє відповідати на такі питання, як: “чи є канарка птахом?”, “чи може канарка літати?” і т.д. (рис.2.1). У ході експериментів із представленою моделлю було встановлено, що людина прагне запам’ятовувати інформацію, що відповідає найбільш абстрактному рівню. Наприклад, замість того, щоб безпосередньо запам’ятовувати факт “канарка вміє літати”, людина представляє цей факт у вигляді властивості, і зв’язує його з поняттям “птах”. Аналогічно властивості: “дихати”, “їсти”, “переміщатися” – зв’язуються з поняттям “тварина”. Такий метод запам’ятовування інформації дозволяє виключити її дублювання в базі знань завдяки спадкуванню властивостей у відповідності з ako ієрархією. Наприклад, властивість “мати крила” притаманне птахам, отже, воно притаманне і канарці.

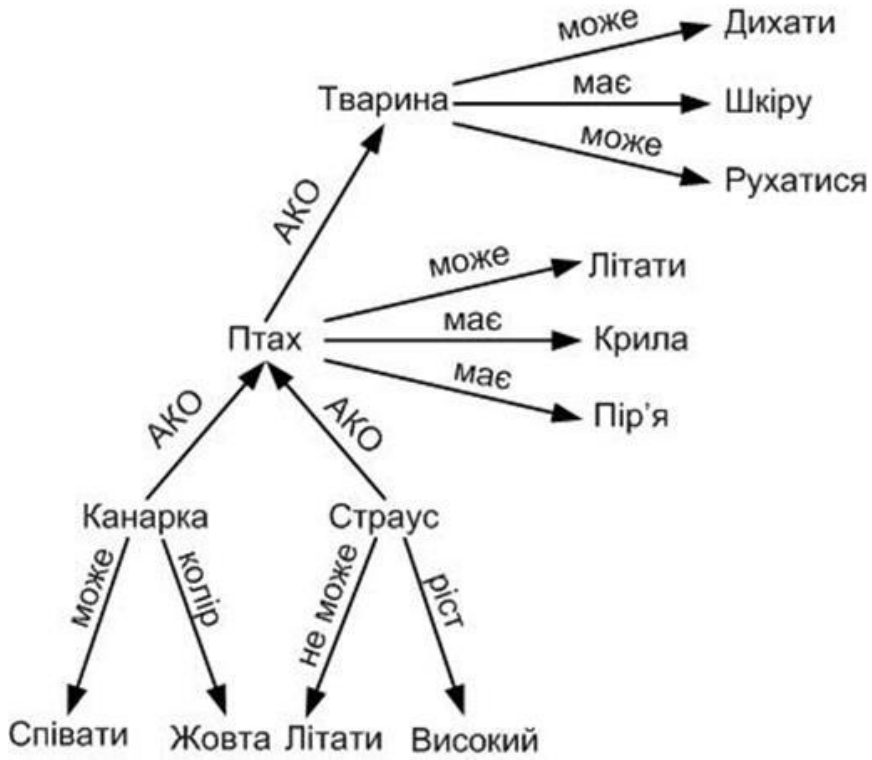


Рисунок 2.1 – Світ тварин у вигляді семантичної мережі

2.3.1 Приклад для реалізації в програмному середовищі:

Будемо вважати, що система «Хлібний магазин» складається з таких елементів: хліб, продавець, покупець, прилавок, автомобіль, шофер, вантажник, гроші, чек. Побудувати семантичну мережу, в якій вершинами будуть перераховані об'єкти, а дугами — відносини між ними.

1. Відкрити програмне середовище та присвоївши їй ім'я «Хлібний магазин»

2. Розмістимо відомі нам елементи (виконати подвійне клацання мишею, ввести назву).

3. Встановимо зв'язки між об'єктами:

- продавець: отримує гроші, продає хліб, вручає чек;
- покупець: дає гроші, отримує хліб, отримує чек;

- шофер: водить автомобіль;
- автомобіль: перевозить хліб;
- вантажник: вивантажує хліб;
- автомобіль: перевозить хліб;
- хліб: лежить на прилавку.

4. Розмістимо зв'язки у вигляді ліній.
5. Підписати лінії (рис.2.2).
6. Засобами MS Visio створити схему семантичної мережі (рис.2.3).

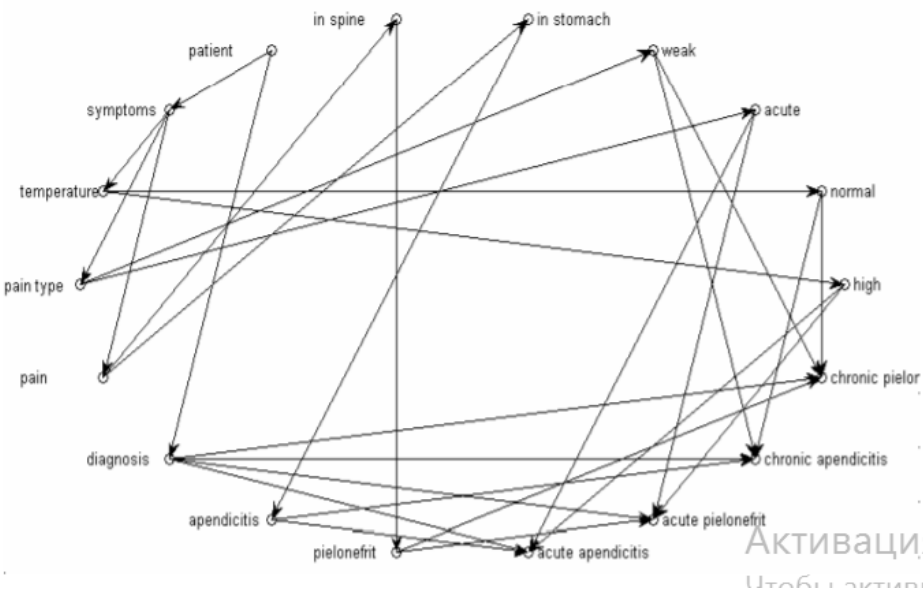


Рисунок 2.2 – Програмна побудова семантичної мережі

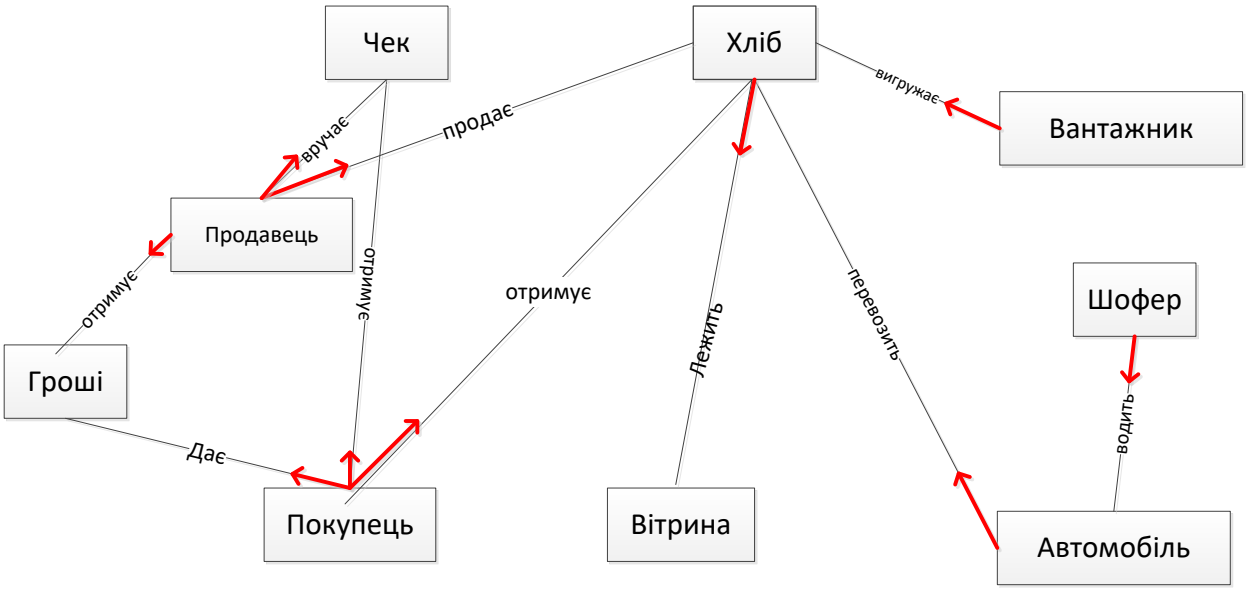
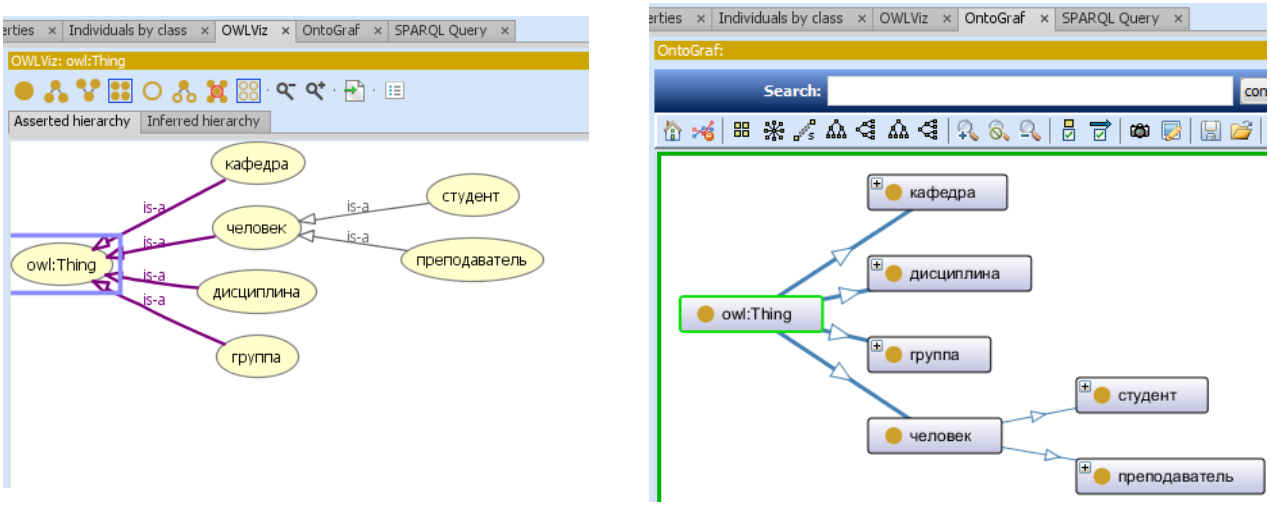


Рисунок 2.3 – Схема семантичної мережі

Автоматичне побудова та відображення графа в **Protégé** та взаємозв'язків між класами виконується при виборі вкладок "*OWLViz*" та "*OntoGraf*" (рис.2.4).



а) OWLViz б) OntoGraf

Рисунок 2.4 – Графічне відображення мережі

2.4 Приклад виконання роботи

Крок 1. Для створення онтології відкриваємо редактор Protégé 5.0 (завантажити можна з офіційного сайту або скористатися онлайн версією) та вводимо її назву – наприклад, NQF_FQF, та версію – /1.0 (рис.2.5).

Примітка: Ми створюємо онтологію на прикладі LAB1

Для додавання інструкції, яка пояснювала б про що ваша онтологія, необхідно натиснути на «плюсик» біля слова Annotation.

У полі Value введіть текст анотації та натисніть Ок.

Оскільки ми пишемо онтологію з використанням української мови, то мову анотації ми не обираємо.

Для редагування інструкції тут та інших розділах редактора потрібно натиснути на «кружечок» у правому куті інструкції (рис.2.6).

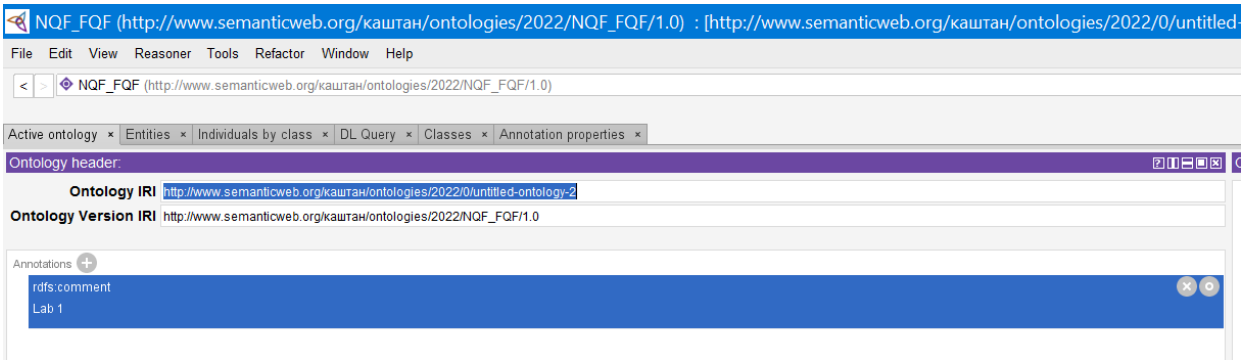


Рисунок 2.5 – Редактор Protégé 5.0

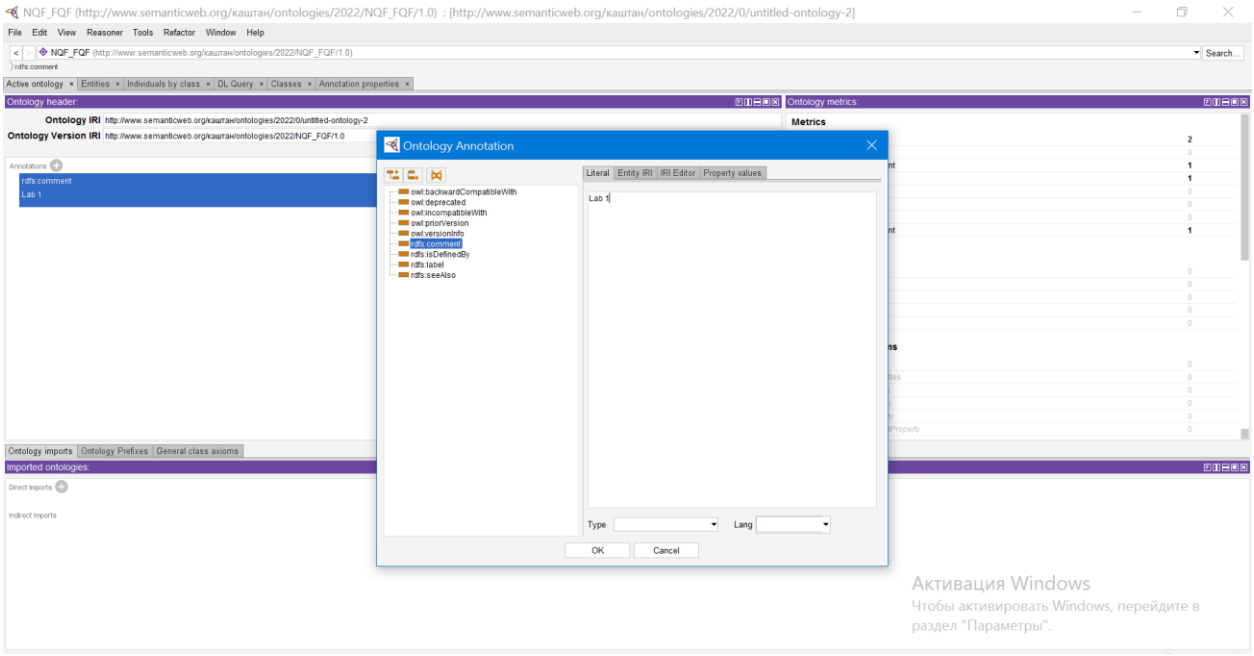


Рисунок 2.6 – Назва схеми

Крок 2. Створення класів. Для створення класів необхідно перейти на вкладку Classes. Якщо у редакторі вона не відображається, потрібно на панелі інструментів вибрати Window – Tabs – Class views (рис.6).

Далі створюємо свій клас. Для цього потрібно натиснути кнопку Add subclass.

У вікні (рис.2.7) необхідно ввести назву класу. Тут слід зазначити, що кириличні символи працюють, але для української мови не працює – апостроф. При написанні слова з апострофом – частина відсікається текст до апострофа.

Примітка: використовуємо кирилицю у назвах класів для наочного уявлення в графі. Рекомендовано латинцю використовувати.

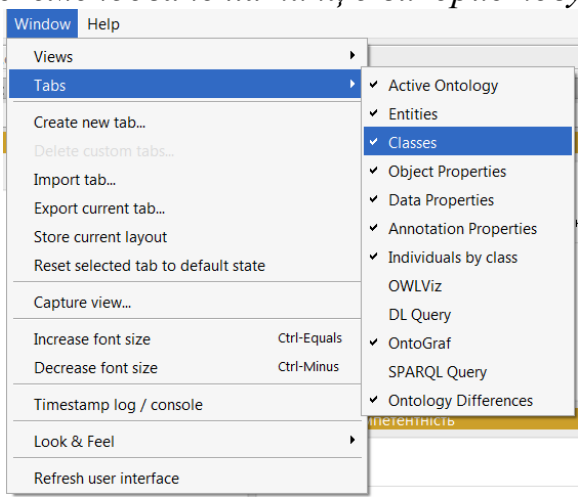


Рисунок 2.7 – Активація кнопки «Клас»

Зверніть увагу на назву класів: 1_word НЕ пишуться. Замість цього в редакторі виходить *_word*

Але так можна написати:

Word_1

Тобто редактору не подобається, коли назва класу починається з цифри (рис.2.7).

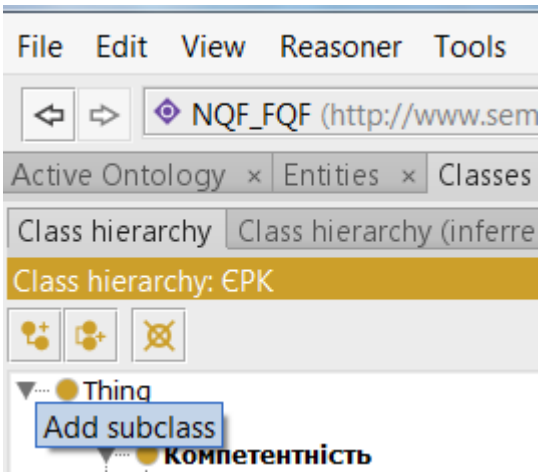


Рисунок 2.7 – Назва класу

Після запровадження назва класу у структурі онтології з'являється новий клас.

Тепер ми створили один клас та три підкласи. Для кращого опису онтології зробимо не пов'язаними між собою класи.

Щоб це зробити, необхідно вибрати клас, потім у вікні Description натиснути на "плюс" біля Disjoint With (рис.2.8).

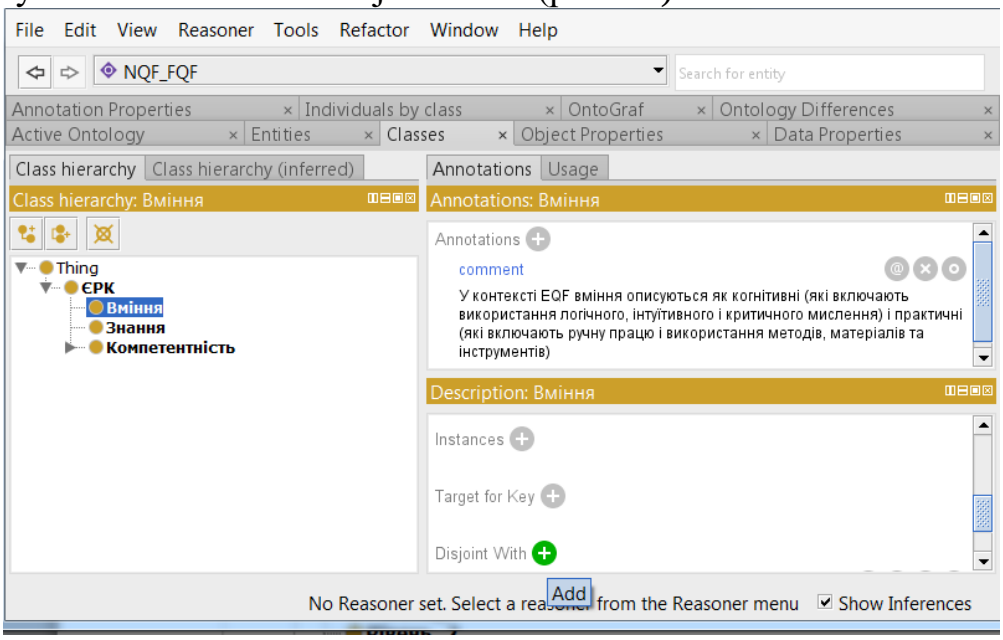


Рисунок 2.8 – Додання коментарів

У вікні класів вибрати ті, які ви не хочете пов'язувати. Комбінація Ctrl+ліва кнопка миші дозволяє вибрати відразу кілька класів. Далі - Ок.

Тепер необхідно створити класи 2 рівня для кожного з класів 1 рівня. Чудовою особливістю редактора і те, що зробивши пов'язаним один клас, інші робляться автоматично.

Для цього у підкласі «Компетентність» створюємо 8 підкласів: Рівень_1, Рівень_2…Рівень_8.

Для створення підкласів ми скористаємось функцією створення ієрархії класів. У головному меню інструментів Protege Tools виберіть Create Class hierarchy.

У вікні виберіть основний клас і натисніть Continue.

Потім у полі Prefix пишемо «Рівень_», а нижче в полі потрібно написати назви доданих класів через пробіл і натиснути Continue (рис.2.9).

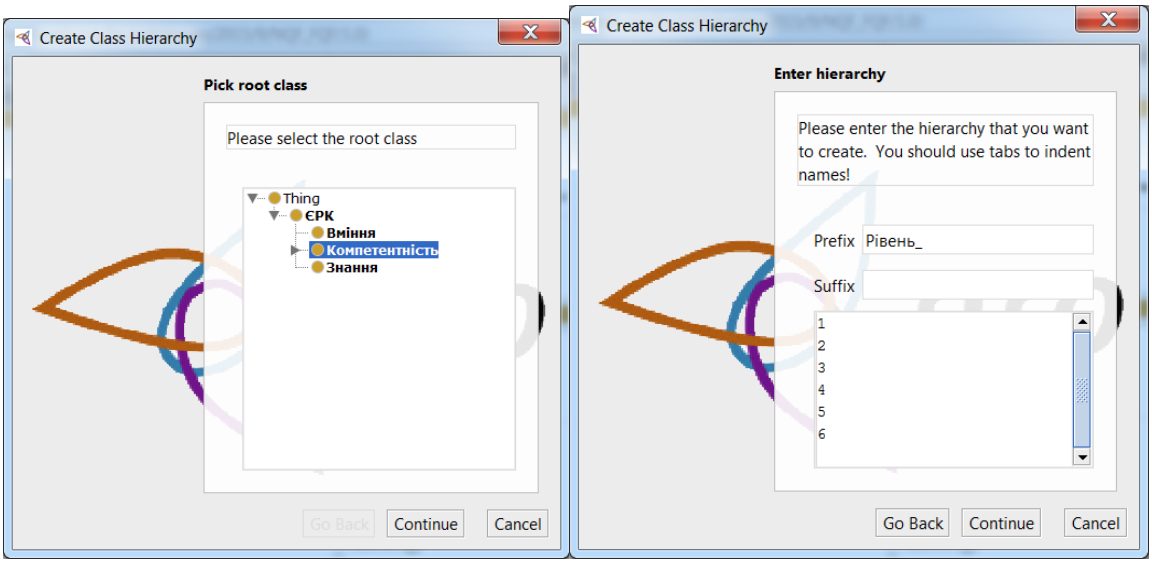


Рисунок 2.9 – Створення підкласів

Отримуємо результат на рис.2.10.

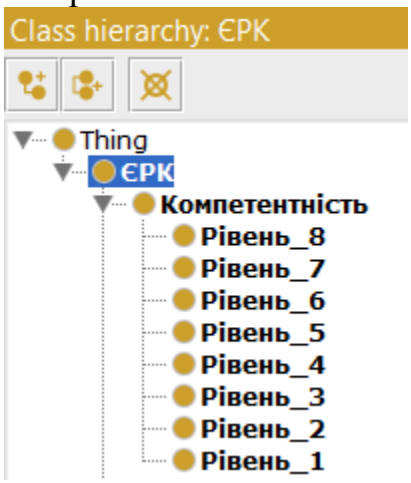


Рисунок 2.10 – Результат створених підкласів

Крок 3. Створення зв'язків.

Ці підкласи (Рівень_1… Рівень_8) не можна скопіювати у всі інші (а нам потрібно, щоб ці рівні були у всіх попередніх класів), але можна додати зв'язки.

Для цього слід вибрати клас та у вікні Description натиснути на «плюсик» біля SubClass Of (рис.2.11).

У вікні вибрати вкладку Class hierarchy і зі списку класів вибрати ті, які ви хочете зв'язати. Комбінація Ctrl+ліва кнопка миші дозволяє вибрати відразу кілька класів. Далі - Ок.

Таким чином, у SubClass Of з'явиться перелік пов'язаних класів (рис.2.12).

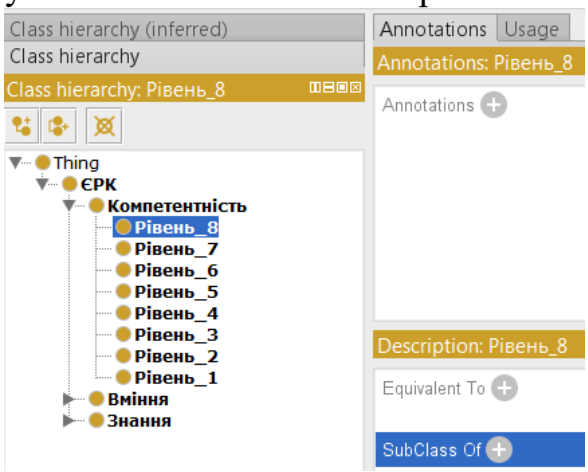


Рисунок 2.11 – Додавання зв’язків

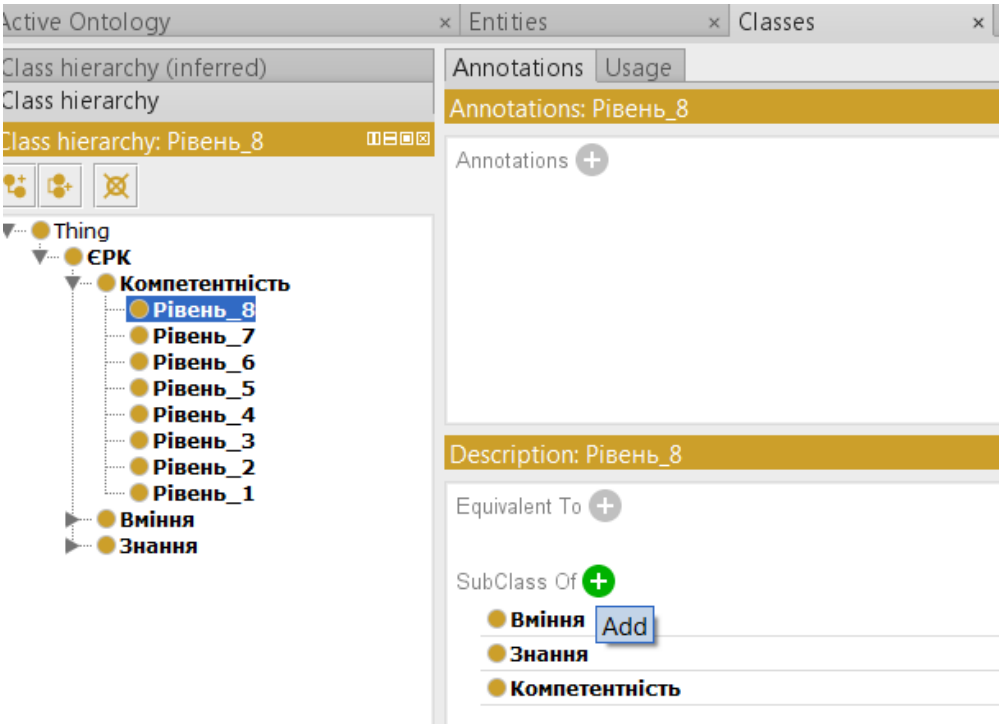


Рисунок 2.12 – Вибір класів для зв’язку

Якщо додати плагін OntoGraf, можна побачити структуру онтології як

графа після переходу вкладку OntoGraf (рис.2.13).

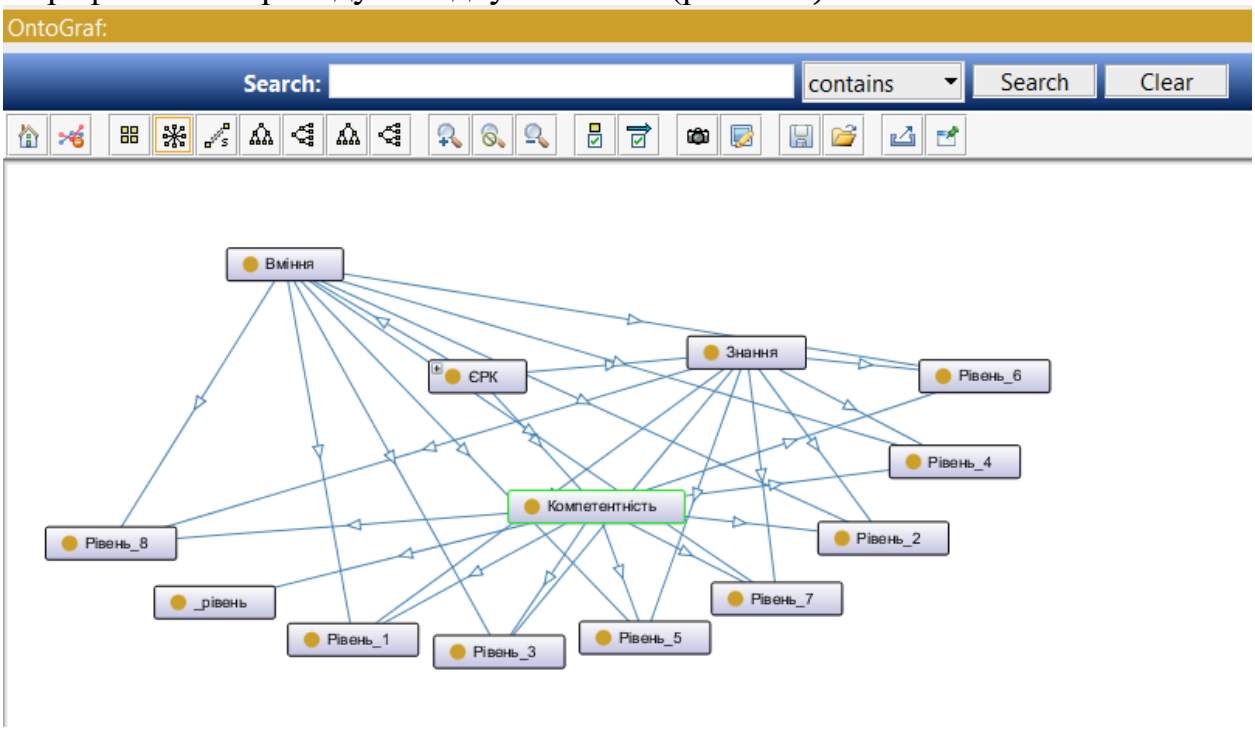


Рисунок 2.13 – Виведення графу

Крок 4. Додавання властивостей.

Примітка: ми створили додаткові класи, тому з'являться назви класів, про які не було написано вище.

Створимо якість на вкладці Object Properties. Як і під час створення класу вибираємо властивість і натискаємо кнопку  .

У вікні пишемо назви властивості. Правила написання назв тут такі ж, як і назв класів.

Тепер цю властивість надають об'єкту, наприклад «Елементарні_загальні_знання».

Для цього на тій же вкладці у вікні Description натискаємо на плюсик біля Domens (intersection) (рис.2.14).

У вікні, що відкрилося, вибираємо клас «Елементарні_загальні_знання».

Далі натискаємо «плюсик» біля Ranges (intersection) та у вікні, що відкрилося вибираємо клас «Рівень_0». Ми пов'язуємо ці класи через властивість «належати».

Це відображається відповідними лініями на графі OntoGraf.

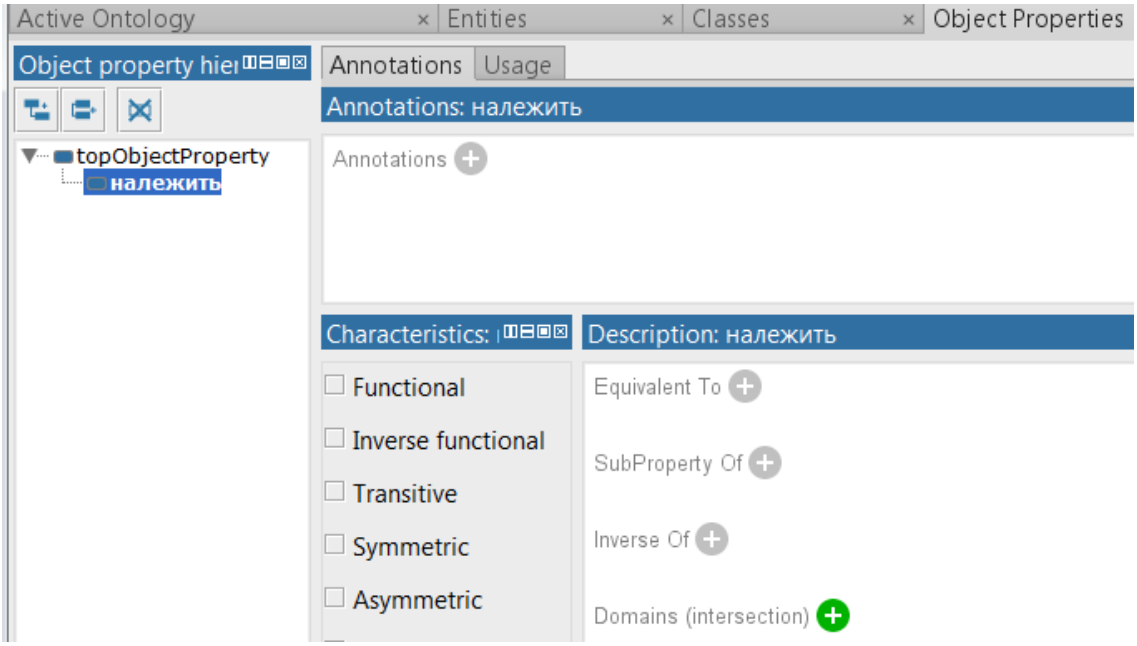


Рисунок 2.14 – Додавання властивостей

Крок 5. Додавання характеристики.

Для того, щоб додати до властивості характеристику, потрібно його виділити, потім у вікні Characteristics натиснути «галочку» біля Symmetric (рис.2.15).

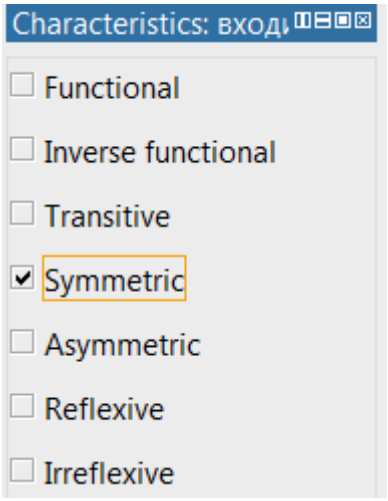


Рисунок 2.15 – Додавання характеристик

Так само чинимо з іншими класами, підкласами, властивостями тощо…

У результаті отримуємо граф. Для того, щоб уявити граф в зручному вигляді потрібно «вручну» пересунути елементи, щоб все було добре видно, т.к. OntoGraf автоматично розміщує не дуже красиво. Результат наведено на рис.2.16.

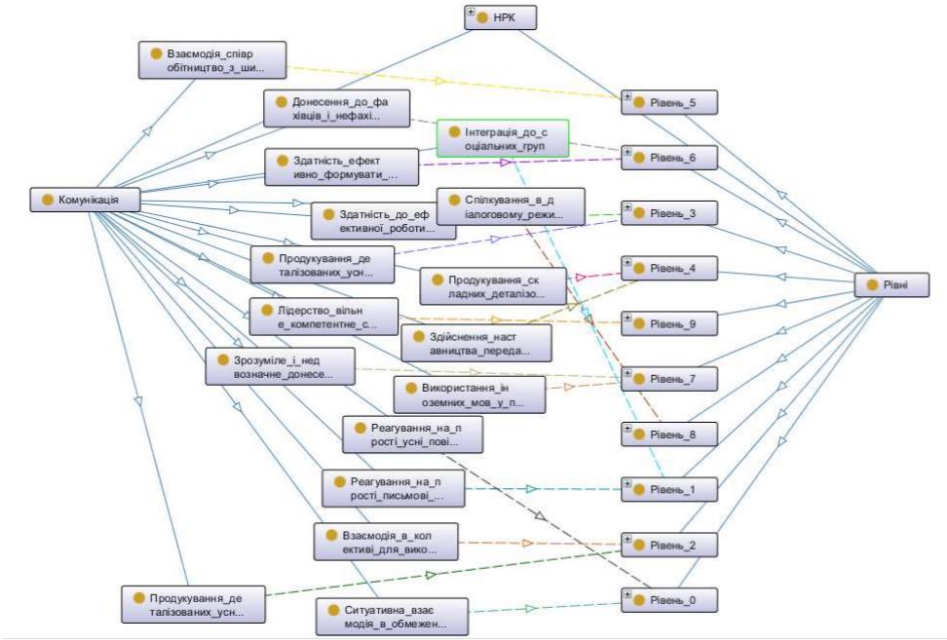


Рисунок 2.16 – Фрагмент семантичної мережі

2.5 Контрольні питання

1. Що називають «семантичної мережею»?
2. З яких елементів складається семантична мережа?
3. Які існують класифікації семантичних мереж?
4. Які типи відносин підтримуються семантичними мережами?
5. У чому переваги і недоліки семантичних мереж?

ЛАБОРАТОРНА РОБОТА №3
ПРЕДСТАВЛЕННЯ ЗНАНЬ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ НА ОСНОВІ ФРЕЙМОВОЇ МОДЕЛІ

3.1 Мета роботи

Навчити представляти фрагменти знань у вигляді фреймової моделі.

3.2Завдання до лабораторної роботи

1. Студенти за варіантом представляють знання інтелектуальної системи для конкретної предметної області відповідно до лабораторної роботи №2 (таб.3.1).

2. Фреймова модель повинна містити щонайменше 6 фреймів, пов'язаних двома типами зв'язку.

3. Представити у вигляді таблиці структуру фрейму .

4. Представити графічний вид фреймової моделі (MS Office, MS Visio та Protégé 5.0).

5. Детально описати представлену область знань та навести результати у звіті.

Таблиця 3.1 – Варіанти завдань

N варіанту	Предметна область
31.	Кухня.
32.	Вітальня.
33.	Спальня.
34.	Дитяча
35.	Санітарний вузол
36.	Ванна кімната
37.	Кладовка
38.	Вбиральня
39.	Кімната відпочинку
40.	Ігрова кімната
41.	Спортивна кімната
42.	Тренажерний зал
43.	Комп’ютерний клас
44.	Центральні органи — це президент, уряд і парламент з двох палат (нижня — Палата депутатів, верхня — Сенат).
45.	Органи місцевого самоврядування

3.2 Теоретичні відомості

Фреймова модель представлення знань була запропонована М. Мінським в 1979році і є розвитком семантичних мереж.

Фрейм (англ. frame) - абстрактний образ для представлення деякого стереотипу сприйняття. Кожен фрейм має власну назву і список слотів і їх

значень.

Значеннями можуть бути дані будь-якого типу, а також назва іншого фрейма. Таким чином, фрейми утворюють мережу. Крім того, існує зв'язок між фреймами типу **АКО** (a kind of), яка вказує на фрейм більш високого рівня ієрархії, звідки неявно успадковуються список і значення слотів. При цьому можливе множинне спадкування — перенесення властивостей від декількох прототипів. Будь який фрейм може бути представлений таким чином:

(*ІМ'Я ФРЕЙМА:(Ім'я 1-го слоту: значення 1-го слоту),(Ім'я 2-го слоту: значення 2-го слоту),...(Ім'я N-го слоту: значення N-го слоту))*.

Табличне представлення слоту виглядає наступним чином (табл. 3.2):

Таблиця 3.2 - Структура фрейму

ІМ'Я РЕЙМУ			
Ім'я слоту	Значення слоту	Спосіб отриманнязначення	Приєднана процедура \Демон

Ім'я фрейму служить для ідентифікації фрейму в системі і має бути унікальним. Фрейм являє собою сукупність слотів, число яких може бути довільним.

Кількість слотів в кожному фреймі встановлюється проектувальником системи, при цьому частина слотів визначається самою системою для виконання специфічних функцій (системні слоти), прикладами яких є: слот- покажчик предка даного фрейма (IS-A), слот-покажчик дочірніх фреймів, слот для введення імені користувача, слот для введення дати визначення фрейму, слот для введення дати зміни фрейму і т.д.

Ім'я слота має бути унікальним в межах фрейму. Зазвичай ім'я слота являє собою ідентифікатор, який наділений певною семантикою. Як ім'я слота може виступати довільний текст.

Наприклад, *<Ім'я слота>* = Головний герой роману Ф. М. Достоєвського

«Ідіот*», <Значення слота>* = Князь Мишкін. Імена системних слотів зазвичай зарезервовані, в різних системах вони можуть мати різні значення.

Значення слота повинно відповідати зазначеному типу даних і умові успадкування.

Спосіб отримання значення визначає, як саме встановлюється значення конкретного слота. Існує кілька способів отримання значень слота (табл. 3.3), вибір способу залежить від властивостей самих даних.

Таблиця 3.3 - Спосіб отримання значень слотів

Спосіб	**Опис**
За замовчуванням від прототипу (предка)	Слоту присвоюється значення, визначене за замовчуванням під фреймі-прототипі, деякі стандартнізначення.
Через успадкування	Відрізняється від першого способу тим, щозначення задано в спеціальному слоті батьківського фрейма, поєднаного з поточнимзв'язком АКО.
За формулою	Слоту призначається формула, результатобчислення якої є значенням слоту.
Через приєднану процедуру	Слоту призначається процедура, що дозволяєотримати значення слота алгорітмічно.
Із зовнішніх джерел даних	При використанні моделі в інтелектуальних системах дані, що є значеннями слотів, можуть надходити з баз даних, від системи датчиків, від користувача.

Демоном називається процедура (табл.3.4), що автоматично запускається при виконанні деякої умови. Демони автоматично запускаються при зверненні до відповідного слоту. Типи демонів пов'язані з умовою запуску процедури. Демон з умовою IF-NEEDED запускається, якщо в момент звернення до слоту його значення не було встановлено. Демон типу IF- ADDED запускається при спробі зміни значення слота. Демон IF-REMOVED запускається при спробі видалення значення слота. Можливі також інші типи демонів. Демон є різновидом пов'язаної процедури.

Таблиця 3.4 - Найбільш поширені демони

Демон	**Подія**	**Опис**
IF-REMOVED	якщо видалено	Виконується, коли інформація видаляється з слота.
IF-ADDED	якщо додано	Виконується, коли нова інформація записується у слот.
IF-NEEDED	на вимогу	Виконується, коли запитується інформація зпустого слота.
IF-DEFAULT	за замовчуванням	Виконується, коли встановлюється значення за замовчуванням.

Приєднана процедура запускається за повідомленням, переданим з іншого фрейму. Демони і приєднані процедури є процедурними знаннями, об'єднанимиразом з декларативними в єдину систему. Ці процедурні знання є засобами управління виводу 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 і часу розробників. Тому проектування фреймових систем виконується, як правило, фахівцями, які мають високий рівень кваліфікації в галузі штучного інтелекту.

Існує кілька видів фреймів, які дозволяють описати предметну область і вирішувати завдання. У табл.3.5 представлені найбільш поширені типи фреймів, вказані типи знань, які вони відображають, а також приклади фреймів даного типу з різних предметних областей

Таблиця 3.5 - Типи фреймів

Тип фрейму	**Тип знання**	**Опис**	**Приклад**
За пізнавальним призначенням			
Фрейми-прототипи (шаблони, зразки)	інтенсіональні	відображають знання про абстрактністереотипних поняттях, які є класами якихось конкретних об'єктів	людина, автомобіль
Фрейми-екземпляри (приклади)	екстенсіональні	відображають знання про конкретніфакти предметноїобласті	Іванченко І. І.,ЗАЗ-2110
За функціональним призначенням			
Фрейми-структури (об'єкти)	декларативні	відображають абстрактні і конкретні предмети і поняття предметної області (містять набір характеристик, що описує об'єкт або поняття)	кредит, застава, вексель, людина, лекція
Фрейми-операції	процедурні	відображають різні процеси перетворення або використання об'єктів предметної області (містять набір характеристик процесу)	процеси отримання кредиту, синтезу пристроїв

Тип фрейму	Тип знання	Опис	Приклад
Фрейми-ситуації	прагматичні	відображають типові ситуації, в яких можуть перебувати фрейми об'єкти і	аварія, тривога,
		фрейми ролі (містять набір характеристик, що ідентифікуютьситуацію)	робочий режим пристрою
Фрейми-сценарії	технологічні	відображають розвиток ситуації, типову структуру для деякої дії, поняття, події, відображає динаміку (містять набір характеристик, дозволяютьзабезпечити розвиток системи за даним сценарієм)	банкрутство, здача іспиту
Фрейми-ролі	функціональні		менеджер, касир, клієнт, студент

Слід зазначити близькість понять, що використовуються у фреймовій і об'єктно-орієнтованої моделях представлення знань, а також в базах даних (таб.3.6).

Таблиця 3.6 - Основні поняття фреймовой і об'єктно-орієнтованої моделей

Фреймова модель	Об’єктно-орієнтованамодель	База даних
Фрейм (фрейм-зразок)	Класс	Таблиця
Фрейм-екземпляр	Об’єкт (екземпляр класу)	Рядок
Слот	Атрибут	Стовбчик
Демон	Подія (з методом її обробки)	Тригер
Приєднана процедура	Метод	Процедура щозберігаєтьс

Переваги і недоліки фреймових моделей.

Переваги фреймових моделей:

- фрейми дозволяють організувати чітко виражену ієрархію знань.
- процедурні вкладення є важливою властивістю фреймів, так як вони дозволяють тісно пов'язати процедурні та декларативні знання про об'єкт, тобтоз'єднати інформаційну, функціональну та поведінкові складові об'єкта в єдине ціле;
- розвинена процедура спадкування значень;
- дозволяють представляти складні об'єкти не у вигляді великої семантичної структури, а у вигляді єдиної сутності.

Недоліком фреймовой моделі є відсутність спеціального механізму управління висновком, у зв'язку з чим, розробники повинні реалізувати даний механізм за допомогою приєднаних процедур

3.3 Приклад виконання роботи

Розглянемо на прикладі предметної області «Ресторан» (відвідування ресторану) та побудуємо фреймову модель.

Опис процесу розв'язання. Для побудови фреймової моделіподання знань необхідно виконати наступні кроки:

1. Визначити абстрактні об'єкти і поняття предметної області, необхідні для вирішення поставленого завдання. Оформити їх у вигляді фреймів-прототипів (фреймів-об'єктів, фреймів-ролей).
2. Задати конкретні об'єкти предметної області. Оформити їх у вигляді фреймів-екземплярів (фреймів-об'єктів, фреймів-ролей).
3. Визначити набір можливих ситуацій. Оформити їх у вигляді фреймів-ситуацій (прототипи). Якщо існують прецеденти по ситуацій в предметної області, додати фрейми-екземпляри (фрейми-ситуації).
4. Описати динаміку розвитку ситуацій (перехід від одних до інших) через набір сцен. Оформити їх у вигляді фреймів-сценаріїв.
5. Додати фрейми-об'єкти сценаріїв і сцен, які відображають дані конкретного завдання.

Розв'язання.

1) Ключові поняття даної предметної області — ресторан, той, хто відвідує ресторан (клієнт) і ті, хто його обслуговують (для простоти обмежимося тільки офіціантами). У обслуговуючого персоналу і клієнтів є спільні характеристики, тому доцільно виділити загальне абстрактне поняття — людина. Тоді фрейми «Ресторан» і «Людина» є прототипами-зразками, а фрейми «Офіціант» і «Клієнт» — прототипами-ролями. Також потрібно визначити основні слоти фреймів — характеристики, що мають значення для розв'язуваної задачі. В таблицях 3.7-3.8 наведемо структуру фрему у вигляді таблиць.

Таблиця 3.7 – Фрейм людина

ЛЮДИНА			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Стать	Чоловіча абожіноча	Із зовнішніхджерел	
Вік	Від 0 до 120 років	Із зовнішніхджерел	

Таблиця 3.8 – Фрейм ресторан

РЕСТОРАН			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Назва		Із зовнішніхджерел	
Адреса		Із зовнішніхджерел	
Час роботи		Із зовнішніхджерел	
Спеціалізація		Із зовнішніхджерел	
Клас	Середній абовищий	Із зовнішніхджерел	

Фрейми-нащадки містять всі слоти своїх батьків, вони явно прописуються тільки в разі зміни будь-якого параметра (таб.3.9-3.10).

Таблиця 3.9 – Фрейм офіціант

ОФІЦІАНТ (*АКО* ЛЮДИНА)			
Ім’я слоту	**Значення слоту**	**Спосіб отриманняданих**	**Демон**
Вік	Від 18 до 55 років	Із зовнішніх джерел	
Стаж роботи		Із зовнішніхджерел	
Заробітна платня		Із зовнішніхджерел	
Графік роботы		Із зовнішніхджерел	
Місто роботы	Фрейм-об’єкт	Із зовнішніхджерел	

Таблиця 3.10 – Фрейм клієнт

КЛІЄНТ (*АКО* ЛЮДИНА)			
Ім’я слоту	**Значення слоту**	**Спосіб отриманняданих**	**Демон**
Вид оплати	Готівка абокартка	За замовчуванням (готівка)	
Статус	Звичайний абоVIP	За замовчуванням (звичайний)	
Форма замовлення	Замовлення є або нема	За замовчуванням (замовлення нема)	
Чайоі		Із зовнішніх джерел	

Фрейми-зразки описують конкретну ситуацію: які ресторани є в місті, як саме організовується відвідування, хто є відвідувачем, хто працює в обраному ресторані та ін..

Тому визначимо такі фрейми-зразки, що є спадкоємцями фреймів-прототипів табл. 3.11-3.13.

Таблиця 3.11 – Фрейм кафе-ресторан «СМАКОТА»

КАФЕ-РЕСТОРАН «СМАКОТА» (*АКО* РЕСТОРАН)			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Назва	Смакота	Із зовнішніхджерел	
Адреса	м. Одеса вул.Буніна 25	Із зовнішніхджерел	
Час роботи	9:00 — 23:00	Із зовнішніхджерел	
Спеціалізація	Піцерия	Із зовнішніхджерел	
Клас	Середній	Із зовнішніхджерел	

Таблиця 3.12 – Фрейм кафе-ресторан «СМАЧНА ЇЖА»

КАФЕ-РЕСТОРАН «СМАЧНА ЇЖА» (*АКО* РЕСТОРАН)			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Назва	Смачна їжа	Із зовнішніхджерел	
Адреса	м. Одеса вул. Грушевського 14	Із зовнішніхджерел	
Час роботи	9:00 — 23:00	Із зовнішніхджерел	
Спеціалізація	Паб	Із зовнішніх джерел	
Клас	Вищій	Із зовнішніх джерел	
СЕРГІЙ (*АКО* ОФІЦІАНТ)			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Вік	27	Із зовнішніхджерел	
Стать	Чоловіча	Із зовнішніхджерел	
Стаж роботи	5	Із зовнішніхджерел	
Заробітна платня	7000	Із зовнішніхджерел	
Графік роботи	Через день з 18:00 дд 23:00	Із зовнішніхджерел	
Місце роботи	**КАФЕ «СМАЧНАЇЖА»**	Із зовнішніхджерел	

Таблиця 3.13 – Фрейм офіціант

МАРИНА (*АКО* ОФІЦІАНТ)			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Вік	24	Із зовнішніхджерел	
Стать	Жіноча	Із зовнішніхджерел	
Стаж роботи	2	Із зовнішніхджерел	
Заробітна платня	8200	Із зовнішніхджерел	
Графік роботи	Кожного дня з 9:00 до 14:00	Із зовнішніхджерел	
Місце роботи	Кафе-ресторан «Смакота»	Із зовнішніхджерел	
ПЕТРО (*АКО* КЛІЄНТ)			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Стать	Чоловіча	Із зовнішніхджерел	
Вік	19	Із зовнішніхджерел	
Спосіб оплати	Готівка	За замовчуванням (готівка)	
Статус	Звичайний	За замовчуванням (звичайний)	
Форма замовлення	Замовлення немає	За замовчуванням (замовлення немає)	
Чайові	7% від суми замовлення	Із зовнішніхджерел	

Фрейми-ситуації описують можливі ситуації. У ресторані клієнт

потрапляє вкілька типові ситуацій: замовлення і оплата.

Можливі й інші не типові ситуації: клієнт подавився, у клієнта немає готівки для оплати рахунку і т.д. (таб.3.14).

Таблиця 3.14 – Фрейм замовлення

ЗАМОВЛЕННЯ			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Перелік блюд		Із зовнішніхджерел	IF-ADDED (змінює слот «Перелік цін»)
Перелік цін		Приєднана процедура	IF-ADDED (змінює слот «Сума замовник»)
Сума замовлення		Приєднана процедура	
Прийняв замовлення	Фрейм-зразок	Із зовнішніхджерел	
Зробив замовлення	Фрейм-зразок	Із зовнішніхджерел	

Ситуації виникають після настання якихось подій, виконання умов і можуть слідувати одна за одною. Динаміку предметної області можна відобразити в фреймах-сценаріях. Їх може бути безліч, нехай в рамках нашого завдання Петро відвідав ресторан «Смачна їжа». Тоді фрейми будуть

заповнені як в таб.3.15.

Таблиця 3.15 – Фрейм «Смачна їжа»

ВІДВІДУВАННЯ «СМАЧНОЇ ЇЖІ» **(**АКО **ВІДВІДУВАННЯ РЕСТОРАНУ)**			
Ім’я слоту	**Значення слоту**	**Спосіб Отримання даних**	**Демон**
Відвідувач	**ПЕТРО**	Із зовнішніхджерел	
Ресторан	**КАФЕ «СМАЧНА ЇЖА»**	Із зовнішніх джерел	
Офіціант	**СЕРГІЙ**	Приєднана процедура (визначає по обраному ресторану)	IF-ADDED, IF-REMOVED (змінює слот «Офіціант»)
Сцена 1	Вхід, вибір	Із зовнішніхджерел	
Сцена 2	**ЗАМОВЛЕННЯ ПЕТРА**	Із зовнішніхджерел	
Сцена 3	Прийом єжі	Із зовнішніхджерел	
Сцена 4	**СПЛАТАПЕТРА**	Із зовнішніхджерел	
Сцена 5	Вихід	Із зовнішніхджерел	

ЗАМОВЛЕННЯ ПЕТРА (АКО **ЗАМОВЛЕННЯ)**			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**

Перелік блюд	Відбивна, темне пиво	Із зовнішніхджерел	IF-ADDED (змінює слот «Перелі кцін»)
Перелік цін	25; 7	Приєднана процедура	IF-ADDED (змінює слот «Сума замовник»)
Сума замовлення	32	Приєднана процедура	
Прийняв замовлення	**СЕРГІЙ**	Із зовнішніхджерел	
Зробив замовлення	**ПЕТРО**	Із зовнішніхджерел	

Взаємозв'язок різних видів фреймів відображається графічно у вигляді графу (рис.3.1).

Використання фреймової моделі аналогічно семантичної, тільки в процесі отримання відповіді крім вершин враховуються і слоти. Наприклад, отримати відповідь на питання «Хто працює офіціантом в ресторані "Смачна їжа"?» можна наступним чином: із запиту зрозуміло, що необхідно знайти фрейм

«Ресторан "Смачна їжа"» і простежити зв'язок з фреймом «Сергій», який є нащадком фрейму «Офіціант».

Також можна знайти слот «Місце роботи» і перевіривши його значення у фреймах спадкоємців фрейму «Офіціант» визначити, що офіціантом в ресторані "Смачна їжа" працює Сергій.

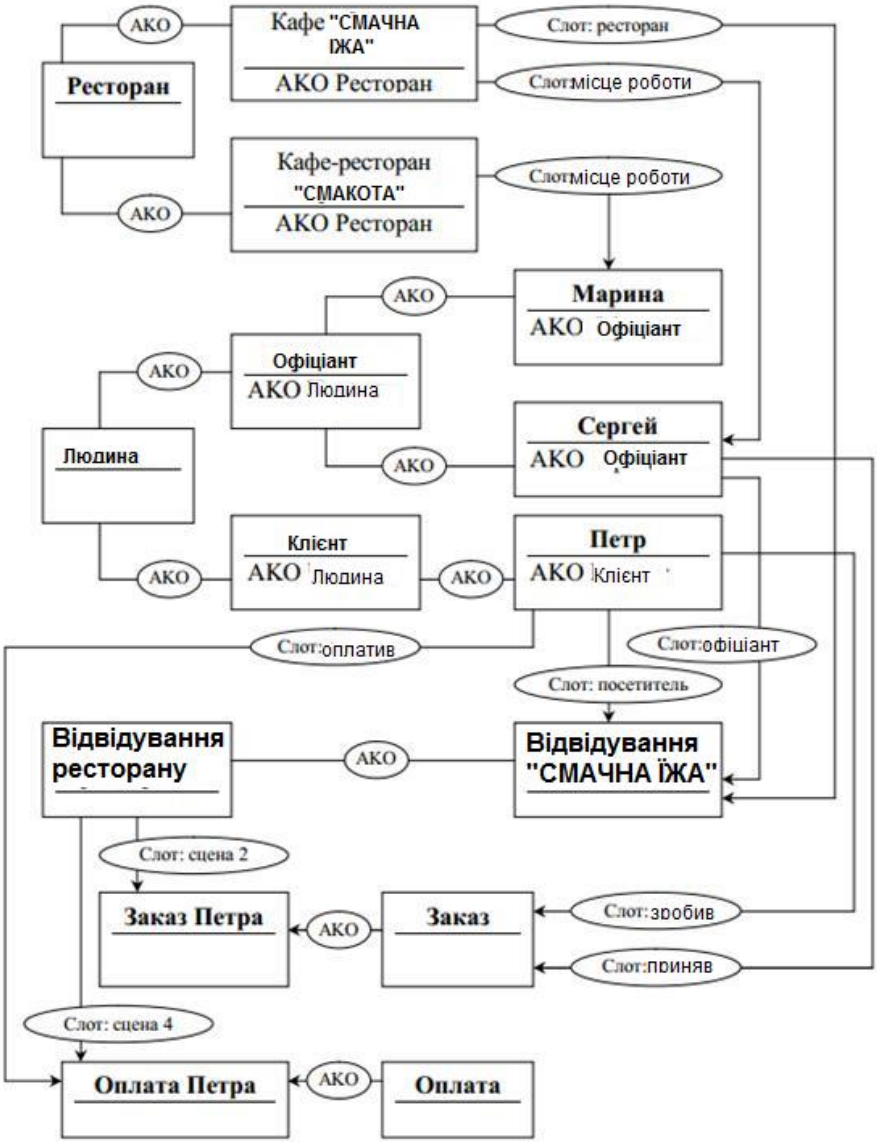


Рисунок 3.1 – Схема фреймів для предметної області «Ресторан»

3.4 Контрольні питання

1. Що називають «фреймовою моделлю»?
2. З яких елементів складається фреймова модель?
3. Які існують класифікації фреймів?
4. Які типи відносин підтримуються фреймовими моделями?
5. У чому переваги і недоліки фреймової моделі?

ЛАБОРАТОРНА РОБОТА №4
АНАЛІЗ ТА СОРТУВАННЯ ДАНИХ НА ОСНОВІ БІБЛІОТЕЦІ PANDAS

4.1 Мета роботи

Навчитися користуватися бібліотекою Pandas та її вбудованими об'єктами для аналізу даних у датасетах.

4.2 Завдання до лабораторної роботи.

Частина 1. Використовуйте бібліотеку Pandas в середовищі Python виконайте завдання щодо попередньої обробки даних.

1. Створіть власну датафрейм за варіантом (таб.4.1), де № варіанту відповідає порядковому номеру вашої ПІБ у списку групи вибіркової дисципліни.

Таблиця 4.1 – Індивідуальне завдання до частини 1

№	Завдання
1	Створити датафрейм для 5 міст Дніпропетровської області.
2	Створити датафрейм для 5 міст Полтавської області.
3	Створити датафрейм для 5 міст Харьківської області.
4	Створити датафрейм для 5 міст Запорізької області.
5	Створити датафрейм для 5 міст Сумської області.
6	Створити датафрейм для 5 міст Чернігівської області.
7	Створити датафрейм для 5 міст Київської області.
8	Створити датафрейм для 5 міст Черкаської області.
9	Створити датафрейм для 5 міст Кіровоградської області.
10	Створити датафрейм для 5 міст Хмельникої області.
11	Створити датафрейм для 5 міст Рівненської області.
12	Створити датафрейм для 5 міст Львівської області.
13	Створити датафрейм для 5 міст Ужгородської області.
14	Створити датафрейм для 5 міст Івано-Франківської області.
15	Створити датафрейм для 5 міст ХОдеської області.

2. Додати чисельність населення до стовреної датафрейм в п.1.

3. Представити як список із словника та список списків. Чи змінилось представлення даних? Що саме змінено?

4. Виконати транспонування.

Частина 2. Створення дата фрейм із файлу.

1. Необхідно скачати файл у форматі cvs із сайту https://vizhub.healthdata.org/gbd-results/ (рис.4.1). Інститут оцінки показників здоров'я США.

Рисунок 4.1 – Сайт файлу дата фрейм

2. З сайту обрати параметри за варіантом (таб.4.2), де № варіанту відповідає порядковому номеру вашої ПІБ у списку групи вибіркової дисципліни.

Таблиця 4.2 – Індивідуальне завдання до частини 2

№	Завдання
1	Location: Central Asia Year: 2000-2019 Content: Cause Age: All ages Metric: Rate Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes
2	Location: Central Europe Year: 2000-2019 Content: Cause Age: All ages Metric: Rate Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes
3	Location: Eastern Europe Year: 2000-2019 Content: Cause Age: All ages Metric: Rate Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes
4	Location: High-income Year: 2000-2019 Content: Cause Age: All ages

	Metric: Rate Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes
5	Location: Australasia Year: 2000-2019 Content: Cause Age: All ages Metric: Rate Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes
6	Location: High-income Asia Pacific Year: 2000-2019 Content: Cause Age: All ages Metric: Rate Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes
7	Location: Japan Year: 2000-2019 Content: Cause Age: All ages Metric: Rate Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes
8	Location: High-income North America Year: 2000-2019 Content: Cause Age: All ages Metric: Rate Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes
9	Location: Southern Latin America Year: 2000-2019 Content: Cause Age: All ages Metric: Rate

	Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes
10	Location: Western Europe Year: 2000-2019 Content: Cause Age: All ages Metric: Rate Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes
11	Location: Latin America and Caribbean Year: 2000-2019 Content: Cause Age: All ages Metric: Rate Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes
12	Location: North Affrica and Middle East Year: 2000-2019 Content: Cause Age: All ages Metric: Rate Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes
13	Location: Central Sub-Saharan Africa Year: 2000-2019 Content: Cause Age: All ages Metric: Rate Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes
14	Location: Eastern Sub-Saharan Africa Year: 2000-2019 Content: Cause Age: All ages Metric: Rate Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes
15	Location: Southern Sub-Saharan Africa Year: 2000-2019 Content: Cause

	Age: All ages Metric: Rate Measure: Deaths Sex: Male, Female, Both Cause: level 2 causes

3. Завантажити скачаний файл в датафрейм на основі метода метода *pd.read_csv().*

4. Виведіть кількість рядків та стовпців вашого файлу.

5. Виведіть статистику за числовими стовпцями вашого файлу: середнє, максимум і мінімум, квартили, стандартне відхилення (*Метод describe*).

6. Вивести показники смертності тільки для чол.(наведіть скрін), тільки для жін.(наведіть скрін) та - Both sexes (наведіть скрін).

7. Вивести показники вашого датафрейму за варіантом (таб.4.3), де № варіанту відповідає порядковому номеру вашої ПІБ у списку групи вибіркової дисципліни.

Таблиця 4.3 – Індивідуальне завдання до частини 2

№	Завдання
1	лише 2019 рік, лише серцево-судинні захворювання.
2	період за 5 років, лише серцево-судинні захворювання.
3	лише 2019 рік, лише захворювання Туберкульозу .
4	період за 5 років, лише захворювання Туберкульозу
5	лише 2019 рік, лише інфекції нижніх дихальних шляхів
6	період за 5 років, лише інфекції нижніх дихальних шляхів
7	лише 2019 рік, лише інфекції верхніх дихальних шляхів
8	період за 10 років, лише інфекції верхніх дихальних шляхів
9	лише 2019 рік, лише захворювання Ебола
10	період за 10 років, лише захворювання Ебола
11	лише 2019 рік, лише захворювання Zika virus
12	період за 20 років, лише захворювання Zika virus
13	лише 2019 рік, лише захворювання астми
14	період за 25 років, лише захворювання астми
15	період за 6 років, лише захворювання діабету

8. На основі завдання п.7 додати умову лише понад 600 смертей на 100 тисяч жителів та вивести результат. Якщо за заданою умовою результатів немає, то спробувати змінити умову.

9. Зберегти вибірки.

10. Побудувати графік смертності на основі даних датафрейму.

Частина 3. Додаткова для отримання оцінки 96-100 балів.

1. Виконати аналіз даних вашого індивідуального дата фрейму за

параметрами: максимальна народжуваність в якому була році; максимальна смертність за хворобами. Побудувати графіки.

2. Отримати у викладача файл дата фрейм та провести аналіз даних та вивести:

– кількість жін. та чол. за даним файлом;

– середній вік чол.;

– кількість жителів Таїланду (у %) відповідно до наданих даних;

– гістограму розподілу освіти за ОКР людей за варіантом (таб.4.4), де № варіанту відповідає порядковому номеру вашої ПІБ у списку групи вибіркової дисципліни;

– середні та середньоквадратичні відхилення віку тих, хто отримує більше 50K на рік (ознака s*alary*) та тих, хто отримує менше 50K на рік.

Таблиця 4.4 – Індивідуальне завдання до частини 3

№	Завдання
1	Бакалавр
2	Магістри
3	Навчання на базі 11 класів
4	Навчання в коледжі
5	Навчання в аспірантурі
6	Навчання в докторнатурі
7	Професійні школи

4.3Теоретична частина.

4.3.1 Можливості бібліотеки Pandas

Pandas — програмна бібліотека, написана для мови програмування Python для маніпулювання даними та їхнього аналізу. Вона, зокрема, пропонує структури даних та операції для маніпулювання чисельними таблицями та часовими рядами.

Основними можливостями є:

1. Об'єкт DataFrame із вбудованим індексуванням для маніпулювання даними.

2. Інструменти для зчитування та записування даних між структурами даних у пам'яті та різними форматами файлів.

3. Вирівнювання даних та вбудована підтримка пропущених даних.

4. Переформатовування для отримання зведених наборів даних.

5. Отримання зрізів за мітками, індексування з розширеними можливостями[6] та отримання піднаборів з великих наборів даних.

6. Вставляння та вилучення стовпчиків у структурах даних.

7. Рушій групування, що дозволяє робити з наборами даних операції розділення-зміни-об'єднання (англ. split-apply-combine).

8. Злиття та з'єднання наборів даних.

9. Ієрархічне індексування осей для роботи з даними високої

вимірності в структурі даних нижчої вимірності.

10. Функціональність для часових рядів: породження діапазонів дат та перетворення частоти, статистики рухливого вікна, лінійні регресії рухливого вікна, зсування дат та запізнювання.

Наприклад, щоб прочитати дані з CSV, у стандартному Python треба спочатку вирішити, як зберігати дані, потім відкрити файл, прочитати його рядково, відокремити значення один від одного і очистити дані від спеціальних символів.

```
> with open('file.csv') as f:
... content = f.readlines()
... content = [x.split(',').replace('\n','') for x in content]
```

У Pandas все простіше. По-перше, не треба думати, як зберігатимуться дані — вони лежать у датафреймі. По-друге, достатньо написати одну команду:

```
> data = pd.read_csv('file.csv')
```

Pandas додає в Python нові структури даних - серії та датафрейми.

Підключення бібліотеки можливо в Anaconda або PyPI або Linux (таб.4.1).

Таблиця 4.1 – Підключення бібліотеки Pandas

Команда	Середовище
conda install pandas	Anaconda
pip install pandas	PyPI
sudo apt-get install python-pandas	Linux (Debian та Ubuntu)
zypper in python-pandas	Linux (OpenSuse та Fedora)

4.3.2 Структури даних

У основі Pandas лежить DataFrame (рис.4.1) – структура даних табличного типу. Будь-яке табличне подання даних, наприклад, електронні таблиці або бази даних, можна використовувати як DataFrame. Об'єкт DataFrame складається з об'єктів Series - одновимірних масивів, об'єднаних під однією назвою та типом даних. Series можна розглядати як стовпець таблиці.

Рисунок 4.1 – Представлення DataFrame та Series

4.32.1 Серії

Серії - одновимірні масиви даних. Вони дуже схожі на списки, але відрізняються за поведінкою, наприклад, операції застосовуються до списку цілком, а в серіях поелементно.

Тобто якщо список помножити на 2, отримайте той же список, повторений 2 рази.

```
> vector = [1, 2, 3]
> vector * 2
[1, 2, 3, 1, 2, 3]
```

А якщо помножити серію, її довжина не зміниться, а елементи подвоїться.

```
> import pandas as pd
> series = pd.Series([1, 2, 3])
> series * 2
0 2
1 4
2 6
dtype: int64
```

Зверніть увагу на перший стовпчик виводу. Це індекс, де зберігаються адреси кожного елемента серії. Кожен елемент потім можна отримувати, звернувшись на потрібну адресу.

```
> series = pd.Series(['foo', 'bar'])
> series[0]
'foo'
```

Ще одна відмінність серій від списків - як індекси можна використовувати довільні значення, це робить дані наочнішими. Припустимо, що ми аналізуємо помісячні продажі. Використовуємо як індекси назви місяців, значеннями буде виручка:

```
> months = ['jan', 'feb', 'mar', 'apr']
> sales = [100, 200, 300, 400]
> data = pd.Series(data=sales, index=months)
> data
jan 100
feb 200
бер 300
apr 400
dtype: int64
```
Тепер можемо набувати значення кожного місяця:

```
> data['feb']
200
```

Так як серії - одновимірний масив даних, у них зручно зберігати виміри по одному. На практиці зручніше групувати дані разом. Наприклад, якщо ми аналізуємо помісячні продажі, корисно бачити не лише виручку, а й кількість проданих товарів, кількість нових клієнтів та середній чек. Для цього чудово підходять датафрейми.

4.32.2 Датафрейми

Датафрейми це таблиці. У них є рядки, колонки та осередки.

Технічно, колонки датафреймів – це серії. Оскільки в колонках зазвичай описують одні й самі об'єкти, то всі колонки ділять один і той же індекс:

```
> months = ['jan', 'feb', 'mar', 'apr']
> sales = {
... 'Revenue': [100, 200, 300, 400],
... 'items_sold': [23, 43, 55, 65],
... 'new_clients': [10, 20, 30, 40]
...}
> sales_df = pd.DataFrame(data=sales, index=months)
> sales_df
    revenue items_sold new_clients
jan 100 23 10
feb 200 43 20
бер 300 55 30
apr 400 65 40
```

Розглянемо, як створювати датафрейми та завантажувати в них дані.

4.3.3. Створення датафрейми та завантаження даних

Буває, що ми не знаємо, що являють собою дані, і не можемо задати

структуру заздалегідь. Тоді зручно створити порожній датафрейм і потім заповнити його даними.

```
> df = pd.DataFrame()
```

А іноді дані вже є, але зберігаються у змінній зі стандартного Python, наприклад, у словнику. Щоб отримати датафрейм, цю змінну передаємо в ту саму команду:

```
> df = pd.DataFrame(data=sales, index=months))
```

Трапляється, що в деяких записах не вистачає даних. Наприклад, подивіться на список goods_sold — у ньому продаж, розбитий за товарними категоріями. За перший місяць ми продали машини, комп'ютери та програмне забезпечення. У другому машин немає, зате з'явилися велосипеди, а третьому знову з'явилися машини, але велосипеди зникли:

```
> goods_sold = [
... {'computers': 10, 'cars': 1, 'soft': 3},
... {'computers': 4, 'soft': 5, 'bicycles': 1},
... {'computers': 6, 'cars': 2, 'soft': 3}
... ]
```

Якщо завантажити дані до датафрейму, Pandas створить колонки для всіх товарних категорій і, де це можливо, заповнить їх даними:

```
> pd.DataFrame(goods_sold)
   bicycles cars computers soft
0 NaN 1.0 10 3
1 1.0 NaN 4 5
2 NaN 2.0 6 3
```

Зверніть увагу, продажі велосипедів у першому та третьому місяці рівні *NaN* – розшифровується як **Not a Number**. Так Pandas позначає відсутні значення.

Тепер розберемо, як завантажувати дані із файлів. Найчастіше дані зберігаються в еселівських таблицях або csv-, tsv-файлах.

Екселівські таблиці читаються за допомогою pd.read_excel(). Параметрами потрібно передати адресу файлу на комп'ютері та назву аркуша, який потрібно прочитати. Команда працює як з xls, так і з xlsx:

```
> pd.read_excel('file.xlsx', sheet_name='Sheet1')
```

Файли формату csv і tsv — це текстові файли, в яких дані відокремлені один від одного комами або табуляцією:

```
# CSV
month,customers,sales
feb,10,200
```

```
#TSV
month\tcustomers\tsales
feb t10 t200
```

Обидва читаються за допомогою команди .read_csv(), символ табуляції передається параметром sep (від англ. Separator - роздільник):

```
> pd.read_csv('file.csv')
> pd.read_csv('file.tsv', sep='\t')
```

Під час завантаження можна призначити стовпець, який буде індексом. Уявіть, що ми завантажуємо таблицю із замовленнями. Кожне замовлення має свій унікальний номер. Якщо призначимо цей номер індексом, зможемо вивантажувати дані командою df[order_id]. Інакше доведеться писати фільтр df[df['id'] == order_id].

Про те, як отримувати дані з датафреймів, я розповім в одному з наступних розділів. Щоб призначити колонку індексом, додамо в команду read_csv() параметр index_col, що дорівнює назві потрібної колонки:

```
> pd.read_csv('file.csv', index_col='id')
```

Після завантаження даних у датафрейм, добре їх досліджувати — особливо, якщо вони вам незнайомі.

4.3.4. Робота з завантаженими даними

Припустимо, що ми аналізуємо продажі американського інтернет-магазину. У нас є дані про замовлення та клієнтів. Завантажимо файл з продажем інтернет-магазину в змінну *orders.* Якщо завантажуємо замовлення, вкажемо, що колонка id піде в індекс:

```
> orders = pd.read_csv('orders.csv', index_col='id')
```

Як правило, у будь-якого датафрейму є чотири атрибути: .shape, .columns, .index і .dtypes.

.shape показує, скільки в датафреймі рядків та колонок. Він повертає пару значень (n_rows, n_columns). Спочатку йдуть рядки, потім колонки.

```
> orders.shape
(5009, 5)
```

У датафреймі 5009 рядків та 5 колонок.

Окей, масштаб оцінили. Тепер подивимося, яка інформація міститься у кожній колонці. За допомогою .columns дізнаємося назви колонок:

```
> orders.columns
Index(['order_date', 'ship_mode', 'customer_id', 'sales'], dtype='object')
```

Тепер бачимо, що у таблиці є дата замовлення, метод доставки, номер клієнта та виручка.

За допомогою *.dtypes* дізнаємося типи даних, що знаходяться в кожній колонці та зрозуміємо, чи треба їх обробляти. Буває, що цифри завантажуються як тексту. Якщо ми спробуємо скласти дві текстові значення '1' + '1', то отримаємо не число 2, а рядок '11':

```
> orders.dtypes
order_date object
ship_mode об'єкт
customer_id object
sales float64
dtype: object
```

Тип object – це текст, float64 – це дробове число типу 3,14.

За допомогою атрибута *.index* подивимося, як називаються рядки:

```
> orders.index
Int64Index([100006, 100090, 100293, 100328, 100363, 100391, 100678, 100706,
        100762, 100860,
        ...
        167570, 167920, 168116, 168613, 168690, 168802, 169320, 169488,
        169502, 169551],
       dtype='int64', name='id', length=5009)
```

Очікується, в індексі датафрейму номери замовлень: 100762, 100860 і таке інше.

У колонці sales зберігається вартість кожного проданого товару. Щоб дізнатися про розкид значень, середню вартість і медіану, використовуємо метод .describe():

```
> orders.describe()
        sales
count 5009.0
mean 458.6
std 954.7
min 0.6
25% 37.6
50% 152.0
```

```
75% 512.1
max 23661.2
```

Нарешті, щоб побачити кілька прикладів записів датафрейма, використовуємо команди .head() і .sample(). Перша повертає 6 записів із початку датафрейму. Друга - 6 випадкових записів:

```
> orders.head()
      order_date ship_mode customer_id sales
id
100006 2014-09-07 Standard DK-13375 377.970
100090 2014-07-08 Standard EB-13705 699.192
100293 2014-03-14 Standard NF-18475 91.056
100328 2014-01-28 Standard JC-15340 3.928
100363 2014-04-08 Standard JM-15655 21.376
```

Отримавши перше уявлення про датафрейми, тепер обговоримо, як працювати з даними.

4.3.5. Обробка даних з дата фреймів. Запити.

Дані з датафреймів можна отримувати по-різному: вказавши номери колонок і рядків, використовуючи умовні оператори або мову запитів. Розкажу докладніше про кожний спосіб.

4.3.5.1 Вказуємо потрібні рядки та колонки

Продовжуємо аналізувати продажі інтернет-магазину, які завантажили у попередньому розділі. Допустимо, я хочу вивести стовпець sales. Для цього назву стовпця потрібно укласти у квадратні дужки та поставити після них назви датафрейму: orders['sales']:

```
> orders['sales']
id
100006 377.970
100090 699.192
100293 91.056
100328 3.928
100363 21.376
100391 14.620
100678 697.074
100706 129.440
...
```

Зверніть увагу, результат команди – новий датафрейм з таким самим індексом.

Якщо потрібно вивести кілька стовпців, у квадратні дужки потрібно вставити список із їх назвами: orders[['customer_id', 'sales']]. Будьте уважні:

квадратні дужки стали подвійними. Перші – від датафрейму, другі – від списку:

```
> orders[['customer_id', 'sales']]
      customer_id sales
id
100006 DK-13375 377.970
100090 EB-13705 699.192
100293 NF-18475 91.056
100328 JC-15340 3.928
100363 JM-15655 21.376
100391 BW-11065 14.620
100363 KM-16720 697.074
100706 LE-16810 129.440
...
```

Перейдемо до рядків. Їх можна фільтрувати за індексом та по порядку. Наприклад, ми хочемо вивести тільки замовлення 100363, 100391 та 100706, для цього є команда .loc[]:

```
> show_these_orders = ['100363', '100363', '100706']
> orders.loc[show_these_orders]
       order_date ship_mode customer_id sales
id
100363 2014-04-08 Standard JM-15655 21.376
100363 2014-04-08 Standard JM-15655 21.376
100706 2014-12-16 Second LE-16810 129.440
```

А в інший раз буває потрібно дістати просто замовлення з 1 по 3 по порядку, незалежно від їх номерів у таблиці. Тоді використовують команду .iloc[]:

```
> show_these_orders = [1, 2, 3]
> orders.iloc[show_these_orders]
       order_date ship_mode customer_id sales
id
100090 2014-04-08 Standard JM-15655 21.376
100293 2014-04-08 Standard JM-15655 21.376
100328 2014-12-16 Second LE-16810 129.440
```

Можна фільтрувати датафрейми по колонках та стовпцях одночасно:

```
> columns = ['customer_id', 'sales']
> rows = ['100363', '100363', '100706']
> orders.loc[rows][columns]
      customer_id sales
id
```

```
100363 JM-15655 21.376
100363 JM-15655 21.376
100706 LE-16810 129.440
...
```

Часто ви не знаєте наперед номерів замовлень, які вам потрібні. Наприклад, якщо завдання отримати замовлення, вартістю понад 1000 рублів. Це завдання зручно вирішувати за допомогою умовних операторів.

4.3.5.2 Запити. Умовні оператори

Завдання: потрібно дізнатися, звідки надходять найбільші замовлення. Почнемо з того, що дістанемо всі покупки вартістю понад 1000 доларів:

```
> filter_large = orders['sales'] > 1000
> orders.loc[filter_slarge]
      order_date ship_mode customer_id sales
id
101931 2014-10-28 First TS-21370 1252.602
102673 2014-11-01 Standard KH-16630 1044.440
102988 2014-04-05 Second GM-14695 4251.920
103100 2014-12-20 First AB-10105 1107.660
103310 2014-05-10 Standard GM-14680 1769.784
...
```

Пам'ятаєте, на початку статті я згадував, що у серіях всі операції застосовуються по-елементно? Так ось, операція orders['sales'] > 1000 йде по кожному елементу серії і якщо умова виконується, повертає True. Якщо не виконується – False. Серію, що вийшла, ми зберігаємо в змінну filter_large.

Друга команда фільтрує рядки датафрейму за допомогою серії. Якщо елемент filter_large дорівнює True, замовлення з'явиться, якщо False – ні. Результат — датафрейм із замовленнями вартістю понад 1000 доларів.

Цікаво, скільки найдорожчих замовлень було доставлено першим класом? Додамо до фільтра ще одну умову:

```
> filter_large = df['sales'] > 1000
> filter_first_class = orders['ship_mode'] == 'First'
> orders.loc[filter_large & filter_first_class]
      order_date ship_mode customer_id sales
id
101931 2014-10-28 First TS-21370 1252.602
103100 2014-12-20 First AB-10105 1107.660
106726 2014-12-06 First RS-19765 1261.330
112158 2014-12-02 First DP-13165 1050.600
116666 2014-05-08 First KT-16480 1799.970
...
```

Логіка не змінилася. У змінну filter_large зберегли серію, яка задовольняє умову orders['sales'] > 1000. У filter_first_class — серію, яка задовольняє orders['ship_mode'] == 'First'.

Потім об'єднали обидві серії за допомогою логічного І: filter_first_class & filter_first_class. Отримали нову серію тієї ж довжини, в елементах якої True тільки у замовлень вартістю понад 1000, доставлених першим класом. Таких умов може бути скільки завгодно.

4.3.6. Метрики

Продовжуємо розглядати на прикладі. Завдання: порахуємо, скільки грошей магазин заробив за допомогою кожного класу доставки. Почнемо з простого - підсумуємо виторг з усіх замовлень. Для цього використовуємо метод .sum():

```
> orders['sales'].sum()
2297200.8603000003
```

Додамо клас доставки. Перед підсумовуванням згрупуємо дані за допомогою методу .groupby():

```
> orders.groupby('ship_mode')['sales'].sum()
ship_mode
First 3.514284e+05
Same Day 1.283631e+05
Second 4.591936e+05
Standard 1.358216e+06
```

3.514284e+05 – науковий формат виведення чисел. Означає 3.51*105. Нам така точність не потрібна, тому можемо сказати Pandas, щоб округляти значення до сотих:

```
> pd.options.display.float_format = '{:,.1f}'.format
> orders.groupby('ship_mode')['sales'].sum()
ship_mode
First 351,428.4
Same Day 128,363.1
Second 459,193.6
Standard 1,358,215.7
```

Тепер бачимо суму виручки за кожним класом доставки. За сумарним виторгом неясно, стає краще чи гірше. Додамо розбивку за датами замовлення:

```
> orders.groupby(['ship_mode', 'order_date'])['sales'].sum()
ship_mode order_date
First 2014-01-06 12.8
```

```
        2014-01-11 9.9
        2014-01-14 62.0
        2014-01-15 149.9
        2014-01-19 378.6
        2014-01-26 152.6
...
```

Видно, що виручка стрибає з кожним днем: іноді 10 доларів, а іноді 378. Цікаво, це змінюється кількість замовлень або середній чек? Додамо до вибірки кількість замовлень. Для цього замість .sum() використовуємо метод .agg(), який передамо список з назвами потрібних функцій.

```
> orders.groupby(['ship_mode', 'order_date'])['sales'].agg(['sum', 'count'])
                    sum count
ship_mode order_date
First 2014-01-06 12.8 1
        2014-01-11 9.9 1
        2014-01-14 62.0 1
        2014-01-15 149.9 1
        2014-01-19 378.6 1
        2014-01-26 152.6 1
...
```

Виходить, що це так стрибає середній чек. Цікаво, а який був найвдаліший день? Щоб дізнатися, відсортуємо датафрейм, що вийшов: виведемо 10 самих грошових днів за виручкою:

```
>orders.groupby(['ship_mode',
'order_date'])['sales'].agg(['sum']).sort_values(by='sum', ascending=False).head(10)
                      sum
ship_mode order_date
Standard 2014-03-18 26,908.4
        2016-10-02 18,398.2
First 2017-03-23 14,299.1
Standard 2014-09-08 14,060.4
First 2017-10-22 13,716.5
Standard 2016-12-17 12,185.1
        2017-11-17 12,112.5
        2015-09-17 11,467.6
        2016-05-23 10,561.0
        2014-09-23 10,478.6
```

Команда розрослася, і тепер її незручно читати. Щоб спростити, можна розбити на кілька рядків. Наприкінці кожного рядка ставимо **зворотний слєш**:

```
> orders \
... .groupby(['ship_mode', 'order_date'])['sales'] \
... .agg(['sum']) \
... .sort_values(by='sum', ascending=False) \
... .head(10)
                        sum
ship_mode order_date
Standard 2014-03-18 26,908.4
         2016-10-02 18,398.2
First 2017-03-23 14,299.1
Standard 2014-09-08 14,060.4
First 2017-10-22 13,716.5
Standard 2016-12-17 12,185.1
         2017-11-17 12,112.5
         2015-09-17 11,467.6
         2016-05-23 10,561.0
         2014-09-23 10,478.6
```

Найвдалішого дня — 18 березня 2014 року — магазин заробив 27 тисяч доларів за допомогою стандартного класу доставки. Цікаво, звідки були клієнти, котрі зробили ці замовлення? Щоб дізнатися, треба поєднати дані про замовлення з даними клієнтів.

4.3.7 Методи Pandas DataFrame

4.3.7.1 Drop

Метод *del df['col_name']* видаляє колонку DataFrame, що має імена у вигляді col_name.

```
drop(self, labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')
```

Метод drop видаляє зазначені мітки з рядків чи стовпців.

labels може бути одиночною міткою або списком міток або стовпців, які потрібно опустити.

axis визначає, чи видаляються мітки з індексу/рядка (0 або *index*) або стовпця (1 або c*olumns*).

4.3.7.2 Rename ()

Іноді ми хочемо перейменувати стовпці та індекси в об'єкті PandaFrame PandaS. Ми можемо використовувати функцію ***Pandas DataFrame Rename ()*** для перейменування стовпців та індексів. Він підтримує такі параметри.

Mapper : словник або функція для застосування до стовпців та індексів. Параметр "AXIS" визначає цільову вісь - стовпці або індекси.

Індекс : Має бути словник або функція, щоб змінити імена індексу.

Колони : Має бути словник або функція, щоб змінити імена стовпців.

Вісь : Може бути int або рядок. Він використовується з параметром Mapper, щоб визначити цільову вісь. Допустимі значення (індекс, стовпці) або число (0, 1). Значення за умовчанням є індексом.

У приміщенні: Якщо true, файл dataframe змінюється. В іншому випадку повертається новий dataframe і поточний dataframe залишається без змін. Значення за промовчанням неправильне".

Рівень: Може бути INT або ім'я рівня. Він використовується у випадку багатоіндексу, тільки перейменовує етикетки на вказаному рівні.

Помилки: Можливі значення («ігноруйте», «Rising»), за умовчанням «ігнорувати». Якщо вказано як "ROING", то keyError піднято, коли піднімається "Mapper", "Index" або "стовпці" або "стовпці", які не присутні в перетворенні індексу. Якщо "ігнорувати", існуючі ключі будуть перейменовані, а додаткові ключі будуть ігноруватися.

```
df1 = df.rename(mapper={'Name': 'EmpName', 'ID': 'EmpID', 'Role': 'EmpRole'},
            axis='columns')  # axis=1 corresponds to columns
```

4.3.7.3 Збереження даних

Ми часто стикаємося з ситуаціями, коли нам потрібно зберегти величезні дані, створені в результаті списання або аналізу, у легкій та читаній, а не загальнодоступній формі.

Тепер ми можемо зробити це, зберігши кадр даних у файлі csv.

```
dataframe.to_csv('file.csv')
```

4.3.7.4 Describe ()

Метод *Pandas DataFrame describe ()* використовується для надання всієї необхідної інформації про набір даних, які надалі можна використовувати для аналізу даних та отримання різних математичних припущень для подальшого вивчення. Тобто показує статистику за всіма числовими стовпцями.

```
DataFrame.describe(percentiles= None, include= None, exclude=None)
```

4.3.7.5 Read_CSV()

Метод Pandas Read_CSV() використовується для читання файлу CSV в об'єкт dataframe. CSV-файл схожий на двовимірну таблицю, де значення розділені за допомогою роздільника.

```
df = pandas.read_csv('employees.csv')
```

4.3.7.6 Tail()

Функція tail() використовується для отримання останніх n рядків.

Ця функція повертає останні n рядків з об'єкта на основі позиції. Це корисно для швидкої перевірки даних, наприклад, після сортування або додавання рядків.

```
DataFrame.tail(self, n=5)
```
Приклад
```
df = pd.DataFrame({'animal':['snake', 'bat', 'tiger', 'lion',
                'fox', 'eagle', 'shark', 'dog', 'deer']})
df
```

4.3.7.7 Df. info()

Метод df.info() дає опис датафрейму: скільки у ньому рядків і стовпців, які типи даних містяться, скільки порожніх значень (non-null), скільки пам'яті займає.

```
DataFrame.info ( verbose = None , buf = None , max_cols = None , memory_usage = None , show_counts = None , null_counts = None )
```

```
df.info()
```

4.4 Контрольні питання

1. Яка бібліотека використовується в Python для роботи з даними?
2. Якого типу даны можна завантажувати в Pandas?
3. Які етапи представлення об’єкту Series як одномірного масиву? Наведіть приклад.
4. Що таке датафрейми та серії? Порівняйте. Наведіть приклади
5. Які етапи представлення об’єкту DataFrame? Наведіть приклад.
6. Які операції над даними можна виконати в бібліотеці Pandas? Наведіть приклад.
7. Що робити якщо відсутні дані в бібліотеці Pandas?
8. Що означає значення NaN та None? Чи реагує бібліотека Pandas на такі значення? Якщо так, то як саме?

ЛАБОРАТОРНА РОБОТА №5
ПРЕДСТАВЛЕННЯ, ОБРОБКА ДАНИХ НА ОСНОВІ БІБЛІОТЕКИ NUMPY. ПОБУДОВА МОДЕЛІ НЕЙРОННОЇ МЕРЕЖІ

5.1 Мета роботи.

Набути навичок аналізу даних на основі бібліотеки NumPy, вивчити архітектури формального нейрона роботи в середовище розробки Python тапровести регресійний аналіз даних.

5.2 Завдання до лабораторної роботи.

Частина 1. Відкрите програмне середовище Python та підключити бібліотеку numpy. Всі дані повинні зчитуватися з файлу у вигляді numpy масиву. Результати необхідно зберігати у файл.

Рівень 1 (від 60 до 73 балів)

1. Дано множину з p матриць (n,n) і множину з p векторів $(n,1)$. Написати функцію для знаходження суми p добутку матриць (результат має розмірність $(n,1)$). Індивідуальний варіант взяти у викладача.

Рівень 2 (від 74 до 89 балів)

1. Виконати завдання рівня 1.
2. Написати функцію, що перетворює вектор чисел на матрицю бінарних уявлень. (Матриця із завдання рівня1.)
3. Написати функцію, яка повертає всі унікальні рядки матриці. (Матриця із завдання рівня 1).

Рівень 3 (від 90 до 95 балів)

1. Виконати завдання рівня 1.
2. Виконати завдання рівня 2.
3. Написати функцію, яка заповнює матрицю з розмірами (M,N) випадковими числами, розподіленими за нормальним законом. Потім рахує мат. очікування та дисперсію для кожного зі стовпців, а також будує для кожного рядка варто гістограму значень (використовувати функцію hist з модуля matplotlib.plot).
4. Написати функцію, яка повертає тензор, що представляє зображення кола із заданим кольором і радіусом у схемі rgd на чорному тлі.

Рівень 4 (від 96 до 100 балів)

1. Виконати завдання рівня 1.
2. Виконати завдання рівня 2.
3. Виконати завдання рівня 3.
4. Написати функцію, що виділяє частину матриці фіксованого розміру з центром у цьому елементі.
5. Написати функцію, яка стандартизує всі значення тензор (відібрати мат. очікування та поділити на середньоквадратичне відхилення).

Частина 2. Використати модуль scikit-learn (sklearn) у Python та NumPy для реалізації операцій з даними.

Рівень 1 (від 60 до 73 балів)

1. Згенерувати синтетичні дані **forge.** Побудувати графік даних. Дати відповідь на питання: скільки ознак та точок містять синтетичні дані **forge?**

Рівень 2 (від 74 до 89 балів)

1. Виконати завдання рівня 1.

2. Згенерувати рандомно (*модуль random из numpy*) синтетичні дані на основі бібліотек Numpy та Scikit-learn. Кольору модель обрати RdYlBu. Представити розподіл 1D та побудувати графіки.

Рівень 3 (від 90 до 95 балів)

1. Виконати завдання рівня 1.
2. Виконати завдання рівня 2.
3. Згенерувати синтетичні дані для регресії (*пакет sklearn.datasets*). Побудувати графіки.

Рівень 4 (від 96 до 100 балів)

1. Виконати завдання рівня 1.
2. Виконати завдання рівня 2.
3. Виконати завдання рівня 3.
4. Згенерувати синтетичні дані для кластеризації (*функції make_blobs() та make_circles()).*Побудувати графіки для функцій *make_blobs() та make_circles().*

Частина 3. Побудова моделі нейронної мережі в середовищі Python.

Рівень 1 (від 60 до 89 балів)

1. Побудувати модель нейронної мережі на рис.5.1, з трьома рівнями. Перший – початковий рівень, має два входи - x1 і x2 . Один прихований рівень, на прихованому рівні два нейтрони – h1 і h2 . Один шар виходу, на ньому наділиться один нейрон – O1.

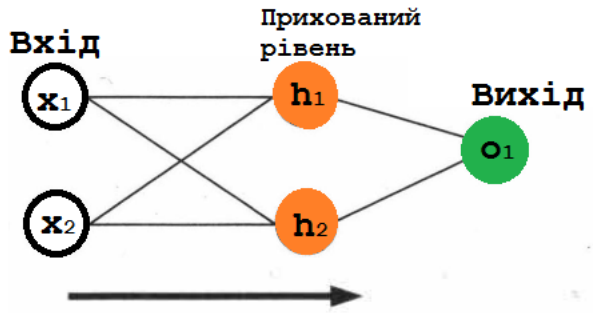


Рисунок 5.1 – Модель нейрона

2. Використати бібліотеку NumPy: функції однірного масиву; скалярного добудку.

3. Використати клас Neuron (нейрон).

Рівень 2 (від 90 до 100 балів)

1. Використовуючи Python для сформованих вибірок даних вирішити задачу побудови нейромереж Хопфілда та Елмана.

2. Зберегти початкові параметри нейромоделей, результати їхнього навчання (матрицю ваг та структуру з зазначенням функцій активації і кількості нейроелементів у шарах) та роботи для навчальної та контрольної вибірок (помилку класифікації (оцінювання), значення на входах і виході, час навчання, час класифікації).

3. Результати виконання пп. 1–2 занести у таблицю, стовпці якої повинні мати назви: назва архітектури нейромережі, кількість шарів, кількість нейронів у шарах, функції активації нейронів у шарах, метод навчання, час навчання, час класифікації навченої мережі для навчальної вибірки, помилка класифікації для навчальної вибірки, час класифікації навченої мережі для тестової вибірки, помилка класифікації для тестової вибірки.

4. Проаналізувати отримані результати і дати порівняльну характеристику мереж Хопфілда та Елмана.

5.3Теоретична частина.

5.3.1 Представлення даних

Усі сучасні системи машинного навчання використовують тензори як основну структуру даних. Тензори є фундаментальною структурою даних - настільки фундаментальною, що це вплинуло на назву бібліотеки Google TensorFlow. У Python для роботи з тензорами Найчастіше використовується бібліотека NumPy.

Фактично тензор - це контейнер для даних, практично завжди числових. Іншими словами, це контейнер для чисел. Наприклад, матриця є двовимірним тензором: тензори – це узагальнення матриць із довільною кількістю вимірювань.

5.3. 1.1 Скаляри (тензор 0 рангу)

Тензор, що містить однину, називається скаляром (скалярним, або тензором нульового рангу). Numpy число типу float32 або float64 - це скалярний тензор (або скалярний масив).

Визначити кількість осей тензора Numpy можна за допомогою атрибуту ndim; скалярний тензор має 0 осей (ndim == 0). Кількість осей тензора також називають його рангом. Приклад скаляра в NumPy:

```
import numpy as np
x = np.array(12)
print(x)
print(x.ndim)
print(type(x))
```

5.3.1.2 Вектори (тензор 1 рангу)

Одновимірний масив чисел називають вектором, або тензором першого рангу. Тензор першого рангу має єдину вісь. Приклад створення вектора:

```
import numpy as np
x = np.array([12, 3, 6, 14, 7])
print(x)
print(x.ndim)
print(type(x))
```

Цей вектор містить п'ять елементів, тому називається п'ятимірним вектором. Необхідно розрізняти розмірність вектора і тензора, тому що п'ятимірний вектор і п'ятимірний тензор є абсолютно різними структурами. П'ятимірний вектор має лише одну вісь і п'ять значень на цій осі, тоді як п'ятимірний тензор має п'ять осей (і може мати будь-яку кількість значень кожної з них). Мірність може позначати або кількість елементів на цій осі (як у випадку з п'ятивимірним вектором), або кількість осей у тензорі (як у п'ятивимірному тензорі), що іноді може спричинити плутанину.

5.3.1.3 Матриці (тензори 2 ранги)

Масив векторів – це матриця, або двомірний тензор. Матриця має дві осі (часто їх називають рядками та стовпцями). Матрицю можна як прямокутну таблицю з числами:

```
import numpy as np
x = np.array([[5, 78, 2, 34, 0],
              [6, 79, 3, 35, 1],
              [7, 80, 4, 36, 2]])
print(x)
print(x.ndim)
print(type(x))
```

5.3.1.4 Тензори третього та вищого рангів

Якщо запакувати такі матриці в новий масив, вийде тривимірний тензор, який можна подати як числовий куб. Приклад тривимірного тензора:

```
import numpy as np
x = np.array([[[5, 78, 2, 34, 0],
               [6, 79, 3, 35, 1],
               [7, 80, 4, 36, 2]],
               [[5, 78, 2, 34, 0],
               [6, 79, 3, 35, 1],
               [7, 80, 4, 36, 2]],
               [[5, 78, 2, 34, 0],
               [6, 79, 3, 35, 1],
               [7, 80, 4, 36, 2]]])
print(x)
print(x.ndim)
print(type(x))
```

Упакувавши тривимірні тензори в масив, ви отримаєте чотиривимірний тензор і т. д. У глибокому навчанні найчастіше використовуються тензори від нульового рангу до чотиривимірних, але іноді, наприклад, при обробці відеоданих, справа може дійти і до п'ятивимірних тензорів.

5.3.1.5 Ключові атрибути тензорів

Тензор визначається трьома ключовими атрибутами:

– кількість осей (ранг) – наприклад, тривимірний тензор має три осі, а матриця – дві. У бібліотеках Python, таких як Numpy, цей атрибут тензоров має ім'я ndim;

– форма - кортеж цілих чисел, що описують кількість вимірювань кожної осі тензора. Наприклад, матриця в попередньому прикладі має форму (3, 5), а тривимірний тензор має форму (3, 3, 5). Вектор має форму з єдиним елементом, наприклад (5,) тоді як скаляр має порожню форму (). У NumPy форма зберігається в атрибуті shape;

– тип даних (зазвичай у бібліотеках для Python йому дається ім'я dtype) - це тип даних, що містяться в тензорі; наприклад, тензор може мати тип float32, uint8, float64 та ін. У поодиноких випадках можна зустріти тензори типу char. Зверніть увагу, що в Numpy (і в більшості інших бібліотек) відсутні рядкові тензори, тому що тензори зберігаються в заздалегідь виділених, безперервних сегментах пам'яті та рядки, будучи сутностями із довжиною, що змінюється, перешкоджають використанню такої реалізації. У NumPy форма зберігається в атрибуті dtype.

5.3.1.6 Зчитування тензорів із файлу

NumPy представляє функціонал для зчитування даних із файлу. Припустимо, є файл із вмістом:

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]
<class 'numpy.ndarray'>
<class 'numpy.int32'>
1
```

У функції *np.fromfile* перший параметр відповідає за ім'я файлу, dtype визначає тип даних елемента тензора, sep відповідає за роздільник між

```
1;2;3;
4;5;6;
7;8;9;
10;11;12;
13;14;15;
```

Можна помітити, що дані вважалися вектор (тензор 1 рангу). Для того щоб отримати матрицю (тензор 2 рангу) таку як у файлі, необхідно змінити форму тензора, для цього використовується наступний скрипт:

```
y = np.reshape(x,[5,3])
print(y)
print(type(y))
print(type(y[0]))
print(y.ndim)
```

Функція *np.reshape* приймає тензор, який потрібно змінити і перелік розмірностей нового тензора. Після виконання скрипта буде отримано такий висновок:

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]
 [13 14 15]]
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
2
```

Змінювати розмірність кожного разу після зчитування не завжди зручно, якщо заздалегідь не відомі розміри тензора. Тому є зручніша функція для зчитування з файлу np.genfromtxt :

```
z = np.genfromtxt('data1.csv',dtype = 'int',delimiter=';')
print(z)
print(type(z))
print(z.ndim)
```

Після виконання скрипту буде наступний висновок:

```
[ 4  5  6 -1]
 [ 7  8  9 -1]
 [10 11 12 -1]
 [13 14 15 -1]]
<class 'numpy.ndarray'>
2
```

Можна замінити, що зчитування відбулося одразу у двомірний тензор, але з'явилася зайва колонка. Це з тим, що наприкінці кожного рядка

стоїть символ ';'. Тому необхідно змінити вміст файлу таким чином:

```
1;2;3
4;5;6
7;8;9
10;11;12
13;14;15
```

Цей формат точно відповідає формату CSV. Тепер виконаємо наступний код:

```
w = np.genfromtxt('data2.csv',dtype = 'int',delimiter=';')
print(w)
print(type(w))
print(w.ndim)
```

Отримуємо результат:

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]
 [13 14 15]]
<class 'numpy.ndarray'>
2
```

Як видно, дані відразу набувають необхідної форми. На жаль, у такого представлення даних у файлі є обмеження, що таким чином можна подати дані у вигляді тензора не більше 2 рангу.

Загалом перша вісь (вісь з індексом 0, тому що нумерація починається з 0) у всіх тензорах, з якими вам доведеться зіткнутися в глибокому навчанні, буде віссю зразків (іноді її називають виміром зразків).

Практичні приклади тензорів із даними

Перелічимо кілька прикладів тензорів із даними, які можуть зустрітися вам у майбутньому.

Дані, якими вам доведеться маніпулювати, майже завжди ставитимуться до однієї з наступних категорій:

– векторні дані – двомірні тензори з формою (зразки, ознаки). Форма даних, що найчастіше зустрічається. У таких наборах кожен зразок може бути представлений вектором, а пакет відповідно двовимірним тензором (тобто масивом векторів), де перша вісь - це вісь зразків, а друга - вісь ознак;

– тимчасові ряди або послідовності - тривимірні тензори з формою (зразки, мітки_часу, ознаки). Щоразу, коли час (або поняття послідовної впорядкованості) відіграє важливу роль у ваших даних, такі дані краще зберігати в тривимірному тензорі з явною віссю часу. Кожен зразок може бути представлений як послідовність векторів (двовимірних тензорів), а сам пакет даних як тривимірний тензор, а сам пакет даних як

тривимірний тензор. Відповідно до угодами, вісь часу є другою віссю (віссю з індексом 1);

– зображення - чотиривимірні тензори з формою (зразки, висота, ширина, колір) або з формою (зразки, колір, висота, ширина). Зазвичай зображення мають три виміри: висоту, ширину та колір. Навіть при тому, що чорно-білі зображення мають лише один канал кольору і могли б зберігатися у двовимірних тензорах, за угодами тензори із зображеннями завжди мають три виміри, де для чорно-білих зображень приділяється лише один канал кольору;

– відео – п'ятивимірні тензори з формою (зразки, кадри, висота, ширина, колір) або з формою (зразки, кадри, колір, висота, ширина). Відеодані — один із небагатьох типів даних, для зберігання яких потрібні п'ятивимірні тензори. Відео можна подати як послідовність кадрів, де кожен кадр – кольорове зображення. Кожен кадр можна зберегти у тривимірному тензорі (висота, ширина, колір), відповідно, їх послідовність можна зберегти у чотиривимірному тензорі (кадри, висота, ширина, колір), а пакет різних відеороликів – у п'ятивимірному тензорі з формою (зразки, кадри, висота , ширина, колір).

5.3.2. Функції для роботи з бібліотекою numpy

2.1 Функція arange()

Функція аналогічна функції range, але відразу повертає масив numpy.ndarray

```
import numpy as np

A = np.arange(10)

print(type(A)) #<class 'numpy.ndarray'>
print(A.shape) #(10,)
print(A) #[0 1 2 3 4 5 6 7 8 9]
```

5.3.2.2 Функція transpose()

Функція транспонування тензора. Для одновимірного випадку:

```
import numpy as np

A = np.arange(5)
print(A.shape) #(5,)
print(A) #[0 1 2 3 4]
print(A.transpose().shape) #(5,)
print(A.transpose()) #[0 1 2 3 4]
```

Так як розмірність одна, то транспонування немає. Якщо вектор представлений як матриця 5 на 1:

```
import numpy as np

A = np.arange(5).reshape([5,1])
print(A.shape) #(5, 1)
print(A) #[[0]
         #[1]
         #[2]
         #[3]
         #[4]]
print(A.transpose().shape) #(1, 5)
print(A.transpose()) #[[0 1 2 3 4]]
```

Для тензорів більшої розмірності:

```
import numpy as np

A = np.arange(120).reshape([2,3,4,5])
print(A.shape) #(2, 3, 4, 5)
print(A.transpose().shape) #(5, 4, 3, 2)
```

5.3.2.3 Функції ones (), zeros(), identity()

Функції дозволяють створити матрицю заповнену одиницями, нулями, а також одиничну відповідно.

```
import numpy as np

print(np.ones([3,4]))
'''
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
'''
print(np.zeros([4,3],dtype=float))
'''
[[0 0 0]
 [0 0 0]
 [0 0 0]
 [0 0 0]]
'''
print(np.identity(3))
'''
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
'''
```

5.3.3. Математичні операції

Для тензорів математичні операції працюють поелементно

```
import numpy as np

A = np.array([[1,2],[3,4]])
print(A)
'''
[[1 2]
 [3 4]]
'''
B = np.array([[5,6],[7,8]])
print(B)
'''
[[5 6]
 [7 8]]
'''

print(A+B)
'''
[[ 6  8]
 [10 12]]
'''
print(A*B)
'''
[[ 5 12]
 [21 32]]
'''
print(A/B)
'''
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
'''
```

Якщо розміри тензорів не збігаються, то менший тензор буде перетворено на використання. Найчастіше він буде повторений кілька разів.

```
import numpy as np

A = np.array([1,2])
print(A)
'''
[1 2]
'''
B = np.array([[5,6],[7,8]])
print(B)
'''
[[5 6]
 [7 8]]
'''

print(A+B)
'''
[[ 6  8]
 [ 8 10]]
'''
print(A*B)
'''
[[ 5 12]
 [ 7 16]]
'''
print(A/B)
'''
[[0.2        0.33333333]
 [0.14285714 0.25      ]]
'''
```

Звідси випливає, що можна виконувати математичні операції тензора зі скаляром. Також, у numpy є стандартні математичні функції, такі як log, asb, sin, exp, і т.д., які можна застосовувати до тензорів.

5.3.3.1 Функції dot () і tensordot()

NumPy забезпечує багато функцій для роботи з векторами та матрицями. Функція dot повертає вектор скалярний добуток.

```
import numpy as np

a = np.array([1,2,3])
b = np.array([3,2,1])
c = np.dot(a,b)
print(c) #10
```

Для перемноження матриць:

```
import numpy as np

a = np.array([[1,2],[2,1]])
b = np.array([[1,1],[0,1]])
c = np.dot(a,b)
print(c)
'''
[[1 3]
 [2 3]]
'''
```

Для функції dot виконуються такі умови:

– якщо a та b одномірні тензори - виконується скалярний добуток векторів;

– якщо a та b двовимірні тензори - виконується матричний твір, але рекомендується використовувати функцію matmul або оператор a@b;

– якщо a і b скаляри - виконується звичайне множення, що дорівнює функції multiply та оператору a * b;

– якщо a N-вимірний тензор і b 1-вимірний тензор (вектор) - виходить сума творів по останній осі a і b.

```
a = np.array([[1,2],[3,4]])
b = np.array([-1,2])
c = np.dot(a,b)
print(c)
'''
[3 5]
c[0] = 3 = 1 * -1 + 2 * 2 = a[0][0] * b[0] + a[0][1] + b[1]
c[1] = 5 = 3 + -1 + 2 * 4 = a[1][0] * b[0] + a[1][1] + b[1]
Совпадают последние индексы a и b
'''
```

Якщо a N-вимірний тензор і b M-вимірний тензор (M >= 2) - виходить сума творів по останній осі a і передостанній осі b.

```
dot(a, b)[i,j,k,m] = sum(a[i,j,:] * b[k,:,m])
```

```
import numpy as np

a = np.arange(24).reshape([2,3,4])
b = np.arange(100).reshape([5,4,5])
c = np.dot(a,b)
print(c.shape) #(2, 3, 5, 5)
```

Функція *tensordot* дозволяє вручну задати по яких осях виробляти суму творів:

```
import numpy as np

A = np.arange(5).reshape([1,5])
B = np.arange(5).reshape([5,1])

print(A)
'''
[[0 1 2 3 4]]
'''
print(B)
'''
[[0]
 [1]
 [2]
 [3]
 [4]]
'''


print(np.tensordot(A,B,axes=[0,1]))
'''
[[ 0  0  0  0  0]
 [ 0  1  2  3  4]
 [ 0  2  4  6  8]
 [ 0  3  6  9 12]
 [ 0  4  8 12 16]]
'''
print(np.dot(B,A)) #эквивалентная запись


print(np.tensordot(A,B,axes=[1,0]))
'''
[[30]]
'''
print(np.dot(A,B)) #эквивалентная запись
```

5.3.4 Моделювання нейронних мереж засобами бібліотек мови Python

Мова Python є однією з найбільш популярних мов програмування в останнє десятиріччя. Завдяки безкоштовним засобам розробки та потужним бібліотекам Python знаходить широке використання при розробці прикладного програмного забезпечення та вирішенні наукових задач. Зокрема велика кількість потужних бібліотек Python для моделювання нейронних мереж та розпізнавання образів дозволяє вирішувати задачі штучного інтелекту та інтелектуального аналізу даних. На рис.5.1 наведено класичну модель перцептрона.

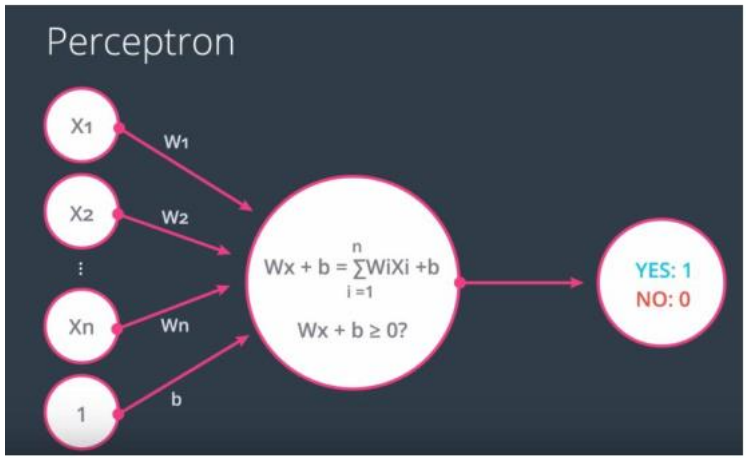


Рисунок 5.1 – Класичний перцептрон

Бібліотека NeuroLab (https://pythonhosted.org/neurolab/) – це бібліотека для Python, що містить моделі та методи навчання нейромереж з інтерфейсом.

Інсталяція бібліотеки здійснюється одним зі способів:

– використовуючи setuptools:

easy_install neurolab

– використовуючи pip:

pip install neurolab

– використовуючи python:

python setup.py install

Модуль net – модуль, що містить базові архітектури нейромереж в табл.5.1.

Таблиця 5.1 – Типи мереж

Тип мережі	Функція створення	Кількість шарів	Підтримка функцій навчання	Функція помилки
Одношаровий персептрон	newp	1	train_delta	SSE
Багатошаровий персептрон (БНМ)	newff	≥ 1	train_gd, train_gdm, train_gda, train_gdx, train_rprop, train_bfgs, train_cg	SSE
Конкуруючий шар (SOM)	newc	1	train_wta, train_cwta*	SAE
LVQ	newlvq	2	train_lvq	MSE
Мережа Елмана	newelm	≥ 1	train_gdx	MSE
Мережа Хопфілда	newhop	1	–	–
Мережа Хеммінга	newhem	2	–	–

neurolab.net.newp(minmax, cn, transf)– створення одношарового

персептрона. Параметри: minmax – список списків, які містять мінімальне та максимальне значення відповідної вхідної ознаки, cn – кількість вихідних нейронів, transf – функція активації. Повертає мережу net.

```
>>> # створення мережі з двома входами та 10 нейронами
>>> net = newp([[-1, 1], [-1, 1]], 10)
```

neurolab.net.newff(minmax, size, transf=None) – створення багатошарового персептрона (БНМ). Параметри: minmax – список списків, які містять мінімальне та максимальне значення відповідної вхідної ознаки, size – перелік, довжина якого відповідає кількості шарів, не враховуючи вхідний шар, елементи якого відповідають кількості нейронів відповідного шару, transf – функція активації. Повертає мережу net.

```
>>> # створення нейромережі з двома входами
>>> # діапазон для кожного входу: [-0.5, 0.5]
>>> # 3 нейрони у прихованому шарі, 1 вихідний нейрон
>>> # 2 шари, включаючи прихований та вихідний шари
>>> net = newff([[-0.5, 0.5], [-0.5, 0.5]], [3, 1])
>>> net.ci
2
>>> net.co
1
>>> len(net.layers)
2
```

neurolab.net.newhop(target, transf=None, max_init=10, delta=0) – створює рекурентну мережу **Хопфілда**. Параметри: target –масив цільових шаблонів, transf – функція активації, max_init – максимальна кількість рекурентних ітерацій, delta – мінімальна різниця між двома виходами для зупинення рекурентного циклу. Повертає нейромережу net.

```
>>> net = newhop([[-1, -1, -1], [1, -1, 1]])
>>> output = net.sim([[-1, 1, -1], [1, -1, 1]])
```

neurolab.net.newhem(target, transf=None, max_iter=10, delta=0) – створення рекурентної мережі **Хеммінга** з двома шарами. Параметри: target – масив з навчальними шаблонами, transf – функція активації вхідного шару, max_init – максимальна кількість рекурентних ітерацій, delta – мінімальна різниця між двома виходами для зупинення рекурентного циклу. Повертає мережу net.

```
>>> net = newhem([[-1, -1, -1], [1, -1, 1]])
>>> output = net.sim([[-1, 1, -1], [1, -1, 1]])
```

neurolab.net.newelm(minmax, size, transf=None) – створення

рекурентної мережі Елмана. Параметри: minmax – перелік мінімальних та максимальних значень для входів, size – список довжиною, що дорівнює кількостів шарів, без урахування вхідного шару, елементами якого є кількості нейронів відповідного шару. Повертає мережу net.

```
>>> # один вхід, діапазон входу [-1, 1],
>>> # один вихідний нейрон,
>>> # один шар, включаючи вихідний шар
>>> net = newelm([[-1, 1]], [1], [trans.PureLin()])
>>> # встановлення ваг для усіх вхідних нейронів в 1
>>> net.layers[0].np['w'][:] = 1
>>> # встановлення порогів усіх вхідних нейронів у 0
>>> net.layers[0].np['b'][:] = 0
>>> net.sim([[1], [1] ,[1], [3]])
array([[ 1.],
[ 2.],
[ 3.],
[ 6.]])
```

neurolab.net.newc(minmax, cn) – створення конкурентного шару (мережі **Кохонена**). Параметри: minmax – перелік мінімальних та максимальних значень для входів, cn – кількість вихідних нейронів. Повертає мережу net.

neurolab.net.newlvq(minmax, cn0, pc) – створює мережу LVQ. Параметри: minmax – список списків, які містять мінімальне та максимальне значення відповідної вхідної ознаки, cn0 – кількість нейронів у вхідному шарі, pc – список процентів, сума яких має дорівнювати одиниці. Повертає мережу net.

init – функції ініціалізації шарів мережі.

neurolab.init.InitRand(minmax, init_prop) – ініціалізація заданих властивостей шару випадковими числами у заданих обмеженнях.

neurolab.init.init_rand(layer, min=-0.5, max=0.5, init_prop='w') – ініціалізація заданої властивості шару випадковими числами у заданих обмеженнях.

neurolab.init.init_zeros(layer) – встановлення усіх властивостей шару layer у нуль.

neurolab.init.initnw(layer) – ініціалізація ваг шару layer методом Нгуена-Уідроу.

neurolab.init.initwb_lin(layer) – ініціалізація лінійного простору ваг та порогів layer невипадковими значеннями.

neurolab.init.initwb_reg(layer) – ініціалізація ваг та порогів шару layer у діапазоні, заданому активаційною функцією.

neurolab.init.midpoint(layer) – встановлення вагових коефіцієнтів шару layer у центрі діапазонів входів.

train – методи навчання нейромереж. Типові параметри: input – значення вхідних ознак розпізнаваних екземплярів, target – цільові значення вихідних ознак для розпізнаваних екземплярів, epochs – максимально припустима кількість циклів навчання, show – період відображення поточних результатів навчання, goal – значення цільової функції, при досягненні якого навчання зупиняється, lr – крок навчання.

5.3.4.1 Навчання одношарового персептрона.

neurolab.train.train_delta() – дельта-правило. Градієнтні методи навчання БНМ. Типові параметри градієнтних методів (у доповнення до розглянутих вище параметрів train): adapt – тип навчання, rr – коефіцієнт регуляризації навчання, mc – константа моменту навчання, lr_inc – коефіцієнт збільшення швидкості навчання, lr_dec – коефіцієнт зменшення швидкості навчання, max_perf_inc –

максимально припустиме збільшення цільової функції, rate_dec – зменшення зміни ваг, rate_inc – приріст зміни ваг, rate_min – мінімальне значення градієнта цільової функції, rate_max – максимальне значення зміни вагових коефіцієнтів.

neurolab.train.train_gd() – градієнтний спуск зі зворотним поширенням помилки.

neurolab.train.train_gdm() – градієнтний спуск з моментом зі зворотним поширенням помилки.

neurolab.train.train_gda() – градієнтний спуск з адаптивним навчальним кроком.

neurolab.train.train_gdx()– градієнтний спуск з моментом зі зворотним поширенням помилки та адаптивним навчальним кроком.

neurolab.train.train_rprop() – еластичне зворотне поширення помилки.

neurolab.train.train_bfgs() – метод Бройдена-Флетчера–Гольдфарба-Шанно.

neurolab.train.train_cg() – метод спряжених градієнтів.

neurolab.train.train_ncg() – метод спряжених градієнтів Ньютона.

5.3.4.2 Методи навчання на основі правила "переможець отримує усе" (Winner Take All).

neurolab.train.train_wta() – метод "переможець отримує усе" для мережі SOM. Параметри: input – значення вхідних ознак екземплярів вибірки, epochs – максимально припустима кількість епох, show – період друку у процесі навчання, goal –значення цільової функції, при якому навчання зупиняється.

neurolab.train.train_cwta() – совісливий метод "переможець отримує усе" для мережі SOM. Параметри: input – значення вхідних ознак екземплярів вибірки, epochs – максимально припустима кількість епох, show – період друку у процесі навчання, goal –значення цільової функції,

при якому навчання зупиняється.

5.4 Контрольні питання.

1. Яка бібліотека використовується в Python для роботи з тензером?
2. Що таке фактичний тензор?
3. Що позначає число типу float32 або float64 в бібліотеці Numpy?
4. Наведіть приклад скаляра в NumPy .
5. Що таке тензор першого рангу? Наведіть приклад.
6. Які ви знаєте програмні засоби для моделювання НМ?
7. У якому модулі Python містяться засоби для моделювання нейромереж?
8. Наведіть приклад математичної моделі штучного нейрону.
9. Які задачі дозволяє розв’язати штучна нейронна мережа?
10. Яка існує класифікація нейронних мереж і їх властивості?
11. Опишіть процес моделювання і навчання НМ Хопфілда у Python.
12. Яка особливість принципу «Winner Takes All» («Переможець отримує все»)?

ЛАБОРАТОРНА РОБОТА №6
СТВОРЕННЯ ТА НАВЧАННЯ ПРОСТОЇ НЕЙРОННОЇ МЕРЕЖІ В KERAS

6.1 Мета роботи.

Навчитися користуватися бібліотекою Keras та її вбудованими об'єктами; виконувати класифікацію чорно-білих зображень; працювати з вбудованим набором даних Predict Sentiment From Movie Reviews в Keras.

6.2 Завдання до лабораторної роботи.

Частина 1. Необхідно побудувати нейронну мережу, яка класифікуватиме чорно-білі зображення простих геометричних фігур.

Рівень 1 (від 60 до 73 балів)

1. Створіть власну нейрону мережу за варіантом (таб.6.1), де № варіанту відповідає порядковому номеру вашої ПІБ у списку групи вибіркової дисципліни. **До кожного варіанту додається код, що генерує зображення та наведено в додатку Б.**

Таблиця 6.1 – Індивідуальне завдання до частини 1

№	Завдання
1	Класифікація зображень у вигляді прямокутника.
2	Класифікація зображень у вигляді кола.
3	Класифікація зображень у вигляді кола із заливкою (кольором).
4	Класифікація зображень у вигляді кола без заливки (кольору).
5	Класифікація зображень у вигляді хреста.
6	Класифікація зображень у вигляді перетину ліній.
7	Класифікація зображень у вигляді квадрату.
8	Класифікація зображень у вигляді еліпсу без заливки (кольору).
9	Класифікація зображень по кількості хрестів на ньому (якщо 1).
10	Класифікація зображень по кількості хрестів на ньому (якщо 2).
11	Класифікація зображень по кількості хрестів на ньому (якщо 3).
12	Класифікація зображень по кількості ліній на ньому (якщо 1).
13	Класифікація зображень по кількості ліній на ньому (якщо 2).
14	Класифікація зображень по кількості ліній на ньому (якщо 3).
15	Класифікація зображень по кількості перетину ліній на ньому (якщо 4).

2. Для генерації даних необхідно викликати функцію **gen_data**, яка

повертає два тензори:

2.1 Тензор із зображеннями рангу 3.

2.2 Тензор із мітками класів.

Зверніть увагу:Вибірки не перемішані, тобто спостереження класів йдуть по порядку. Класи характеризуються рядковою міткою.Вибірка спочатку не розбита на навчальну, контрольну та тестову.

3. Завантажувати необхідно обидва файли. Підключати файл, який починається з var (у ньому і знаходиться функція gen_data)

Рівень 2 (від 74 до 90 балів)

Частина 2. Бінарна класифікація вбудованими засобами Keras.

1. Завантажити дата сет Keras.datasets.imdb.load_data ().

2. Провести векторизацію кожного огляду за варіантом (таб.6.2), де № варіанту відповідає порядковому номеру вашої ПІБ у списку групи вибіркової дисципліни.

Таблиця 6.2 – Індивідуальне завдання до частини 2

№	Завдання
1	Аргумент num_words=10000
2	Аргумент num_words=2000
3	Аргумент num_words=3000
4	Аргумент num_words=8000
5	Аргумент num_words=7000
6	Аргумент num_words=4000
7	Аргумент num_words=6000
8	Аргумент num_words=5000
9	Аргумент num_words=1000
10	Аргумент num_words=9000
11	Аргумент num_words=11000
12	Аргумент num_words=12000
13	Аргумент num_words=13000
14	Аргумент num_words=100
15	Аргумент num_words=200

3. Розділитио датасет на навчальний та тестувальний набори. Навчальний набір складатиметься з 40 000 оглядів, тестувальний - з кількості в табл. 2.

4. Створення та навчання моделі в Keras за варіантом (таб.6.3), де № варіанту відповідає порядковому номеру вашої ПІБ у списку групи вибіркової дисципліни.

5. До моделі додати вхідні, приховані та вихідні рівні.

6. Провести оцінку роботи моделі.

Таблиця 6.3 – Індивідуальне завдання до частини 2

№	Тип моделі
1	послідовна
2	з функціональним API.
3	послідовна
4	з функціональним API.
5	послідовна
6	з функціональним API.
7	послідовна
8	з функціональним API.
9	послідовна
10	з функціональним API.
11	послідовна
12	з функціональним API.
13	послідовна
14	з функціональним API.
15	послідовна

7. Написати функцію, яка дозволяє ввести текст користувача (у звіті навести приклад роботи мережі на тексті користувача).

8. Додатково можна навести графіки.

Частина 3. Додаткова для отримання оцінки 91-100 балів. Групова робота.

Для виконання даної частини пропонується об’єднатись в групи по 3 студенти та виконати завдання. Важливим є розподіли обов’язки групи, виконати свою частину та презентувати міні проект.

1. Необхідно в залежності від варіанта групи згенерувати датасет і зберегти його у форматі CSV.

2. Побудувати модель, яка міститиме автокодувальник і регресійну модель з рис.6.1.

3. Навчити модель і розбити навчену модель на 3 частини: Модель кодування даних (Вхідні дані -> Запрограмовані дані), модель декодування даних (Запрограмовані дані -> Декодовані дані), та регресійну модель (Первинні дані -> Результат регресії).

4. Представити вихідний код, згенеровані дані у форматі csv, кодовані та декодовані дані у форматі csv, результат регресії у форматі csv (що має бути і що видає модель), і самі 3 моделі у форматі h5.

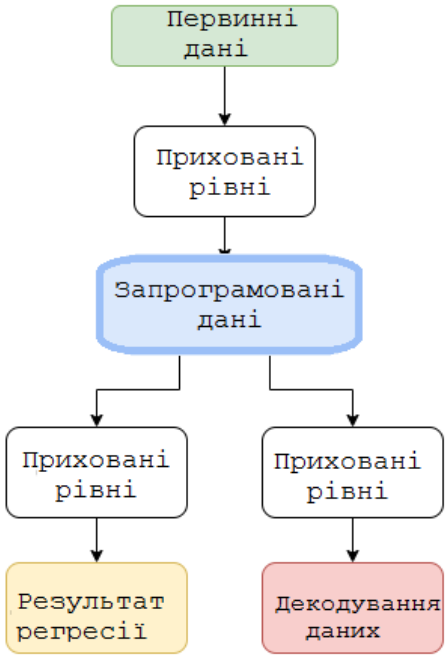


Рисунок 6.1 – Модель до частини 3

5. Зберегти та представити моделі можна як:

```
from keras.models import load_model

model.save('my_model.h5')  # creates a HDF5 file 'my_model.h5'
del model  # deletes the existing model

# returns a compiled model
# identical to the previous one
model = load_model('my_model.h5')
```

6. Варіант групи назначає викладач, мета регресії вибирається по залишку розподілу номера заліковки (останні 2 цифри) на 7 плюс 1. Передбачити треба одне значення, виходячи з шести інших. У назві pr необхідно буде вказати номер групи та номер мети.

Група 1.

X ∈ N(3,10)
e ∈ N(0,0.3)

Ознака	1	2	3	4	5	6	7
Формула	X^2 +e	sin(X/2)+ e	cos(2x) +e	X-3+e	-X+e	|X|+e	(X^3)/4+

Група 2.

X ∈ N(-5,10)
e ∈ N(0,0.3)

Ознака	1	2	3	4	5	6	7
Формула	-X^3+e	ln(|X|)+e	sin(3X)+e	exp(X)+e	X+4+e	-X+sqrt(|X|)+e	X+e

Група 3.

X ∈ N(0,10)
e ∈ N(0,0.3)

Ознака	1	2	3	4	5	6	7
Формула	X^2+X+e	|X|+e	sin(X-PI\4)+e	lg(|X|)+e	-X^3+e	-X/4+e	-X+e

Група 4.

X ∈ N(0,10)
e ∈ N(0,0.3)

Ознака	1	2	3	4	5	6	7
Формула	cos(X)+e	-X+e	sin(X)*X+e	sqrt(|X|)+e	X^2+e	-|X|+4	X-(X^2)/5+e

6.3 Теоретична частина.

6.3.1. Бібліотека Keras

Бібліотека Keras (https://keras.io/) є потужним засобом для моделювання нейронних мереж мовою Python.

За замовчуванням Keras (рис.6.1) використовує бібліотеку Tensorflow для маніпуляції тензорами. Тому перед інсталяцією Keras потрібно встановити бібліотеку Tensorflow.

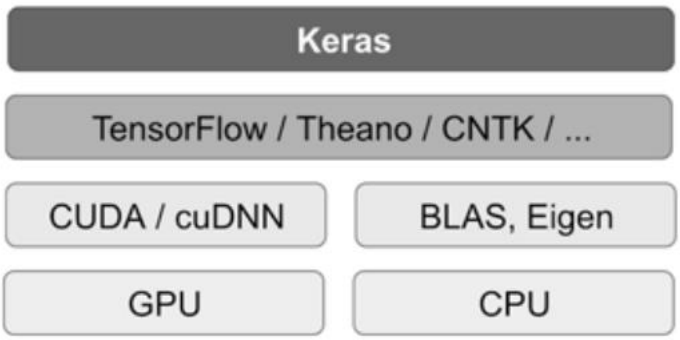


Рисунок 6.1 - Програмно-апаратний стек підтримки глибокого навчання

Інсталяція бібліотеки Keras здійснюється одним із двох способів:

– з PyPI (рекомендується):

```
sudo pip install keras
pip install keras
```

– з джерела GitHub:

```
git clone https://github.com/keras-team/keras.git
cd keras
sudo python setup.py install
```

Етапи використання Keras:

1. Визначити навчальні дані: вхідні та цільові тензори.

2. Визначити слова мережі (модель), що відображають вхідні дані в цільові.

3. Налаштувати процес навчання вибором функції втрат, оптимізатора та деяких параметрів моніторингу.

4. Виконати ітерації за навчальним даним викликом методу fit() моделі.

Модель можна визначити двома способами: з використанням функції keras_model_sequential() (тільки для лінійного стека шарів - найпопулярніша архітектура мереж в даний час) або функціонального API (для орієнтованого ациклічного графа рівнів, що дозволяє конструювати довільні архітектури).

6.3.2. Моделі глибокого навчання у Keras

В основі Keras лежать моделі, основний тип яких - послідовність, що є лінійним стек шарів.

Суть у наступному: створюється послідовність і до неї додаються шари в тому порядку, в якому потрібно виконати обчислення. Після визначення компілюється модель, що використовує базову платформу оптимізації обчислень.

Далі модель повинна відповідати даним - це можна зробити по одній партії даних за один раз або запустивши весь режим навчання моделі. Коли навчання буде завершено, модель можна використовувати для прогнозування нових даних.

Також є ще один тип моделей – це клас Model, який використовується з функціональним API.

6.3.3. Методи бібліотеки Keras

Метод compile(optimizer, loss=None, metrics=None, loss_weights=None, sample_weight_mode=None, weighted_metrics =None, target_tensors=None) конфігурує модель для навчання. Як аргументи використовує: optimizer – рядок з назвою оптимізатора, loss – рядок з ім’ям функції втрат, metrics – список метрик, що обраховуються під час навчання та тестування моделі, loss_weights – необов’язковий перелік скалярних коефіцієнтів втрат, sample_weight_mode –режим визначення вагових коефіцієнтів, weighted_metrics – перелік метрик, що обраховуються та зважуються вагами вибірки або вагами класу під час навчання та тестування, target_tensors – визначення керованих параметрів навчання.

Метод fit(x=None, y=None, batch_size=None, epochs=1, verbose=1, callbacks=None, validation_split=0.0, validation_data=None, shuffle=True, class_weight=None, sample_weight=None, initial_epoch=0, steps_per_epoch= None, validation_steps=None, validation_freq=1, max_queue_size=10, workers=1, use_multiprocessing=False) навчає модель для заданої кількості циклів навчання (епох – проходів вибірки). Як аргументи використовує: x – вхідні дані, y – цільові дані, batch_size – розмір пакету (кількість екземплярів на оновлення градієнта), epochs – кількість циклів навчання моделі на усій вибірці даних, verbose – режим

відображення ходу навчання, callbacks – список викликів, validation_split – для навчальних даних, що використовуються для перевірки моделі, validation_data – дані для обрахунку функції втрат та метрик наприкінці кожної епохи навчання, shuffle – регулювання перемішування навчальних даних перед кожною епохою навчання, class_weight – ваги класів при розрахунку функції втрат, sample_weight – ваги екземплярів, враховувані при визначенні функції втрат, initial_epoch – епоха, з якої починається навчання, steps_per_epoch – кількість пакетів (підвибірок) екземплярів, що обробляються на одній епосі, validation_steps – кількість пакетів (підвибірок) екземплярів для перевірки моделі перед зупиненням, validation_freq – кількість епох навчання, що виконуються до запуску перевірки, max_queue_size – максимальний розмір черги генератора, workers – максимальна кількість процесів, що можуть прискорюватися при використання потоків на основі процесів, use_multiprocessing – прапорець використання потоків. Повертає об'єкт History, властивість якого History.history – це запис значень втрат навчання та метрик навчання на успішних епохах, а також значень втрат та метрик тестування моделі.

Метод predict(x, batch_size=None, verbose=0, steps= None, callbacks=None, max_queue_size=10, workers=1, use_multiprocessing=False) генерує передбачення (оцінку) виходу для вхідних екземплярів. Як аргументи використовує: x – вхідні дані, що розпізнаються, batch_size – розмір пакету на оновлення градієнта, verbose – прапорець пояснень, steps – кількість кроків (пакетів екземплярів) перед завершенням циклу оцінювання, callbacks – список зворотних викликів, max_queue_size – максимальний розмір черги генератора, workers – максимальна кількість процесів, use_multiprocessing – прапорець використання багатопроцесорних потоків. Повертає масив(и) Numpy, що містять передбачення (оцінки) виходу.

6.3.3.1 Приклад

Розглянемо використання засобів бібліотеки на прикладах фрагментів тексту програм (джерело: https://www.datacamp.com/). Базовий приклад:

```
# підключення бібліотек та імпорт їхніх об'єктів
 >>> import numpy as np
>>> from keras.models import Sequential
>>> from keras.layers import Dense # генерація випадкового масиву даних
>>> data = np.random.random((1000,100))
>>> labels = np.random.randint(2,size=(1000,1)) # створення структури моделі нейромережі
>>> model = Sequential()
>>> model.add(Dense(32, activation='relu', input_dim=100)) >>> model.add(Dense(1, activation='sigmoid')) # компіляція
>>> model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy']) # навчання моделі
>>> model.fit(data,labels,epochs=10,batch_size=32) # запуск емуляції моделі для набору даних
>>> predictions = model.predict(data)
```

Дані мають зберігатися як масиви NumPy або як списки масивів NumPy. Бажано первинну вибірку даних розбити на навчальну та тестові вибірки.

6.3.3.2 Завантаження даних з наборів Keras.

Представлено на прикладі:

```
>>> from keras.datasets import boston_housing, mnist, cifar10, imdb
>>> (x_train,y_train),(x_test,y_test) = mnist.load_data()
>>> (x_train2,y_train2),(x_test2,y_test2) = boston_housing.load_data()
>>> (x_train3,y_train3),(x_test3,y_test3) = cifar10.load_data()
>>> (x_train4,y_train4),(x_test4,y_test4)
= imdb.load_data(num_words=20000)
>>> num_classes = 10
```

6.3.4 Попередня обробка даних.

```
# Заповнення послідовності
>>> from keras.preprocessing import sequence
>>> x_train4 = sequence.pad_sequences(x_train4,maxlen=80)
>>> x_test4 = sequence.pad_sequences(x_test4,maxlen=80)
# Кодування даних
>>> from keras.utils import to_categorical
>>> Y_train = to_categorical(y_train, num_classes)
>>> Y_test = to_categorical(y_test, num_classes)
>>> Y_train3 = to_categorical(y_train3, num_classes)
>>> Y_test3 = to_categorical(y_test3, num_classes)
# Формування навчальної та тествої вибірок даних
>>> from sklearn.model_selection import train_test_split
>>> X_train5, X_test5, y_train5, y_test5 =
train_test_split(X, y, test_size=0.33, random_state=42)
# Нормалізація даних
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(x_train2)
>>> standardized_X = scaler.transform(x_train2)
>>> standardized_X_test = scaler.transform(x_test2)
```

6.3.5 Визначення архітектури моделі нейромережі.

```
# Послідовна модель
>>> from keras.models import Sequential
>>> model = Sequential()
>>> model2 = Sequential()
>>> model3 = Sequential()
– багатошаровий персептрон:
# бінарна класифікація:
>>> from keras.layers import Dense
>>> model.add(Dense(12, input_dim=8, kernel_initializer=
'uniform', activation='relu'))
>>> model.add(Dense(8, kernel_initializer='uniform',
```

```
activation='relu'))
>>> model.add(Dense(1, kernel_initializer='uniform',
activation='sigmoid'))
# багатокласова класифікація:
>>> from keras.layers import Dropout
>>> model.add(Dense(512,activation='relu',input_shape=(784,
)))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(512,activation='relu'))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(10,activation='softmax'))
# оцінювання:
>>> model.add(Dense(64, activation='relu',
input_dim=train_data.shape[1]))
>>> model.add(Dense(1))
```

– згорткова нейромережа:

```
>>> from keras.layers import Activation, Conv2D,
MaxPooling2D, Flatten
>>> model2.add(Conv2D(32, (3,3), padding='same',
input_shape=x_train.shape[1:]))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(32, (3,3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))
>>> model2.add(Conv2D(64, (3,3), padding='same'))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(64, (3, 3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))
>>> model2.add(Flatten())
>>> model2.add(Dense(512))
>>> model2.add(Activation('relu'))
>>> model2.add(Dropout(0.5))
>>> model2.add(Dense(num_classes))
>>> model2.add(Activation('softmax'))
```

– рекурентна нейромережа:

```
>>> from keras.klayers import Embedding,LSTM
>>> model3.add(Embedding(20000,128))
>>> model3.add(LSTM(128,dropout=0.2,
recurrent_dropout=0.2))
>>> model3.add(Dense(1,activation='sigmoid'))
```

6.3.6 Огляд моделі.

```
# форма виходу моделі
>>> model.output_shape
# зведене подання моделі
>>> model.summary()
# конфігурація моделі
>>> model.get_config()
# перелік усіх тензорів ваг моделі
>>> model.get_weights()
```

6.3.7 Компіляція моделей

```
– багатошаровий персептрон:
# Бінарна класифікація
>>> model.compile(optimizer='adam',
loss='binary_crossentropy', metrics=['accuracy'])
# Багатокласова класифікація
>>> model.compile(optimizer='rmsprop',
loss='categorical_crossentropy', metrics=['accuracy'])
# Оцінювання
>>> model.compile(optimizer='rmsprop', loss='mse',
metrics=['mae'])
– рекурентна нейромережа:
# Визначення параметрів моделі
>>> model3.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])
```

6.3.8 Навчання моделі.

```
>>> model3.fit(x_train4, y_train4, batch_size=32,
epochs=15, verbose=1, validation_data=(x_test4,
y_test4))
```

6.3.9 Обчислення якості моделі.

```
>>> score = model3.evaluate(x_test, y_test,
batch_size=32)
```

6.3.10 Емуляція (запуск) моделі.

```
>>> model3.predict(x_test4, batch_size=32)
>>> model3.predict_classes(x_test4,batch_size=32)
```

6.3.11 Збереження та завантаження моделі.

```
>>> from keras.models import load_model
>>> model3.save('model_file.h5')
>>> my_model = load_model('my_model.h5')
```

6.3.12 Оптимізація параметрів моделі.

```
>>> from keras.optimizers import RMSprop
>>> opt = RMSprop(lr=0.0001, decay=1e-6)
>>> model2.compile(loss='categorical_crossentropy',
optimizer=opt, metrics=['accuracy'])
```

Раннє зупинення навчання моделі.

```
>>> from keras.callbacks import EarlyStopping
>>> early_stopping_monitor = EarlyStopping(patience=2)
>>> model3.fit(x_train4, y_train4, batch_size=32,
epochs=15, validation_data=(x_test4, y_test4),
callbacks=[early_stopping_monitor])
```

6.3.13 Бінарна класифікація

Класифікація за двома класами, або бінарна класифікація, чи не сама найпоширенішим завданням машинного навчання. Розглянемо приклад бінарної класифікації на задачі класифікації відгуків до фільмів на позитивні та негативні, спираючись на текст відгуків.

Для вирішення завдання буде використовувати набір даних IMDB: безліччю з 50 000 різних відгуків до кінострічок в інтернет-базі фільмів (Internet Movie Database). Набір розбито на 25 000 навчальних та 25 000 контрольних відгуків, кожен набір на 50 % складається з негативних та на 50 % із позитивних відгуків. Цей набір постачається разом із бібліотекою Keras.

Для початку, підключимо цей набір даних, а потім створимо та ініціалізуємо змінні train_data та test_data (списки відгуків; кожен відгук — це список індексів слів) та змінні train_labels і test_labels (списки нулів та одиниць, де нулі відповідають негативним відгукам, а одиниці - позитивним):

```
from keras.datasets import imdb (train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=10000)
```

Тому що не можна передати списки цілих чисел безпосередньо в нейронну мережу. Тому, необхідно перетворити в тензоры. Для цього це зробимо так:виконаємо пряме кодування списків у вектори нулів та одиниць. Наприклад, перетворення послідовності [3, 5] - 10 000-мірний вектор, всі елементи якого містять нулі, крім елементів з індексами 3 та 5, які містять одиниці. Таким чином, враховуватиметься тільки наявність слів у тексті, але не те, скільки разів вони зустрічаються, також не враховуватиметься порядок слів.

```
import numpy as np

def vectorize_sequences(sequences, dimension=10000):
results = np.zeros((len(sequences), dimension))
for i, sequence in enumerate(sequences):
results[i, sequence] = 1.
```

```
return results
x_train = vectorize_sequences(train_data) x_test = vectorize_sequences(test_data)

y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

Вхідні дані представлені векторами, а мітки - скалярами, це найпростіший набір даних, який можна зустріти. Із завданнями цього виду чудово справляються мережі, організовані як простий стек пов'язаних (Dense) шарів з операцією активації ***relu*** (рис.6.2).

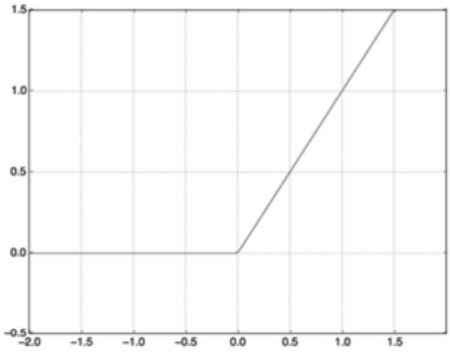


Рисунок 6.2 – Функція активації relu

При побудові такої мережі (ще називається багатошаровий перцептрон, багатошарова мережа прямого поширення) необхідно приймати такі рішення:

– скільки шарів використовувати;
– скільки прихованих нейронів вибрати для кожного шару.

Для цього завдання виберемо таку архітектуру:

1. 2 повнозв'язних прихованих шари з 16 нейронами
2. 1 вихідний шар з 1 нейроном, тому що в результаті необхідно отримати скаляр
3. На повнозв'язкових шарах використовуватиме ф-ція активації relu (рис. 1).
4. На вихідному шарі використовуватиметься сигмоїдна функція (рис. 6.3), яка має область значень [01]. Дані значення можна інтерпретувати як ймовірність того, що відгук позитивний.

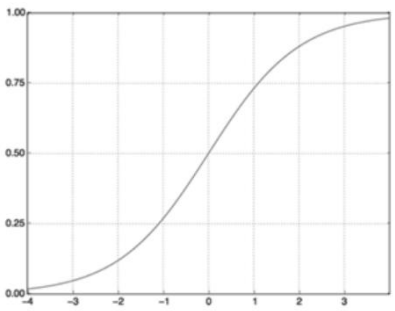


Рисунок 6.3 – Сигмоїдна функція

Побудова побелі в Python:

```
from keras import models
from keras import layers

model = models.Sequential()

model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

При створенні шару Dense також можна вказувати наявність нейронного зсуву, спосіб ініціалізації ваг, обмеження на ваги в шарі, регуляризацію ваг.

Після того, як модель створена, необхідно вибрати оптимізатор (тобто яким методом буде проводиться оптимізація), функція втрат (як відбувається розрахунок помилки мережі) та метрики (значення, які розраховуватимуться під час навчання). Для налаштування навчання моделі, використовується функція compile , для якої як аргументи передаються вищеописані характеристики навчання:

```
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
```

У нашому завдання як оптимізатор буде використовуватися RMSProp, функцією втрат бінарна крос-ентропія (функція, яка в основному використовується при бінарній класифікації), а як метрика використовується точність.

Щоб проконтролювати точність моделі під час навчання на даних, які вона раніше не бачила, створимо перевірочний набір, обравши 10 000 зразків із оригінального набору навчальних даних:

```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

Тепер проведемо навчання моделі протягом 20 епох (виконавши 20 ітерацій за всіма зразками в тензорах *x_train ma y_train)* пакетами по 512 зразків. Водночас стежитимемо за втратами та точністю на 10000 відкладених зразків. Для цього достатньо передати перевірочні дані в аргумент *validation_data:*

```
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

Виклик model.fit() повертає об'єкт History. Цей об'єкт має поле history - словник з даними про все, що відбувалося в процесі навчання. Заглянемо до нього:

```
history_dict = history.history
print(history_dict.keys())
#dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```

Далі, виведемо графіки помилки та точності під час навчання використовуючи бібліотеку MatPlotLib

```
import matplotlib.pyplot as plt #импорт модуля для графіків

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Графік помилки виведено на рис.6.4 (як результат коду):

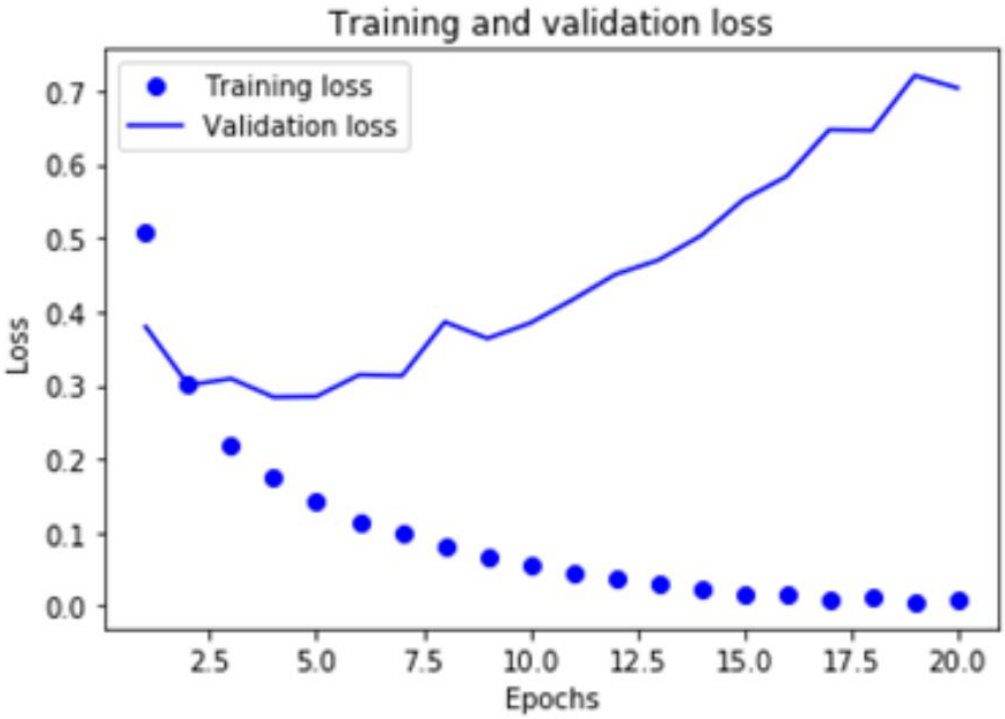


Рисунок 6.4 – Графік помилок

Як бачите, на етапі навчання втрати знижуються з кожною добою, а точність зростає. Саме така поведінка очікується від оптимізації градієнтним спуском: величина, яку ви намагаєтесь мінімізувати, повинна все менше з кожною ітерацією. Але це не стосується втрат і точності на етапі перевірки: схоже, що вони досягли піку у четверту епоху. Це приклад перенавчання.

Для оцінки моделі після навчання на тестових даних використовується функція evaluate:

```
model.evaluate(x_test, y_test)
#25000/25000 [==============================] - 2s 79us/step
#[0.7763081940078735, 0.84952]
```

6.3.14. Класифікація кількох класів

Дане завдання розглянемо на наборі даних Reuters - вибіркою стрічок новин і їх тем, що публікувалися агентством Reuters в 1986 році. Це простий набір даних, які широко використовуються для класифікації тексту. Існує 46 різних тем; деякі теми ширше представлені, деякі — менш, але кожної з них у навчальному наборі є щонайменше 10 прикладів.

Конструювання мережі для цієї задачі здебільшого схоже на конструювання мережі для бінарної класифікації.

Подібно до IMDB, набір даних Reuters поставляється у складі Keras. Тому завантаження, а також підготовка даних аналогічна як і в минулому:

```
from keras.datasets import reuters

(train_data, train_labels), (test_data, test_labels) =
reuters.load_data(num_words=10000)

import numpy as np
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
    return results

x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

Так як в результаті необхідно отримувати інформацію про приналежність до одного з класів, необхідно підготувати відповідно мітки. Для цього. Векторизувати мітки можна одним із двох способів: зберегти їх у тензорі цілих чисел або використовувати пряме кодування. Пряме кодування (one-hot encoding) широко використовується для форматування категорій і називається кодуванням категорій (categorical encoding). У даному випадку пряме кодування міток полягає в конструюванні вектора з нульовими елементами зі значенням 1 елементі, індекс якого відповідає індексу мітки:

```
def to_one_hot(labels, dimension=46):
    results = np.zeros((len(labels), dimension))
    for i, label in enumerate(labels):
        results[i, label] = 1.
    return results

one_hot_train_labels = to_one_hot(train_labels)
one_hot_test_labels = to_one_hot(test_labels)
```

Даний метод вже реалізований у бібліотеці Keras:

```
from keras.utils.np_utils import to_categorical
one_hot_train_labels = to_categorical(train_labels)
one_hot_test_labels = to_categorical(test_labels)
```

Завдання класифікації на теми нагадує попереднє завдання класифікації відгуків: в обох

У випадках ми намагаємося класифікувати короткі фрагменти тексту. Але в даному випадку кількість вихідних класів збільшилася з 2 до 46. Розмірність вихідного простору тепер набагато більша. У стеку шарів Dense, як і в попередньому прикладі, кожен шар має доступ тільки до інформації, наданої попереднім шаром. Якщо один шар відкине якусь

інформацію, важливу вирішення завдання класифікації, наступні шари не зможуть відновити її: кожен шар може бути вузьким місцем інформації. У попередньому прикладі ми використовували 16-мірні проміжні шари, але 16-мірний простір може виявитися занадто обмеженим для класифікації на 46 різних класів: такі малорозмірні шари можуть зіграти роль «пляшкового шийки» для інформації, не пропускаючи важливих даних.

Обиремо 64 нейрона:

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(46, activation='softmax'))
```

Зазначимо ще дві особливості цієї архітектури:

1. Мережа завершується шаром Dense розміром 46. Це означає, що для кожного вхідного зразка мережа виводитиме 46-мірний вектор.

2. Останній шар використовує функцію активації somax. Він означає, що мережа виводитиме розподіл ймовірностей за 46 різними класами - для кожного зразка на вході мережа повертатиме 46-мірний вектор, де output[i] - ймовірність приналежності зразка класу i. Сума 46 елементів завжди дорівнюватиме 1.

Найкращим варіантом у цьому випадку є використання функції втрат

categorical_crossentropy. Вона визначає відстань між розподілами ймовірностей: у разі між розподілом ймовірності на виході мережі і істинним розподілом міток. Мінімізуючи відстань між цими двома розподілами, ми вчимо мережу виводити результат, максимально близький до істинних позначок:

```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

6.3.15 Підсумки

Зазвичай вихідні дані доводиться піддавати деякої попередньої обробки, щоб передати в нейронну мережу як тензоров. Послідовності слів можна перетворити на бінарні вектори, але існують також інші варіанти.

Стек повнозв'язкових рівнів з функцією активації relu здатний вирішувати широке коло завдань (включаючи класифікацію емоційного забарвлення), і ви, ймовірно, найчастіше використовуватимете цю комбінацію.

У задачі бінарної класифікації (з двома вихідними класами) однією одиницею та функцією активації sigmoid: результатом мережі має бути скалярне значення в діапазоні між 0 і 1, що представляє ймовірність.

З таким скалярним результатом, що отримується за допомогою сигмоїдної функції, у задачах бінарної класифікації слід використовувати функцію втрат binary_crossentropy.

У загальному випадку оптимізатор rmsprop є гарним вибором для будь-яких завдань. Цей аспект завдає найменше клопоту.

6.4 Контрольні питання:

1. Яке призначення бібліотеки Keras?
2. Які функції має Keras?
3. Що таке бінарна класифікація? Наведіть приклад.
4. Що таке рівні та приховані рівні? Порівняйте. Наведіть приклади
5. Які етапи створення послідовної моделі? Наведіть приклад.
6. Які існують вбудовані набори даних в Keras?
7. Які існують функції активації? Для яких задач рекомендовано їх використовувати?
8. Які існують функції втрат ? Для яких задач рекомендовано їх використовувати?
9. Чому виконується попередня обробка первинних даних?
10. Що таке бінарні вектори? Де їх використовують.
11. Що таке стек повнозв’язних рівнів з функцією relu? Яке його призначення?
12. В яких задачах ортильно є використання binary_crossentropy?
13. Що таке оптимізатор rmsprop? Для яких задач рекомендовано використати?

ЛАБОРАТОРНА РОБОТА №7
РОЗПІЗНАВАННЯ ОБ'ЄКТІВ ЗОБРАЖЕННЯ НА ОСНОВІ МАШИННОГО НАВЧАННЯ

7.1 Мета роботи.

Навчитися розпізнавати об'єкти на зображеннях (Object Recognition in Photographs); проводити класифікацію даних; дослідити роботу рівня Dropout; ознайомитись із загортковими нейронними мережами.

7.2 Завдання до лабораторної роботи.

Частина 1. Необхідно розпізнати рукописні цифри мовою Python.

Рівень 1 (від 60 до 73 балів)

1. Використати набір даних MNIST вже входить до складу Keras з Лістінгу 1.

Лістінг 1

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist
(train_images, train_labels),(test_images, test_labels) = mnist.load_data()
```

Розпізнати цифри відповідно до варіанту (таб.7.1, рис.7.1), де № варіанту відповідає порядковому номеру вашої ПІБ у списку групи вибіркової дисципліни та вивести відповідне повідомлення про цифру на екран. Зображення взяти у викладача.

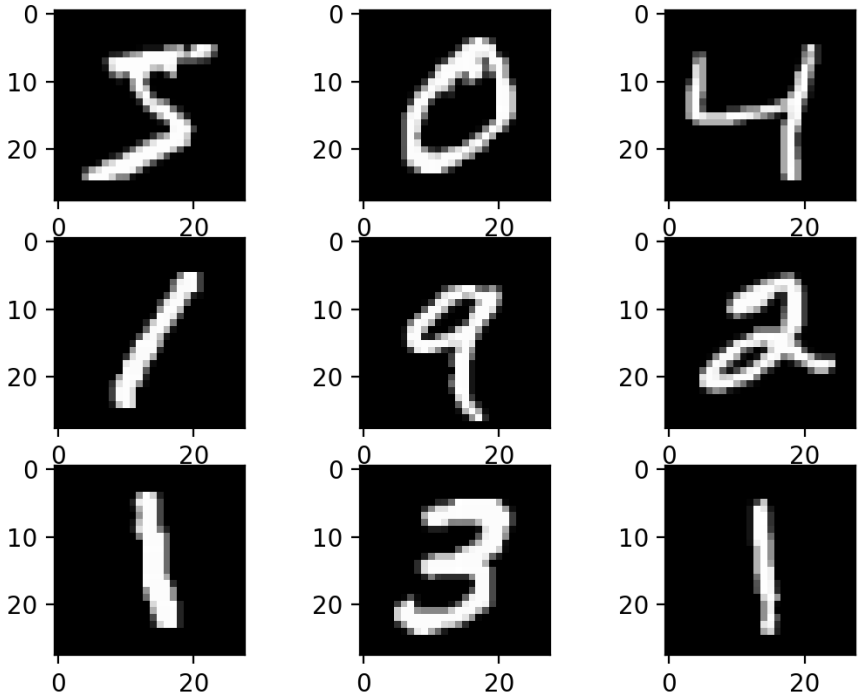


Рисунок 7.1 – Зображення для класифікації

Таблиця 7.1 – Індивідуальне завдання до частини 1

№	Завдання
1	Розпізнати цифру 1. 
2	Розпізнати цифру 10. 
3	Розпізнати цифру 2. 
4	Розпізнати цифру 3. 
5	Розпізнати цифру 4. 
6	Розпізнати цифру 5. 
7	Розпізнати цифру 6. 
8	Розпізнати цифру 7. 
9	Розпізнати цифру 8. 
10	Розпізнати цифру 9. 
11	Розпізнати цифру 0. 
12	Розпізнати цифру 12.

	
13	Розпізнати цифру 13. 
14	Розпізнати цифру 14. 
15	Розпізнати цифру 15. 

2. Відкорегуйте маств чисел Numpy до заданої умови.

3. Створіть базову архітектуру мережі (лістинг 2).

Лістінг 2

```
from tensorflow.keras.layers import Dense, Activation, Flatten
from tensorflow.keras.models import Sequential
model = Sequential()
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

4. Налаштувати параметри мережі для етапу компіляції:

– функцію втрат, яка визначає, як мережа повинна оцінювати якість своєї роботи на навчальних даних та, відповідно, як коригувати її у правильному напрямку;

– оптимізатор - механізм, за допомогою якого мережа буде оновлювати себе, спираючись на дані та функцію втрат, що спостерігаються;

– метрики для моніторингу на етапах навчання та тестування – тут нас цікавитиме лише точність (частка правильно класифікованих зображень).

5. Навчити мережу розпізнавати цифри на основі бібліотеки Keras метод *fit* мережі — він намагається адаптувати (fit) модель під навчальні дані.

6. Вивести результат на екран.

Рівень 2 (від 74 до 95 балів)

Частина 2. Розпізнати цифри з таблиці на основі Keras.

1. Розпізнати цифри з таблиці за варіантом (таб.7.2), де № варіанту відповідає порядковому номеру вашої ПІБ у списку групи вибіркової дисципліни.

Таблиця 7.2 – Індивідуальне завдання до частини 2

№	Завдання				
1	40363	3210	3033	4436	8467
2	44704	3136	3026	1500	2876
3	40072	3286	3344	8489	
4	41112	3111	3020	2321	8749
5	39154	3084	2892	2893	25639
6	36606	2961	3053	7569	
7	39937	2966	2964		1310
8	41893	3253	2925	748	6964
9	39579	3152	4490	9504	
10	39533	2213	2958	1430	
11		4454	3375	2511	13399
12		3150	3128	6460	10166
13		3113	3061	2821	13312
14		3058	2981	8415	18169
15		3312	3143	11461	

2. Створити модель згортокової нейроної мережі.

3. Виконати навчання (тренування) нейроної мережі.

4. Провести аналіз мережі та яку точність розпізнавання отримали. Знайти архітектуру мережі, за якої точність класифікації буде не менше **95%.**

5. Представити результат у вигляді графічного інтерфейсу (GUI), де зліва завантаження зображення за варіантом, праворуч – цифри, які зображено на забражені; знизу – назва архітектури та точність розпізнавання.

6. Провести оцінку роботи моделі.

Частина 3. Додаткова для отримання оцінки 96-100 балів. Розпізнавання об’єктів зображення на основі CIFAR-10.

1. Провести розпізнавання об’єктів зображення з датасету CIFAR-10, що містить 32 х 32 х 3 пікселів за варіантом (таб.7.3, рис.7.2), де № варіанту відповідає порядковому номеру вашої ПІБ у списку групи вибіркової дисципліни.

Таблиця 7.3 – Індивідуальне завдання до частини 3

№	Завдання
1	Розпізнати котів породи Сіамська
2	Розпізнати котів породи Британська
3	Розпізнати котів породи Персидська
4	Розпізнати котів породи Сфінкс

5	Розпізнати котів породи Саванна
6	Розпізнати собак породи Англійський бульдог
7	Розпізнати собак породи Німецька овчарка
8	Розпізнати собак породи Пудель
9	Розпізнати собак породи Хаски
10	Розпізнати собак породи Шпиц
11	Розпізнати автомобілі легкові
12	Розпізнати автомобілі вантажівки
13	Розпізнати парусні кораблі
14	Розпізнати птаха Бджолоїдка
15	Розпізнати чорнудощову жабу

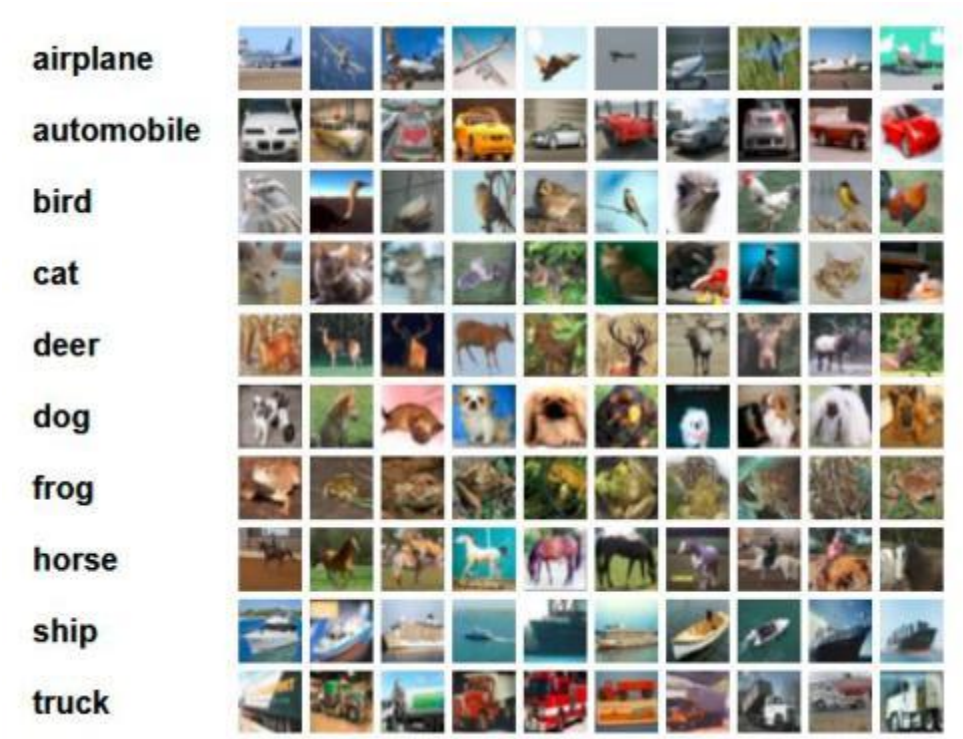


Рисунок 7.2 – Зображення CIFAR-10

2. Описати метод для реалізації поставленої задачі в п1.

3. Представити структурну схему обраної моделі.

4. Описати задані гіперпараметри та яку кількість і чому обрано.

5. Побудувати та навчити згорткову нейронну мережу розпізнавати об’єкти з табл.3.

5. Дослідити роботу мережі за різних розмірів ядра згортки.

6. Представити результат у вигляді графічного інтерфейсу (GUI), що містить: завантаження зображення за варіантом; класифікатор за завданням; назва архітектури та точність розпізнавання.

7.3 Теоретична частина.

7.3.1. Набір даних Fashion Mnist

Набір даних Fashion Mnist складається із 70 000 чорно-білих зображень, кожне з яких має розмір 28x28 та складається з десяти типів зображень одягу. Інший набір даних **MNIST – це рукописні цифри**. Порівняно з Fashion MNIST він більш складний і підходить для перевірки алгоритмів.

7.3.1.1 Розпізнавання рукописних цифр

Розпізнавання рукописних цифр – це здатність комп'ютера впізнавати написані від руки цифри. Для машини це не найпростіше завдання, адже кожна написана цифра може відрізнятись від еталонного написання. У випадку з розпізнаванням рішенням є те, що ПК здатний впізнавати цифру на зображенні.

Для реалізації розпізнавання рукописних цифр та інших об’єктів можна використати різні бібліотеки. В даній роботі розглянемо на прикладі вже відомої бібліотеки (з лабораторної роботи 6) – Keras.

7.3.2. Бібліотеки рекомендовані для розпізнавання

Додайте бібліотеки, які хочете використати з лістингу 1.

Лістінг 1

```
import matplotlib.pyplot as plt
import numpy as np
import sklearn
import pandas as pd
import os
import sys
import time
import tensorflow as tf
from tensorflow import keras
print(tf.__version__)
for moudle in mpl,np,pd,sklearn,tf,keras:
    print(moudle.__name__,moudle.__version__)
```

7.3.2.1 Імпорт даних

Потім імпортуйте набір даних. Набір даних ділиться на навчальний набір і тестовий набір, а навчальний набір далі ділиться на перевірочний набір (перші 5000) та навчальний набір, і вихідні дані виробляються у вигляді масиву. Розглянемо на прикладі *датасет Fashion Mnist (рис.7.3)*.

Лістінг 2

```
fashion_mnist = keras.datasets.fashion_mnist # Імпортувати дані
  (x_train_all, y_train_all), (x_test, y_test) = fashion_mnist.load_data () # Розділити навчальний набір та тестовий набір
```

```
 x_valid, x_train = x_train_all [: 5000], x_train_all [5000:] # далі розбивається на навчальний набір та набір перевірки
y_valid,y_train=x_train_all[:5000],y_train_all[5000:]
print(x_valid.shape,y_valid.shape)
print(x_train.shape,y_train.shape)
print(x_test.shape,y_test.shape)
```

7.32.2 Обробка нормалізації

Потім дані нормалізуються. Так звана нормалізація полягає у відображенні набору даних у діапазон, щоб уникнути впливу, викликаного деякими корисними даними, та полегшити обчислення

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train_scaled=scaler.fit_transform(x_train.astype(np.float32).reshape(-1,1)).reshape(-1,28,28)
x_valid_scaled=scaler.transform(x_valid.astype(np.float32).reshape(-1,1)).reshape(-1,28,28)
x_test_scaled=scaler.transform(x_test.astype(np.float32).reshape(-1,1)).reshape(-1,28,28)
```

Є функції для виведення наборів даних – *def show_single_image (img_arr)*

Лістинг 3

```
def show_single_image (img_arr): # Показати одне зображення
        plt.imshow (img_arr, cmap = "binary") # Двійковий дисплей
    plt.show()
show_single_image(x_train[0])
def show_imgs(n_rows,n_cols,x_data,y_data,class_names):
    assert len(x_data)==len(y_data)
    assert n_rows*n_cols<len(x_data)
    plt.figure(figsize=(n_cols*1.4,n_rows*1.6))
    for row in range(n_rows):
        for col in range(n_cols):
                    index = n_cols * row + col # Обчислити поточну позицію
                    plt.subplot (n_rows, n_cols, index + 1) # Намалюйте підзаголовок, намалюйте кілька зображень на одному інтерфейсі
            plt.imshow(x_data[index],cmap="binary",
                                    інтерполяція = 'найближчий') # Відобразити кожне зображення та виконати операцію інтерполяції
                    plt.axis ('off') # Вимкнути систему координат
                    plt.title (class_names [y_data [index]]) # Додати заголовок
        plt.show()
```

```
class_names=['T-shirt','Trousers','Pullover','Dress','Coat','Sandal','Shirt',
             'Sneaker', 'Bag', 'Ankle boot']
show_imgs(3,5,x_train,y_train,class_names)
```

Рисунок 7.3 – Результат датасет Fashion Mnist

7.3.3. Створення нейронної мережі

Побудуємо для прикладу нейронну мережу з чотирма рівнями на основі методу sgd.

Лістинг 4

```
model=keras.models.Sequential()
 # Розгорніть зображення і зведіть матрицю 28*28 в одномірний вектор 28*28
model.add(keras.layers.Flatten(input_shape=[28,28]))
 # Повнозв'язковий шар, свого роду нейронна мережа, для дослідження нейронної мережі по шарах, нижній та верхній блоки пов'язані один за одним
 model.add (keras.layers.Dense (300, activate = "relu")) # Додайте два, relu: y = max (0, x)
model.add(keras.layers.Dense(100,activation="relu"))
 # Контролювати висновок вектор довжини 10.
 #softmax перетворює вектор на розподіл ймовірностей, x = [x1, x2, x3]
 #softmax operation: y = [e ^ x1 / sum, e ^ x2 / sum, e ^ x3 / sum], (sum = e ^ x1 + e ^ x2 + e ^ x3)
model.add(keras.layers.Dense(10,activation="softmax"))
```

```
 # Додати на графік функцію втрат та метод оптимізації.
 #y – вектор, довжина якого дорівнює кількості вибірок, тому y – значення, тому використовуйте sparse_categorical_crossentropy
model.compile(loss="sparse_categorical_crossentropy",
                    optimizer = "sgd", metrics = ["precision"]) # параметри (функція втрат, метод коригування, інші індикатори, що цікавлять)
```

7.3.4. Тренування нейронної мережі

Для тестування пропонуємо використати функцію *model.fit* (10 разів) і перевірки результатів.

Лістинг 5

```
history=model.fit(x_train_scaled,y_train,epochs=10,validation_data=(x_test_scaled,y_test))
```

4.1 Побудова графіку

Після навчання мережі можна переглянути точність розробленого методу у вигляді графіку. Приклад представлено в лістингу 6.

Лістинг 6

```
test_loss, test_acc = model.evaluate (x_test_scaled, y_test) # Перегляд
print('\nTest accuracy:', test_acc)
def plot_learning_curves(history):

    pd.DataFrame(history.history).plot(figsize=(8,5))
    plt.grid(True)
    plt.gca().set_ylim(0,1)
    plt.show()
plot_learning_curves(history)
```

5. Етапи розпізнавання рукописних цифр мовою Python

Дане завдання є першим рівнем лабораторної роботи №7. Етапи реалізації ідентичні з п.1 по п.4 тільки для іншого датасету. Етапи виконання:

1. Імпорт бібліотек та завантаження набір даних.

```
# скачуємо дані та поділяємо на надор для навчання та тесовий
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

2. Попередня обробка даних. Дані із зображення не можна прямо передати в модель, тому спочатку потрібно виконати певні операції, обробивши дані, щоб нейронна мережа з ними працювала. Розмірність тренувальних даних - (60000, 28, 28). Модель згорткової нейронної мережі потребує одну розмірність, тому потрібно перебудувати форму (60000, 28, 28, 1). Тому, необхідно виконати перетворення векторних класів на бінарні матриці.

```
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

3. Створення моделі нейронної мережі. Наступний етап – створення моделі згорткової нейронної мережі. Вона переважно складається з пакувальних та шарів підвиборки. Модель краще працює з даними, представленими як сіткові структури. Ось чому така мережа відмінно підходить для завдань із класифікацією зображень. Шар виключення використовується для відключення окремих нейронів та під час тренування. Він зменшує ймовірність перенавчання. Потім відбувається компіляція моделі за допомогою оптимізатора Adadelta.

4. Тренування моделі мережі. Функція model.fit() у Keras почне тренування моделі. Вона приймає тренувальні, валідаційні дані, епохи (epoch) та розмір батча (batch). Тренування моделі займає деякий час. Після цього ваги та визначення моделі зберігаються у файлі mnist.h5.

5. Оцінка моделі. Набір даних містить 10 000 зображень, які використовуються для оцінки якості роботи моделі. В п.4.1 описано оцінку моделі у вигляді графіка.

6. ***Створення графічного інтерфейсу*** для розпізнавання (класифікації) цифр. Для графічного інтерфейсу створимо новий файл, у якому буде інтерактивне вікно для малювання цифр на полотні та кнопка, яка відповідає за процес розпізнавання. Бібліотека Tkinter є частиною стандартної бібліотеки Python. Функція ***predict_digit()*** приймає вхідне зображення, а потім використовує натреновану мережу для передбачення. Потім створюємо клас App, який відповідатиме за побудову графічного інтерфейсу програми. Створюємо полотно, де можна малювати, захоплюючи події миші. Кнопка активуватиме функцію ***predict_digit()*** і відображатиме результат. Або обрати інший варіант.

7.3.5. Підсумки

Визначте завдання та дані, на яких проводитиметься навчання. Зберіть ці дані та, якщо потрібно, анотуйте їх.

Виберіть міру успіху — показники, які можна було б відстежувати за перевірочними даними.

Виберіть протокол оцінки: чи оцінюватимете ви за вибраним із загальної вибірки перевірочним набором даних або методом перехресної перевірки за K блоками? Яка частина даних ви могли б використовувати для перевірки?

Напишіть першу модель, більш досконалу, ніж базовий випадок: модель, що має статистичну потужність.

Напишіть модель, яка має ефект перенавчання.

Виконайте регуляризацію моделі та налаштуйте її гіперпараметри, спираючись на оцінку якості за перевірочними даними. Багато досліджень у галузі машинного навчання зосереджені виключно на цьому кроці, проте не беріть до уваги загальну картину.

7.4 Контрольні питання:

1. Яке призначення бібліотеки Keras?
2. Які функції має Keras?

3. Що таке бінарна класифікація? Наведіть приклад.

4. Що таке рівні та приховані рівні? Порівняйте. Наведіть приклади

5. Які етапи створення послідовної моделі? Наведіть приклад.

6. Які існують вбудовані набори даних в Keras?

7. Які існують функції активації? Для яких задач рекомендовано їх використовувати?

8. Які існують функції втрат ? Для яких задач рекомендовано їх використовувати?

9. Чому виконується попередня обробка первинних даних?

10. Що таке бінарні вектори? Де їх використовують.

11. Що таке стек повнозв'язних рівнів з функцією relu? Яке його призначення?

12. В яких задачах ортильно є використання binary_crossentropy?

13. Що таке оптимізатор rmsprop? Для яких задач рекомендовано використати?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура та правила оформлювання.
2. ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання.
3. Субботін С.О. Нейронні мережі : теорія та практика : навчальний посібник. Житомир : Вид.О.О. Євенок, 2020. 184 с
4. Троцько В.В. Методи штучного інтелекту : навчально-методичний посібник. Київ : Університет економіки та права «КРОК», 2020. 86 с.
5. Жуковська О., Файнзільберг Л. Математичні моделі прийняття колективних рішень: монографія. - К.: Осіта України, 2018
6. F. Chollet. Deep Learning with Python. – Manning Publications Co, 2017. – 384 p. ISBN 9781617294433
7. Основи програмування (Python, Java) : лабораторний практикум / Смотр О., Придатко О., Малець І. – Львів : ЛДУ БЖД, 2019. – 134 с.
8. Програмування мовою Python / О.М. Васильєв. — Тернопіль: Навчальна книга – Богдан, 2019. — 504 с.; іл.
9. Шаховська Н. Б. Системи штучного інтелекту: навч. посібник / Н. Б. Шаховська, Р. М. Камінський, О. Б. Вовк. Львів: Видавництво Львівської політехніки, 2018. 392 с.
10. Каштан В.Ю. Методичні вказівки до виконання лабораторних робіт з дисципліни “ Інтелектуальні інформаційні технології ” для студентів галузі знань 12 Інформаційні технології / В.Ю. Каштан;Нац. техн. ун-т «Дніпровська політехніка». – Дніпро: НТУ «ДП», 2022. [Електронне видання].
11. Kashtan V.Yu. Information Technology Analysis of Satellite Data for Land Irrigation Monitoring / V. Yu. Kashtan, V. V. Hnatushenko, S. Zhir // 2021 IEEE International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo), Kyiv, Ukraine, November 29 – December 3, 2021, pp. 12-15.
12. Kashtan V.Yu. Voxel Approach to the Shadow Formation Process in Image Analysis / V. Yu. Kashtan, V. V. Hnatushenko, Vik. Hnatushenko, O. Reuta, I. Udovyk // The 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAASC) 22-25 September, 2021, Cracow, Poland, pp. 33-37.
13. Каштан В.Ю. Дешифрування автодоріг на цифрових космічних знімках на основі нейронних мереж / В.Ю. Каштан, В.В. Гнатушенко, О.Г. Баглай// XVI міжнародна конференція «Проблеми використання інформаційних технологій в освіті, науці та промисловості» / Збірник наукових праць № 6. – м. Дніпро. – 15 грудня 2021, С.71 – 75.
14. Kashtan V.Yu. Deep Learning Technology for Automatic Burned Area Extraction Using Satellite High Spatial Resolution Images / V. Yu. Kashtan, V. V.Hnatushenko // // Lecture Notes in Computational Intelligence and Decision

Making. ISDMCI 2022. Advances in Intelligent Systems and Computing, vol 1246. Pp. 55-76. Springer, Cham.

Додаток А. Титульний аркуш до лабораторної роботи

Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»

Кафедра інформаційних технологій та комп'ютерної інженерії

Лабораторна робота №1

з дисципліни:

«Інтелектуальні інформаційні технології»

Виконав (ла):студент (ка) групи 123-20м-1

Денисенко О.Д.

Перевірила:доцентка кафедри ІТ та КІ

Каштан В.Ю.

Дніпро

2022

Додаток Б. Завдання до лабораторної роботи №6.

Варіант 1

```
import numpy as np


def gen_rect(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    w = np.random.randint(size // 10, size // 2)
    h = np.random.randint(size // 10, size // 2)
    img[x:x + w, y:y + h] = 1
    return img


def gen_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    for i in range(0, size):
        for j in range(0, size):
            if (i-x)**2 + (j-y)**2 <= r**2:
                img[i, j] = 1
    return img


def gen_empty_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    dr = np.random.randint(1, 10) + r
    for i in range(0, size):
        for j in range(0, size):
            if r**2 <= (i - x) ** 2 + (j - y) ** 2 <= dr ** 2:
                img[i, j] = 1
    return img


def gen_h_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size-10)
    y = np.random.randint(10, size-10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img


def gen_v_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x - l:x + l, y - w:y + w] = 1
    return img


def gen_cross(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
```

```
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 5)
    w = 1
    img[x-l:x+l, y-w:y+w] = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img
```

Варіант 2

```
import gens
import numpy as np


def gen_data(size=500, img_size=50):
    c1 = size // 2
    c2 = size - c1

    label_c1 = np.full([c1, 1], 'Square')
    data_c1 = np.array([gens.gen_rect(img_size) for i in range(c1)])
    label_c2 = np.full([c2, 1], 'Circle')
    data_c2 = np.array([gens.gen_circle(img_size) for i in range(c2)])

    data = np.vstack((data_c1, data_c2))
    label = np.vstack((label_c1, label_c2))

    return data, label
```

Варіант 3

```
import numpy as np


def gen_rect(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    w = np.random.randint(size // 10, size // 2)
    h = np.random.randint(size // 10, size // 2)
    img[x:x + w, y:y + h] = 1
    return img


def gen_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    for i in range(0, size):
        for j in range(0, size):
            if (i-x)**2 + (j-y)**2 <= r**2:
                img[i, j] = 1
    return img


def gen_empty_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    dr = np.random.randint(1, 10) + r
    for i in range(0, size):
        for j in range(0, size):
            if r**2 <= (i - x) ** 2 + (j - y) ** 2 <= dr ** 2:
                img[i, j] = 1
    return img
```

```
def gen_h_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size-10)
    y = np.random.randint(10, size-10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img


def gen_v_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x - l:x + l, y - w:y + w] = 1
    return img


def gen_cross(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 5)
    w = 1
    img[x-l:x+l, y-w:y+w] = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img
```

Варіант 4

```
import gens
import numpy as np


def gen_data(size=500, img_size=50):
    c1 = size // 2
    c2 = size - c1

    label_c1 = np.full([c1, 1], 'Horizontal')
    data_c1 = np.array([gens.gen_h_line(img_size) for i in range(c1)])
    label_c2 = np.full([c2, 1], 'Vertical')
    data_c2 = np.array([gens.gen_v_line(img_size) for i in range(c2)])

    data = np.vstack((data_c1, data_c2))
    label = np.vstack((label_c1, label_c2))

    return data, label
```

Варіант 5

```
import numpy as np


def gen_rect(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    w = np.random.randint(size // 10, size // 2)
    h = np.random.randint(size // 10, size // 2)
    img[x:x + w, y:y + h] = 1
    return img
```

```
def gen_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    for i in range(0, size):
        for j in range(0, size):
            if (i-x)**2 + (j-y)**2 <= r**2:
                img[i, j] = 1
    return img


def gen_empty_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    dr = np.random.randint(1, 10) + r
    for i in range(0, size):
        for j in range(0, size):
            if r**2 <= (i - x) ** 2 + (j - y) ** 2 <= dr ** 2:
                img[i, j] = 1
    return img


def gen_h_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size-10)
    y = np.random.randint(10, size-10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img


def gen_v_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x - l:x + l, y - w:y + w] = 1
    return img


def gen_cross(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 5)
    w = 1
    img[x-l:x+l, y-w:y+w] = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img
    return img
```

Варіант 6

```
import gens
import numpy as np


def random_line(img_size=50):
    p = np.random.random([1])
    if p < 0.5:
        return gens.gen_v_line(img_size)
```

```
    else:
        return gens.gen_h_line(img_size)


def gen_data(size=500, img_size=50):
    c1 = size // 2
    c2 = size - c1

    label_c1 = np.full([c1, 1], 'Cross')
    data_c1 = np.array([gens.gen_cross(img_size) for i in range(c1)])
    label_c2 = np.full([c2, 1], 'Line')
    data_c2 = np.array([random_line(img_size) for i in range(c2)])

    data = np.vstack((data_c1, data_c2))
    label = np.vstack((label_c1, label_c2))

    return data, label
```

Варіант 7

```
import numpy as np


def gen_rect(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    w = np.random.randint(size // 10, size // 2)
    h = np.random.randint(size // 10, size // 2)
    img[x:x + w, y:y + h] = 1
    return img


def gen_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    for i in range(0, size):
        for j in range(0, size):
            if (i-x)**2 + (j-y)**2 <= r**2:
                img[i, j] = 1
    return img


def gen_empty_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    dr = np.random.randint(1, 10) + r
    for i in range(0, size):
        for j in range(0, size):
            if r**2 <= (i - x) ** 2 + (j - y) ** 2 <= dr ** 2:
                img[i, j] = 1
    return img


def gen_h_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size-10)
    y = np.random.randint(10, size-10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img
```

```
def gen_v_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x - l:x + l, y - w:y + w] = 1
    return img


def gen_cross(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 5)
    w = 1
    img[x-l:x+l, y-w:y+w] = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img
```

Варіант 8

```
import gens
import numpy as np


def gen_data(size=500, img_size=50):
    c1 = size // 2
    c2 = size - c1

    label_c1 = np.full([c1, 1], 'Square')
    data_c1 = np.array([gens.gen_rect(img_size) for i in range(c1)])
    label_c2 = np.full([c2, 1], 'Circle')
    data_c2 = np.array([gens.gen_empty_circle(img_size) for i in range(c2)])

    data = np.vstack((data_c1, data_c2))
    label = np.vstack((label_c1, label_c2))

    return data, label
```

Варіант 9

```
import numpy as np

def gen_empty_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    dr = np.random.randint(1, 10) + r
    for i in range(0, size):
        for j in range(0, size):
            if r**2 <= (i - x) ** 2 + (j - y) ** 2 <= dr ** 2:
                img[i, j] = 1
    return img


def gen_h_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size-10)
    y = np.random.randint(10, size-10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
```

```
    img[x-w:x+w, y-l:y+l] = 1
    return img


def gen_v_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x - l:x + l, y - w:y + w] = 1
    return img


def gen_cross(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 5)
    w = 1
    img[x-l:x+l, y-w:y+w] = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img
```

Варіант 10

```
import numpy as np


def gen_rect(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    w = np.random.randint(size // 10, size // 2)
    h = np.random.randint(size // 10, size // 2)
    img[x:x + w, y:y + h] = 1
    return img


def gen_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    for i in range(0, size):
        for j in range(0, size):
            if (i-x)**2 + (j-y)**2 <= r**2:
                img[i, j] = 1
    return img


def gen_empty_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    dr = np.random.randint(1, 10) + r
    for i in range(0, size):
        for j in range(0, size):
            if r**2 <= (i - x) ** 2 + (j - y) ** 2 <= dr ** 2:
                img[i, j] = 1
    return img


def gen_h_line(size=50):
```

```
    img = np.zeros([size, size])
    x = np.random.randint(10, size-10)
    y = np.random.randint(10, size-10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img


def gen_v_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x - l:x + l, y - w:y + w] = 1
    return img


def gen_cross(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 5)
    w = 1
    img[x-l:x+l, y-w:y+w] = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img
```

Варіант 11

```
import gens
import numpy as np


def gen_k_cross(k, img_size=50):
    img = np.zeros([img_size, img_size])
    for i in range(k):
        img += gens.gen_cross(img_size)
    img[np.nonzero(img)] = 1
    return img


def gen_data(size=500, img_size=50):
    c1 = size // 3
    c2 = c1
    c3 = size - (c1 + c2)

    label_c1 = np.full([c1, 1], 'One')
    data_c1 = np.array([gen_k_cross(1, img_size) for i in range(c1)])
    label_c2 = np.full([c2, 1], 'Two')
    data_c2 = np.array([gen_k_cross(2, img_size) for i in range(c2)])
    label_c3 = np.full([c3, 1], 'Three')
    data_c3 = np.array([gen_k_cross(3, img_size) for i in range(c3)])

    data = np.vstack((data_c1, data_c2, data_c3))
    label = np.vstack((label_c1, label_c2, label_c3))

    return data, label
```

Варіант 12

```
import numpy as np
```

```
def gen_h_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size-10)
    y = np.random.randint(10, size-10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img


def gen_v_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x - l:x + l, y - w:y + w] = 1
    return img


def gen_cross(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 5)
    w = 1
    img[x-l:x+l, y-w:y+w] = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img
```

Варіант 13

```
import numpy as np


def gen_rect(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    w = np.random.randint(size // 10, size // 2)
    h = np.random.randint(size // 10, size // 2)
    img[x:x + w, y:y + h] = 1
    return img


def gen_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    for i in range(0, size):
        for j in range(0, size):
            if (i-x)**2 + (j-y)**2 <= r**2:
                img[i, j] = 1
    return img


def gen_empty_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    dr = np.random.randint(1, 10) + r
    for i in range(0, size):
```

```
        for j in range(0, size):
            if r**2 <= (i - x) ** 2 + (j - y) ** 2 <= dr ** 2:
                img[i, j] = 1
    return img


def gen_cross(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 5)
    w = 1
    img[x-l:x+l, y-w:y+w] = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img
```

Варіант 14

```
import gens
import numpy as np


def gen_k_cross(k, img_size=50):
    img = np.zeros([img_size, img_size])
    for i in range(k):
        img += gens.gen_cross(img_size)
    img[np.nonzero(img)] = 1
    return img


def gen_data(size=500, img_size=50):
    c1 = size // 3
    c2 = c1
    c3 = size - (c1 + c2)

    label_c1 = np.full([c1, 1], 'One')
    data_c1 = np.array([gen_k_cross(1, img_size) for i in range(c1)])
    label_c2 = np.full([c2, 1], 'Two')
    data_c2 = np.array([gen_k_cross(2, img_size) for i in range(c2)])
    label_c3 = np.full([c3, 1], 'Three')
    data_c3 = np.array([gen_k_cross(3, img_size) for i in range(c3)])

    data = np.vstack((data_c1, data_c2, data_c3))
    label = np.vstack((label_c1, label_c2, label_c3))

    return data, label
```

Варіант 15

```
import numpy as np


def gen_rect(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    w = np.random.randint(size // 10, size // 2)
    h = np.random.randint(size // 10, size // 2)
    img[x:x + w, y:y + h] = 1
    return img


def gen_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
```

```
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    for i in range(0, size):
        for j in range(0, size):
            if (i-x)**2 + (j-y)**2 <= r**2:
                img[i, j] = 1
    return img


def gen_empty_circle(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(0, size)
    y = np.random.randint(0, size)
    r = np.random.randint(size // 10, size // 3)
    dr = np.random.randint(1, 10) + r
    for i in range(0, size):
        for j in range(0, size):
            if r**2 <= (i - x) ** 2 + (j - y) ** 2 <= dr ** 2:
                img[i, j] = 1
    return img


def gen_h_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size-10)
    y = np.random.randint(10, size-10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img


def gen_v_line(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 2)
    w = 1
    img[x - l:x + l, y - w:y + w] = 1
    return img


def gen_cross(size=50):
    img = np.zeros([size, size])
    x = np.random.randint(10, size - 10)
    y = np.random.randint(10, size - 10)
    l = np.random.randint(size // 8, size // 5)
    w = 1
    img[x-l:x+l, y-w:y+w] = 1
    img[x-w:x+w, y-l:y+l] = 1
    return img
```

Навчальне видання

Каштан Віта Юріївна

Методичні вказівки
до виконання лабораторних робіт
з дисципліни “Інтелектуальні інформаційні технології”.
для магістрів галузі знань 12 Інформаційні технології ”

Електронний ресурс

Видано
у Національному технічному університеті
«Дніпровська політехніка».
Свідоцтво про внесення до Державного реєстру ДК №1842 від 11.06.2004.
49005, *м*. *Дніпро*, *просп*. *Дмитра Яворницького*, 19.