МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

КУРСОВА РОБОТА

з дисципліни

“МАШИННЕ НАВЧАННЯ”

на тему: **СИНГУЛЯРНЕ РОЗКЛАДАННЯ ТА ЙОГО ПРАКТИЧНЕ ЗАСТОСУВАННЯ**

Студента 308 групи спеціальності
122 “Комп’ютерні науки”
Гецянина Дмитра Руслановича
Керівник
к. е. н., доц. Бойко Н.І.

Кількість балів:_____Оцінка _____

Члени комісії _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

_____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

_____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Львів – 2020

ЗМІСТ

ВСТУП

Оскільки технології становлять важливу частину повсякденного життя, виникають великі обсяги даних, які генерується щодня. Ці дані можуть бути різних форм, як текст, аудіо, відео, зображення тощо. Однак, для їхньої обробки та аналізу потрібно виділити величезну кількість машинних та людських ресурсів. Та це тільки одна з проблем, яка виникає внаслідок створення великого об'єму даних, ще одна з них - це їхнє зберігання. Зберігання нестиснених даних є дуже дорогим із точки зору вартості. Також передача нестиснених даних вимагає великої пропускної здатності. Через ці причини методи компресії даних перед зберіганням та передачею стали важливою сферою дослідження; особливо в таких сферах, як штучний інтелект, розпізнавання образів, обробка сигналів тощо.

Текст та зображення - одні із найпопулярніших методів обміну даними в соціальних мережах. Оскільки ми часто користуємось соціальними мережами, де обмін зображеннями відіграє велику роль, ми часто обмінюємось ними для передачі інформації. Революція в галузі смартфонів і розумних пристроїв дозволили користувачам зручніше використовувати графічні дані. Таким чином, кожного дня створюється, обробляється, передається величезна кількість цифрових даних.

Курсова робота передбачає дослідження та аналіз теми практичного застосування та значимості методу сингулярного розкладання в сьогоденні.

Актуальність роботи полягає в зменшенні обсягу та розмірності даних для їхнього оптимального аналізу та обробки, що є важливим у сферах машинного навчання й не тільки.

Метою є дослідження теоретичних та практичних методів застосування сингулярного розкладання в обробці та аналізі даних.

Об’єкт дослідження – розкладання матриці, зв’язані з власними або сингулярними значеннями, **предмет –** дослідження способів застосування методу сингулярного розкладання для обробки та аналізу даних.

Основним **методом** дослідження є сингулярне розкладання матриці.

Практичне значення роботи полягає в дослідженні методу сингулярного розкладання у вирішенні задач сфер штучного інтелекту, від стиснення зображення до регресійного аналізу.

Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

1 ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Вивчення об’єкту дослідження

Представлення матриці *A* у вигляді добутку матриць (рис.1.1), які містять певні властивості (наприклад, ортогональність, симетричність, діагональність) – називається розкладанням матриці, або факторизацією матриці (розкладанням на множники) [5]. До прикладу можна привести QR-розкладання, де матриця *A* розкладається на ортогональну матрицю *Q* та верхню трикутну матрицю *R* (рис.1.1) та полярне розкладання, де будь яку квадратну матрицю *A* можна представити у вигляді *S* симетричної та *Q* ортогональної матриці [22]. У кожного класу матричних розкладань існує своя область застосування, так, наприклад, LU-розклад належить до розкладань, які розв’язують системи лінійних рівнянь [5, с.111].

До розкладань матриці, які пов'язані з власними, або сингулярними значеннями, належать: SVD-розкладання (рис. 1.3), спектральне розкладання, жорданова нормальна форма, розкладання Шура та QZ-розкладання. Результатом цих розкладань є матриці, які містять власні значення, або сингулярні (квадратний корінь із додатного власного значення [24]). Власне значення – це коефіцієнт, за яким масштабується власний вектор, а власний вектор – це ненульовий вектор

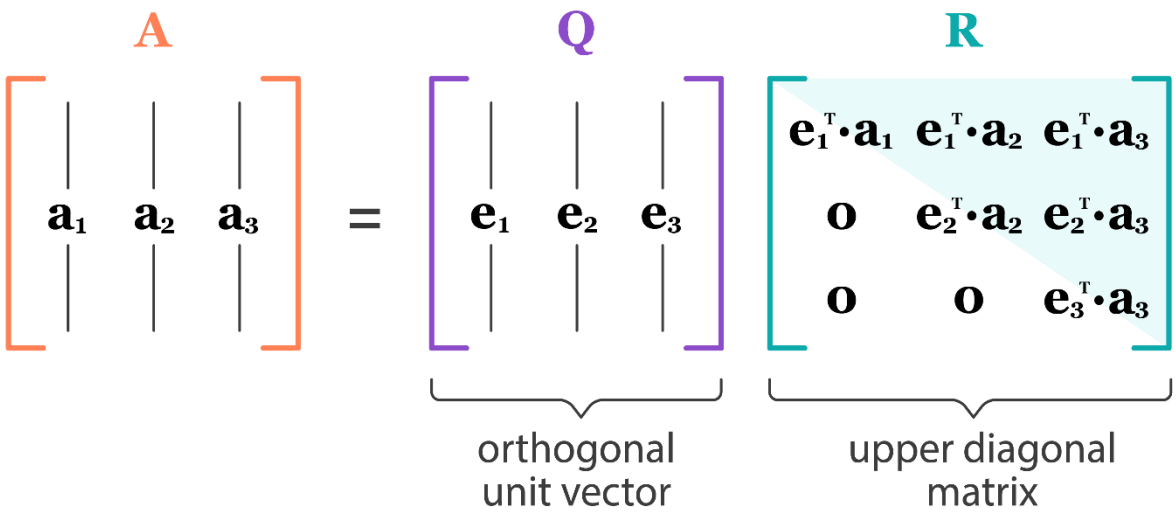


Рис. 1.1 Схематичний приклад QR розкладання

матриці A, який під дією лінійного перетворення, що задається цією матрицею A, не змінюють напрямку, але можуть змінювати довжину на коефіцієнт власного значення [20].

Для прикладу розглянемо векторну трансформацію, тобто зсув даних в одній системі кординат. Вектор, який зображений червоним кольром, змінив свій напрямок, а синій вектор – ні. Оскільки синій вектор не змінив свого напряму, то він є власним вектором цього зсуву, також його власне значення дорівнює одиниці, бо він не розтянувся. Також всі вектори, які колінеарні синьому, є власними векторами.

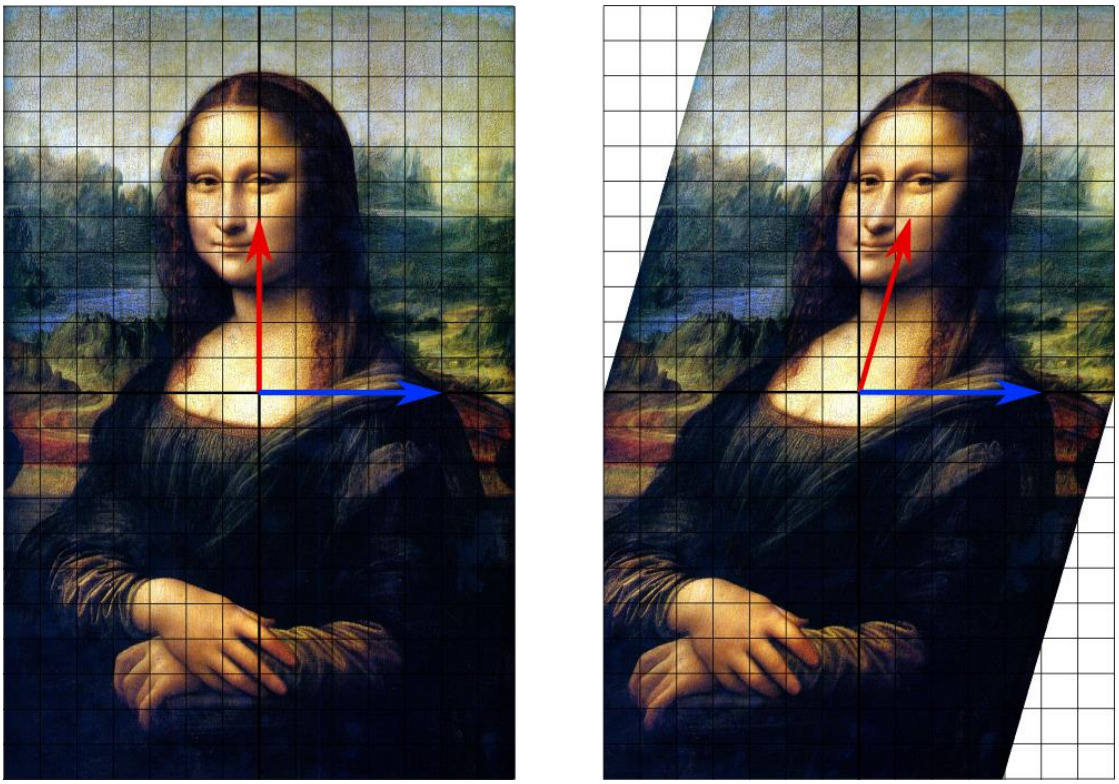


Рис. 1.2 Приклад власних векторів та власних значень

1.2 Огляд існуючих методів та засобів вирішення задачі

Розглянемо та порівняємо декілька методів розкладання матриці для отримання власних векторів та власних значень:

- Спектральне розкладання
- Розкладання Шура
- Приведення до нормальної жорданової форми

Спектральне розкладання – спосіб представлення матриці в канонічному вигляді, унаслідок чого матриця представлена з точки зору її власних значень та власних векторів. Однак тільки матриці, які діагоналізовані можуть бути так розкладені [19].

Розкладання Шура характеризується розкладанням квадратної матриці на унітарну, верхню трикутну та обернену унітарну. Власні значення знаходяться на діагоналі верхньої трикутної матриці [23].

Згідно з теоремою про жорданову нормальну форму [1], для будь-якої квадратної матриці A над алгебраїчно замкнутим полем K (наприклад полем комплексних чисел $\mathbb{C}$) існує така квадратна невироджена матриця C над полем K, що $J = C^{-1}AC$ є жордановою матрицею. При цьому J називається жордановою нормальною формою матриці A. Кожний блок J_λ називається жордановою клітинкою з власним значенням λ.

Отже, існує достатня кількість методів розкладання матриці для отримання їх сингулярних значень та векторів, однак, у всіх них є вагомий недолік – застосування під конкретний тип матриці. Однак, SVD-розкладання можливо застосовувати для будь-якої матриці.

1.3 Характеристика застосовуваних технологій

Для дослідження ми будемо використовувати SVD-розкладання (сингулярне), оскільки вже переконались у перевагах над іншими методами факторизації.

Сингулярне розкладання є методом факторизації матриць. Воно показує геометричну структуру матриці й дозволяє наочно представити наявні дані. Сингулярне розкладання використовується при вирішенні найрізноманітніших

завдань - від наближення методом найменших квадратів до стиснення і відновлення зображень. Як правило, виконує факторизацію вихідної матриці й однією з цілей є наблизити набір даних із великою кількістю вимірів до меншої кількості.

SVD-розкладання виконує розклад будь-якої матриці A на добуток трьох матриць, що може бути записано як:

$$\mathrm{A} = \mathrm{U}\Sigma\mathrm{V}^{\mathrm{T}} \quad (1)$$

де U і V^T – ортогональні матриці з розмірністю $m \times m$ і $n \times n$ відповідно (m – це кількість рядків, а n – кількість стовпців), $\Sigma -$ діагональна матриця з розмірністю $m \times n$. Діагональна матриця Σ складається з додатних значень, які називаються сингулярними значеннями матриці A. А з m стовпців матриці U та з n рядків матриці V^T складаються сингулярні вектори матриці A [16]. На рисунку нижче представлена схема розкладання.

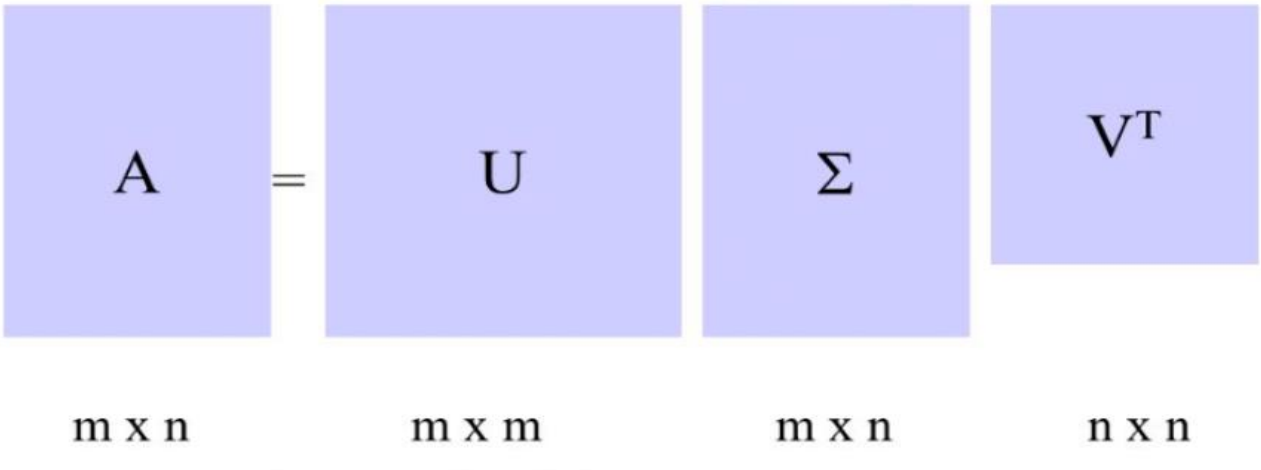


Рис.1.3 Схематичне зображення SVD-розкладу

Алгоритми сингулярного розкладання матриць мають обчислювальну складність порядку $O(n^3)$ [15], де n - число елементів матриці $m \times n$, і реалізовані в багатьох пакетах та бібліотеках мов програмування. Діагональні елементи матриці Σ (після сингулярного розкладання A) відсортовані й перші r чисел додатні: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$, а інші рівні нулю. Квадрати сингулярних чисел σ_i

SVD-розкладання матриці A збігаються з власними числами λ_i матриці AA^T.

Сингулярне розкладання матриць має безліч різних застосувань у цифровій обробці сигналів[11]. Проєкція даних у вигляді точок a_{ij} на зазначену пряму максимально зберігає їхні геометричні властивості в одновимірному представленні. Найпростіший варіант застосування цієї властивості – перетворення кольорового зображення в півтонове шляхом проєкції трьох кольорів на вісь, яка описується вектором v_1.

Одним із важливих застосувань сингулярного розкладання матриць є метод головних компонент, що використовується для зниження розмірності аналізованих даних у розпізнаванні образів [6].

У роботі [8] описана роль сингулярних чисел у зниженні адитивного електронного шуму в сенсорних реєструючих системах. Показано, що знизити рівень шуму можна за допомогою збереження найбільших сингулярних чисел, обнуління інших і відновлення вихідних даних з урізаною матрицею Σ. Стосовно до цифрових зображень, якщо сума перших k сингулярних чисел становить приблизно 90% від суми всіх сингулярних чисел, то зображення, відновлене на базі цих k чисел, візуально мало відрізна від початкового.

Ще одне із застосувань пов'язано з внесенням у зображення водяних знаків для захисту авторських прав [10]. SVD-розкладання використовується для кодування і стиснення зображень із мінімізацією втрат інформації [2].

Ознаки, обчислені на базі 10 менших сингулярних чисел, які отримані після сингулярного розкладання невеликих фрагментів зображення (32 × 32 пікселя), дозволяють класифікувати й сегментувати області з різною текстурою [14]. Періодично з'являються публікації про дослідження з розпізнавання людей за їхніми фотографіями за допомогою сингулярного розкладання матриць (наприклад, [17]). Останнім часом часто публікуються результати досліджень про використання функцій оцінки якості зображень на базі сингулярних чисел [12, 13].

1.4 Постановка задачі

Оскільки в роботі ми будемо розглядати способи застосування сингулярного розкладання матриці, то і використовувати ми будемо різні набори даних. Для стиснення зображення ми будемо використовувати зображення розміром 720 × 810 пікселів (див. рис. 3.1). Для методу головних компонент, розглянемо дата-сет, у якому записані дані, які були зібрані за допомогою білкового мікрочіпа H4 у людей із раком та без нього. Цей набір складається з даних про гени для 216 пацієнтів, 121 з яких мають рак яєчників, а 95 з них - ні. Для кожного пацієнта існує рядок даних, що містить інформацію про 4000 генів. У побудові регресії з застосування сингулярного розкладання ми будемо аналізувати набір даних про ринок житла в Бостоні. Дата-сет складається з 12 атрибутів, які корелюють із ціною на будинок, наприклад, рівень злочинності на душу населення та податок на майно.

2 АНАЛІТИЧНО-ПРОЄКТНИЙ РОЗДІЛ

2.1 Матричне наближення

На сингулярних числах базуються дві популярні норми, одна з них – це норма Фробеніуса, яка позначається як $||A||_F$, що дорівнює кореню квадратному із суми квадратів відстаней від безлічі точок у багатовимірному просторі до прямої, що проходить через вектор v_1 (перший вектор в SVD-розкладанні) [5]:

$$||X||_F^2 = \sum_{i=1}^{R} \sigma_i^2. \quad (2)$$

Зазвичай, у машинному навчанні та інших сферах застосування SVD, повне матричне розкладання не застосовується, а використовується оптимальне, або найкраще r-рангове наближення матриці (r – позначення рангу матриці) для оптимізації обчислень, оскільки повне розкладання доволі ресурсозатратне. Фактично, SVD забезпечує ієрархію низько-рангових наближень, оскільки r-рангове наближення отримується шляхом збереження ключових r сингулярних значень та векторів, а решта відкидається.

Шмідт узагальнив SVD для функціональних просторів (нескінченновимірний простір, точками якого є функції) і розробив теорему наближення, встановивши усічене SVD-розкладання як оптимальне наближення низького рангу основної матриці X [4]. Теорему наближення Шмідта було знайдено та досліджено Екартом та Юнгом [3], тому іноді її називають теоремою Екарта-Юнга.

Теорема звучить так : оптимальне r-рангове наближення матриці X, у сенсі методу найменших квадратів, задається r-ранговим усіченням SVD-розкладу матриці $\tilde{X}$:

$$argmin\ ||X - \tilde{X}||_F = \tilde{U}\tilde{\Sigma}\tilde{V}^T, \quad (3)$$

тут $\tilde{U}$ та $\tilde{V}^T$ позначають перші r ключових стовпців матриці U та V, а $\tilde{\Sigma}$ – ключовий блок розміром $r \times r$ матриці Σ (рис. 2.1).

У роботі базис усіченого SVD-розкладу (а також результат наближеної матриці $\tilde{X}$) ми будемо позначати як $\tilde{X} = \tilde{U}\tilde{\Sigma}\tilde{V}^T$. Оскільки Σ – це діагональна матриця, то r-рангове SVD наближення задається через суму r окремих одно

рангових матриць:

$$\tilde{X} = \sum_{k=1}^{r} \sigma_k u_k v_k^T = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T. \quad (4)$$

Це важлива властивість SVD, до якої ми повернемося ще не раз. Існують численні приклади дата-сетів, які містять великі розмірності, у результаті чого утворюється велика матриця X. Однак у даних часто є домінуючі низько вимірні патерни й усічений SVD базис $\tilde{U}$ забезпечує перетворення з високо вимірних просторів до низько вимірних. Перевага цього є можливість зменшити розмір та розмірність великих наборів даних, створивши зручну основу для візуалізації та аналізу.

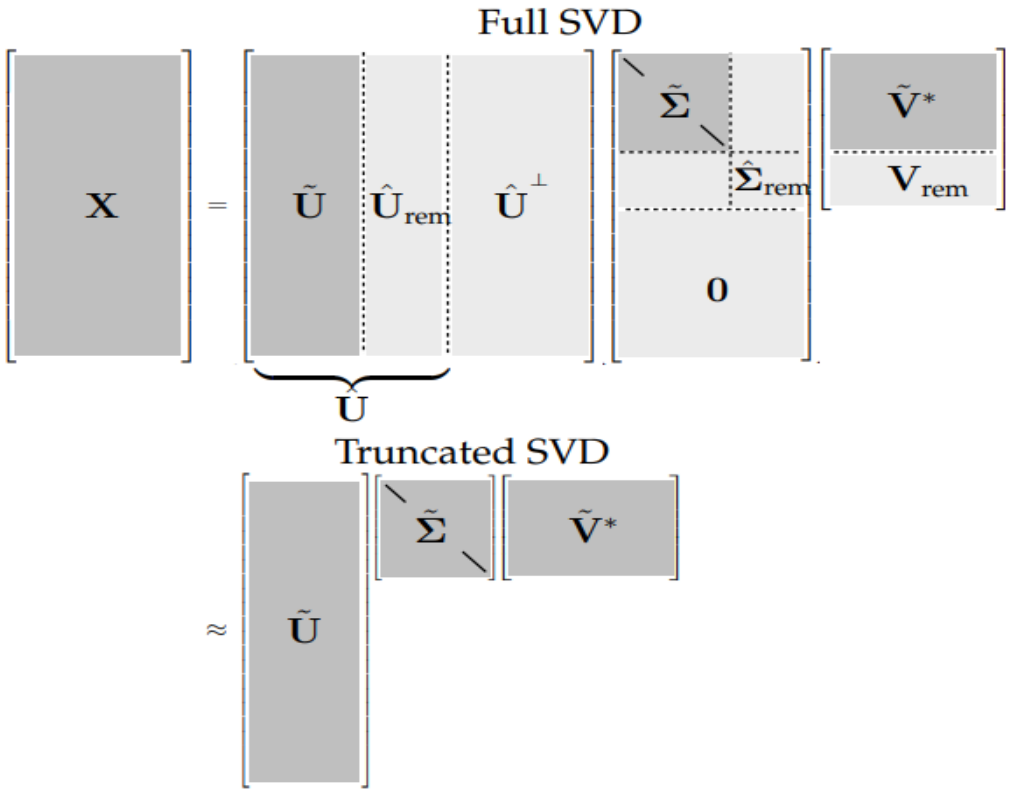


Рис. 2.1 Схематичне зображення r-рангового наближення SVD-розкладу

На схемі зображено *r*-рангового наближення, де $\hat{U}^{\perp}$ та 0 позначає залишок після "економічного" сингулярного розкладання (сингулярний розклад при якому, нульові елементи матриці відкидаються, відповідно й стовпці матриці *U*), а $\hat{U}_{rem}$,

$\hat{\Sigma}_{rem}, V_{rem}$ позначають залишок після усічення. В результаті ми отримуємо наближену матрицю.

2.2 Псевдообернена матриця

Псевдообернена матриця – це узагальнення матриці, оберненої для квадратних матриць, на прямокутні, де кількість рядків та стовпців не однакова. Найвідомішим методом є оберненням Мура-Пенроуза. Псевдообернення, або псевдоінверсія використовується для знаходження найкращого наближення розв'язку систем лінійних рівнянь [18].

Псевдоінверсію позначають як X^+, де X – матриця, яка була обернена. Один зі способів обчислення, є обчислення через сингулярне розкладання матриці X:

$$X^+ = V\Sigma^+V^T, \tag{5}$$

де Σ^+ – псевдоінверсія діагональної матриці Σ; матриці U, V^T, Σ обчислюємо через сингулярне розкладання.

Матриця Σ^+ обчислюється з матриці Σ, де кожний ненульовий елемент записується обернено:

$$\Sigma^+ = \begin{pmatrix} \frac{1}{\sigma_{1,1}} & 0 & 0 \\ 0 & \frac{1}{\sigma_{2,2}} & 0 \\ 0 & 0 & \frac{1}{\sigma_{3,3}} \end{pmatrix}, \tag{6}$$

де $\sigma_{1,1}$ – діагональний елемент матриці Σ.

Псевдоінверсія забезпечує один зі способів вирішення рівняння лінійної регресії, або множинної лінійної регресії, коли рядків більше, ніж стовпців, що є частим випадком, детальніше про це буде розглянуто у наступному підрозділі.

2.3 Лінійна регресія

Лінійна регресія - це метод моделювання взаємозв'язку між двома скалярними значеннями: вхідною змінною x та вихідною змінною y. Модель

передбачає, що y є лінійною функцією або зваженою сумою вхідної змінної [21].

Рівняння лінійної регресії може бути записано як:

$$Ax = b, \tag{7}$$

де A – матриця вхідних даних, де кожний стовпець – це атрибути, b – вектор вихідних значень для кожного рядка, а x – вектор коефіцієнтів (невідомі).

Якщо проблему переформулювати, то вона стає системою лінійних рівнянь, де значення x вектора невідомі. Цей тип системи називають перевизначеною, оскільки існує більше рівнянь, ніж невідомих, тобто кожен коефіцієнт використовується в кожному рядку даних. Цю проблему складно вирішити аналітичним шляхом, оскільки існує кілька рішень, кілька можливих значень коефіцієнтів, або зовсім відсутність рішень. Крім того, усі рішення матимуть певну помилку, оскільки немає лінії, яка проходитиме майже через усі точки, тому метод для розв'язання рівнянь повинен бути в змозі впоратися з цим. Зазвичай це досягається шляхом пошуку рішення, де значення x в моделі мінімізують за допомогою методу найменших квадратів.

Одним із застосування сингулярного розкладання є в якості основи для вирішення системи лінійних рівнянь для лінійної регресії, тобто вирішення проблеми найменших квадратів. SVD є більш стабільним і кращим підходом, у порівнянні з іншими методами. Після розкладання коефіцієнти можна знайти, обчисливши псевдоінверсію вхідної матриці A і помноживши її на вихідний вектор b:

$$x = A^{+}b = V\Sigma^{+}U^{T}b, \tag{8}$$

де V, Σ, U – вихідні матриці сингулярного розкладання, x –невідомі коефіцієнти, b– вихідний вектор, A^{+} – псевдоінверсія вхідної матриці A.

2.4 Метод головних компонент

Часто набір даних має багато стовпців, можливо, десятки, сотні, тисячі й більше. Моделювання даних із багатьма ознаками (атрибутами) є складним

завданням, і моделі, побудовані на основі даних, що включають нерелевантні ознаки, часто менше показують сутність моделі, ніж моделі, підготовлені з найбільш релевантних даних. Важко дізнатися, які особливості даних є релевантними, а які - ні. Методи автоматичного зменшення кількості стовпців набору даних називаються зменшенням розмірності, і, мабуть, найпопулярнішим методом називається аналіз головних компонент або коротко PCA. Цей метод використовується в машинному навчанні для створення проєкцій об'ємних даних, як для візуалізації, так і для навчальних моделей. Основою методу PCA є метод матричної факторизації з лінійної алгебри. Також можуть бути використані методи розкладання для отримання власних векторів та власних значень, наприклад, SVD.

Метод головних компонент (англ. principal component analysis, PCA) – один з основних способів зменшити розмірність даних, втративши найменшу кількість інформації. Він був винайдений Карлом Пірсоном у 1901 [9] році та доповнений і розширений Гарольдом Готелінґом в 1933 [7]. Застосовується в багатьох галузях, зокрема, в економетриці, біоінформатиці, обробці зображень, стиснення даних та в суспільних науках.

З математичної точки зору метод головних компонент являє собою ортогональне лінійне перетворення, яке відображає дані з вихідного простору ознак у новий простір меншої розмірності. При цьому перша вісь нової системи координат будується таким чином, щоб дисперсія даних уздовж неї була б максимальна. Друга вісь будується ортогонально першій так, щоб дисперсія даних уздовж неї, була б максимальною з решти можливих і т.д. Перша вісь називається першою головною компонентою, друга - другою і т.д.

Розглянемо як за допомогою сингулярного розкладання можна виконувати метод головних компонент. Нехай у нас є матриця X, розміром $n \times m$, де n – кількість рядків, а m –кількість стовпців. Припустимо, що дані відцентровані, тобто середні значення кожного з стовпців були відняті відняті від значень конкретного стовпця, і тепер середнє значення кожного стовпця дорівнює нулю.

Тоді $m \times m$ матриця C – коваріаційна матриця, яка обчислюється так:

$$C = \frac{X^T X}{n-1} \tag{9}$$

це симетрична матриця, яка може бути діагоналізованою, тобто записуватись так

$$C = VLV^T, \tag{10}$$

де V – матриця власних векторів (кожен стовпець – власний вектор), а L – діагональна матриця з власними значеннями λ_i, розташованими в порядку спадання. Власні вектори називаються головними осями. Проекціями даних на головні осі називають головними компонентами. Головна компонента j задається j-им стовпцем добутку матриць XV, а координати i-ої точки даних в новому просторі головних компонент задається i-им рядком добутку матриць XV.

Якщо виконати сингулярне розкладання матриці X за формулою (1), то ми отримаємо:

$$C = \frac{V\Sigma U^T U\Sigma V^T}{n-1} = V\frac{\Sigma^2}{n-1}V^T \tag{11}$$

З цієї формули праві сингулярні вектори V є головними осями, сингулярними значеннями є $\lambda_i = \frac{\sigma_i^2}{n-1}$, а головними компонентами є $XV = U\Sigma V^T V = US$.

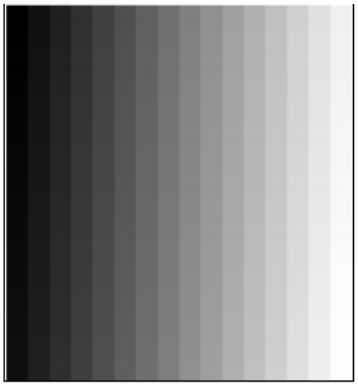
3 ПРАКТИЧНИЙ РОЗДІЛ

3.1 Стиснення зображень

В цьому підрозділі ми проведемо дослідження та аналіз застосування сингулярного розкладання в стисненні зображень та продемонструємо ідею наближення матриці. Для цього нам потрібно виконати певний алгоритм:

1. Перетворення зображення в матрицю (перетворення в матрицю тонів)
2. Застосування SVD-розкладу до матриці тонів
3. Візуалізація та аналіз отриманих матриць
4. Усічення отриманих матриць до значення r та отримання наближеної матриці.
5. Рекунструювання зображення

Крок 1. Розгляньте зображення квітки, яке зображено на рисунку 3.2. Це зображення розмірністю 720 × 810 пікселів, яке ми будемо використовувати для нашого стиснення. Для прикладу ми будемо використовувати чорно-біле зображення, яке ми трансформуємо в матрицю тонів, яку можна представити як $X \in \mathbb{R}^{n \times m}$, де $n \times m$– кількість пікселів по вертикалі та горизонталі відповідно. Кожний піксель зображення буде представлений елементом матриці, значення якого буде цілим числом від 0 (чорний піксель) до 255 (білий піксель). Для прикладу можна розглянути рисунок 3.1, на якому зправа представлене зображення



0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241
2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242
3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243
4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244
5	21	37	53	69	85	101	117	133	149	165	181	197	213	229	245
6	22	38	54	70	86	102	118	134	150	166	182	198	214	230	246
7	23	39	55	71	87	103	119	135	151	167	183	199	215	231	247
8	24	40	56	72	88	104	120	136	152	168	184	200	216	232	248
9	25	41	57	73	89	105	121	137	153	169	185	201	217	233	249
10	26	42	58	74	90	106	122	138	154	170	186	202	218	234	250
11	27	43	59	75	91	107	123	139	155	171	187	203	219	235	251
12	28	44	60	76	92	108	124	140	156	172	188	204	220	236	252
13	29	45	61	77	93	109	125	141	157	173	189	205	221	237	253
14	30	46	62	78	94	110	126	142	158	174	190	206	222	238	254
15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255

Рис. 3.1 Відповідність пікселя до його значення в межах від 0 до 255

256 різних чорно-білих пікселів, а зліва відповідне кожному пікселю число.

Рис. 3.2 Початкове зображення

Крок 2. Застосовуючи на матриці тонів нашого зображення SVD-розклад, за формулою (1111111) ми розкладуємо її на три різні матриці, для цього ми будемо використовувати функцію *svd,* яка знаходиться в бібліотеці *numpy* для мови програмування Python [25].

Крок 3. Отже, ми отримали три матриці, але головну цінність складає діагональна матриця Σ, бо за допомогою неї ми будемо вирішувати наскільки усікати інші матриці. Оскільки матриця сингулярних значень складається з 720 елементів, то для її дослідження раціональніше буде використовувати її візуалізацію. На рисунку 3.3 зображений графік, який показує величину кожного k сингулярного значення в нашій матриці Σ. Можна побачити, що перші елементи матриці мають значення від 10^5 до 10^4, а далі, приблизно з 20 значення, спостерігається плавний спад, що свідчить про розташування елементів у порядку спадання. Але в нашому дослідженні головне не тільки знати значення сингулярних елементів, а й їх величину відносно всієї матриці, для цього побудуємо графік кумулятивної суми (рис. 3.4), це сума послідовності чисел, яка

оновлюється кожного разу, коли нове число додається до послідовності,

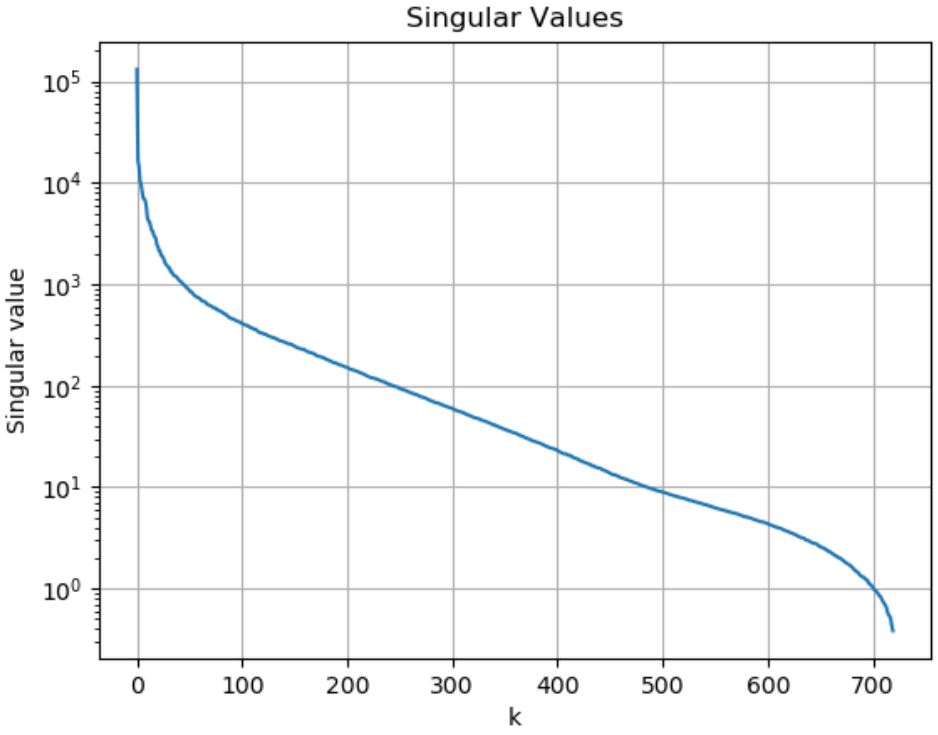


Рис. 3.3 Сингулярні значення σ_k

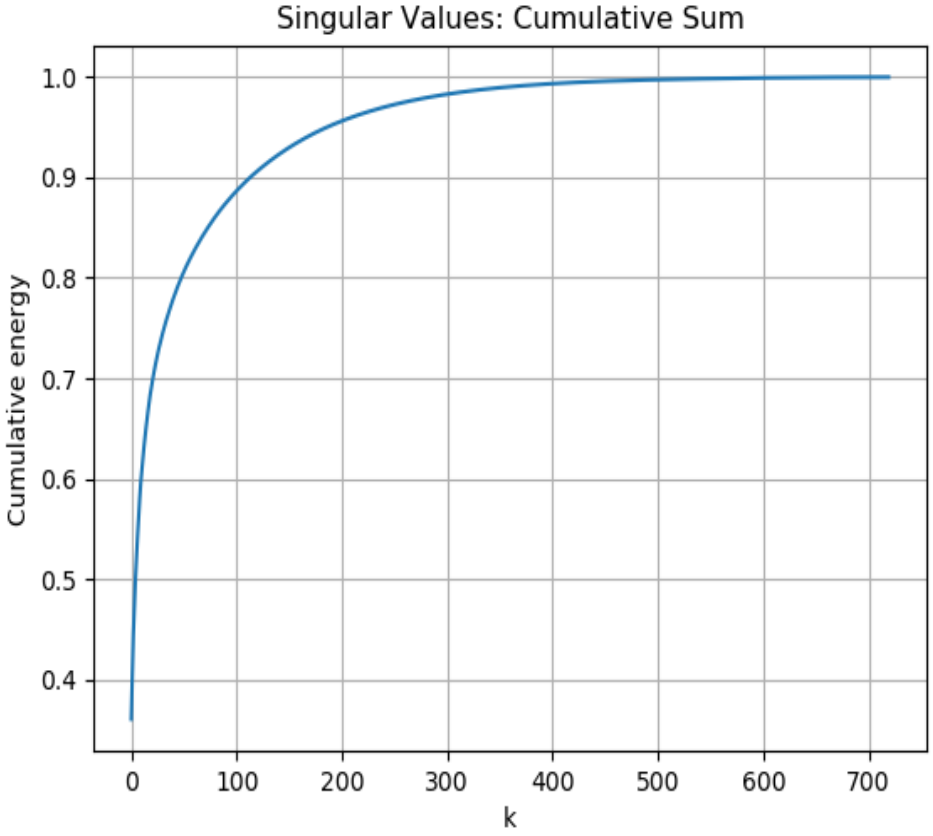


Рис. 3.4 Кумулятивна сума перших k значень сингулярних чисел

збільшуючи попередню проміжну суму на величину нового значення, інша назва терміну - часткова сума. Вертикальна вісь репрезентує відсоткове відношення суми всіх сингулярних значень до суми r перших. Отже, з графіку можна зауважити, що 100 перших сингулярних значень містять майже 90% від суми всіх значень матриці.

Крок 4. Однак, для стиснення нам цього не достатньо, потрібно ще зробити усічення матриць за формулою (4). Для усічення ми будемо використовувати різні значення r , тобто перші r сингулярних значень, стовпців матриці U та рядків матриці V будуть залишені , а інші відкинуті. Після цього, перемноживши наші матриці, ми отримаємо наближену матрицю зображення.

Крок 5. Після усічення матриць, реконструюємо їх назад, для цього матрицю тонів перетворимо в зображення. На рисунку 3.5 показані зображення, які були реконструйовані з наближеної матриці $\tilde{X}$ для різних значень усічення r. При $r = 5$ навіть неможливо сказати, що зображено на фото, а при $r = 100$ відтворене зображення майже не відрізнити від оригінального, оскільки з рисунку 3.4 видно, що перші 100 сингулярних значення складають майже 90% від суми всіх сингулярних значень, отже можна відтворити 90% зображення. Ці маніпуляції вагомо впливають на зменшення якості та розміру файлу, бо ми отримуємо меншу палітру тонів, яка буде зберігатись.

Так ми можемо зробити висновок, що останні сингулярні значення діагональної матриці містять незначну кількість інформації. Цю властивісь можна вдало використати для стиснення зображень, які представлені матрицею. А усунення таких сингулярних значень сприяє зменшенню розміру зображення, уникаючи помітних спотворень початкового.

Рис. 3.5 Стиснення зображення квітки з різним значенням r

3.2 Метод головних компонент

В аналізі даних, як і в будь-якому іншому аналізі, часом буває зручніше створити спрощену модель, яка максимально точно описує реальну. Спрощена модель створюється за допомогою ортогонального проеціювання перших двох головних компонент на площину, або трьох, щоб представити дані в трьох

вимірному просторі, адже набагато легше аналізувати дані, які представлені наочно.

Для реалізації та дослідження методу головних компонент нам потрібно:

1. Центрувати дані
2. Застосувати SVD-розкладання до матриці даних
3. Обрати осі головних компонент та розрахувати головні компоненти
4. Проаналізувати результати

Розглянемо дата-сет, у якому записані дані, які були зібрані за допомогою білкового мікрочіпа H4 та мас-спектрометра (прилад для дослідження та ідентифікації речовини, який дозволяє визначати концентрацію різних компонентів у ній) у людей із раком та без нього. Метою цього техніко-економічного

	0	1	2	3	4	5	6	7	8	9	...	3990	3991
0	0.063915	0.033242	0.018484	0.008618	0.035629	0.037925	0.028865	0.061731	0.063100	0.024787	...	0.035119	0.021515
1	0.025409	0.051085	0.056305	0.021738	0.027410	0.014914	0.022455	0.023957	0.060527	0.047382	...	0.050841	0.055033
2	0.025536	0.036123	0.054195	0.009735	0.027521	0.052255	0.042812	0.069087	0.069873	0.066629	...	0.029078	0.033783
3	0.012817	0.029652	0.079290	0.050677	0.039737	0.057713	0.044492	0.034581	0.042587	0.034147	...	0.054675	0.036083
4	0.019846	-0.010577	-0.007504	0.019042	0.068786	0.061764	0.039036	0.020445	0.025988	0.066716	...	0.063163	0.032044
...	...	...	...	...	...	...	...	...	...	...	...	...	...
211	0.019997	0.002927	0.006809	-0.003585	0.026362	0.026540	0.026112	0.026230	0.021676	0.024205	...	0.016421	0.018509
212	0.042346	0.031884	0.049617	0.031419	0.042043	0.033383	0.054695	0.079029	0.063147	0.040817	...	0.028273	0.019066
213	0.023558	0.021331	0.016210	0.012324	0.022074	0.029829	0.032624	0.022100	0.028950	0.037769	...	0.048846	0.031909
214	0.028351	0.023266	0.004556	0.024095	0.018943	0.025935	0.019066	0.037213	0.041892	0.031092	...	0.017150	0.012613
215	0.027428	0.027021	0.015273	0.026199	0.018178	0.022988	0.021955	0.044174	0.034616	0.030664	...	0.023298	0.040424

216 rows × 4000 columns

	0
0	Cancer
1	Cancer
2	Cancer
3	Cancer
4	Cancer
...	...
212	Normal
213	Normal
214	Normal
215	Normal
216	

217 rows × 1 columns

Рис.3.6 Записи та розміри таблиць

дослідження було дослідити наявність інформації з низькою молекулярною вагою, яка могла б послужити діагностичним класифікатором наявності раку в людини, використовуючи визначений набір досліджень як полігон. Цей набір складається з двох таблиць, одної про результати дослідження для 216 пацієнтів, а друга має два ряди з 216 значеннями (рис. 3.6), кожен з яких або “Cancer”, що вказує на хворого на рак, або “Normal” для здорового пацієнта. Загалом 121 людей яких мають рак, а 95 з них – ні. Кожен рядок з першої таблиці представляє рівень інтенсивності іонів за одним зі 4000 конкретних значень масового заряду для кожного пацієнта. Очевидно, що для візуалізації та дослідження цього дата-сету раціональним буде застосувати зменшення розмірності даних.

Крок 1. Для початку потрібно центрувати наш дата-сет, для цього потрібно розрахувати середнє значення кожного стовпця даних, а потім відняти від кожного його середнє, в результаті ми отримуємо стовпці в яких середнє значення буде дорівнювати нулеві.

Крок 2. Виконаємо розкладання нашого набору даних, який буде записаний у вигляді матриці X за формулою (1). Після розкладу розглянемо нашу матрицю сингулярних значень Σ (рис. 3.7). Можна побачити, що перші сингулярні значення доволі великі, але для більш детального аналізу побудуємо графіки. Перший графік (рис. 3.8) показує величину кожного k сингулярного значення в нашій матриці Σ, можна зауважити, що перші сингулярні значення достатньо великі, але нам потрібно переконатись, що вони складають основну частину діагональної матриці. Для цього побудуємо графік кумулятивної суми (сума послідовності чисел, яка оновлюється кожного разу, коли нове число додається до послідовності, збільшуючи попередню проміжну суму на величину нового значення. Інша назва терміну - часткова сума), що зображений на рисунку 3.9. Вертикальна вісь репрезентує відсоткове відношення суми всіх сингулярних значень до суми k перших. Можна побачити, що перше сингулярне значення складає 55% матриці, отже буде доволі точно показувати варіацію, як перша головна компонента, але ми

будемо використовувати три сингулярних значення, щоб представити данні у трьох вимірах.

	0
0	829.279895
1	103.024891
2	49.376864
3	39.247453
4	31.960807
...	...
211	0.847615
212	0.838476
213	0.834966
214	0.795004
215	0.724811

216 rows × 1 columns

Рис. 3.7 Сингулярні значення

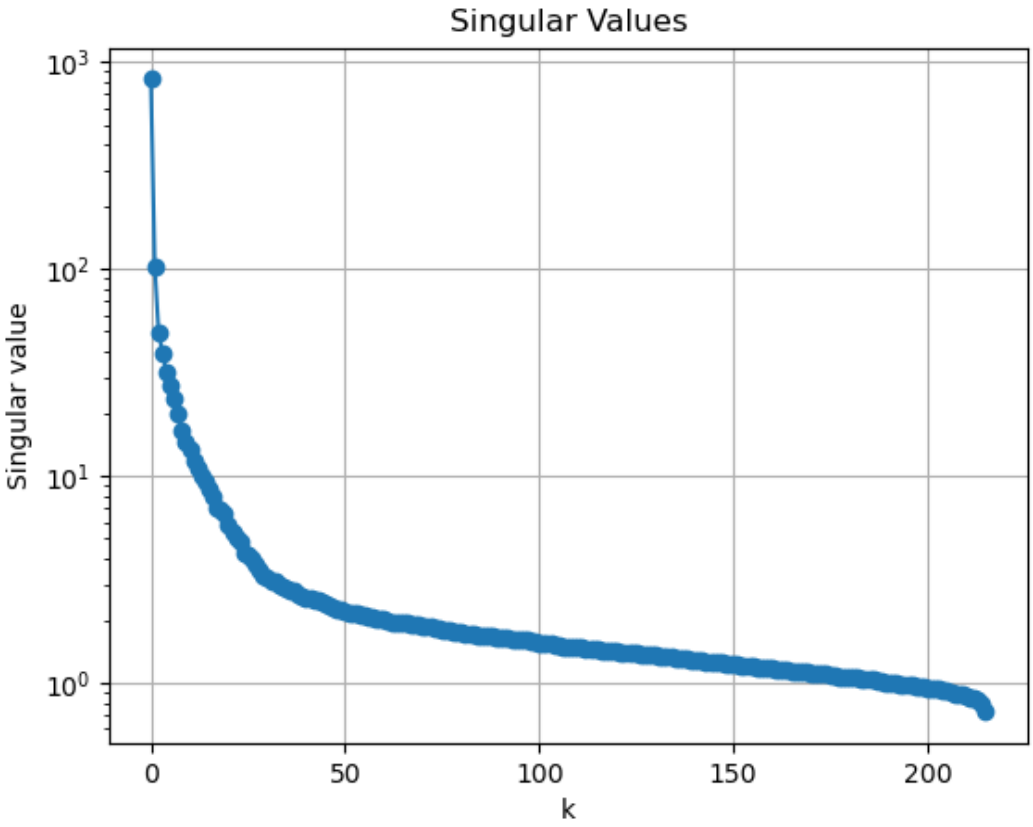


Рис. 3.8 Сингулярні значення σ_k

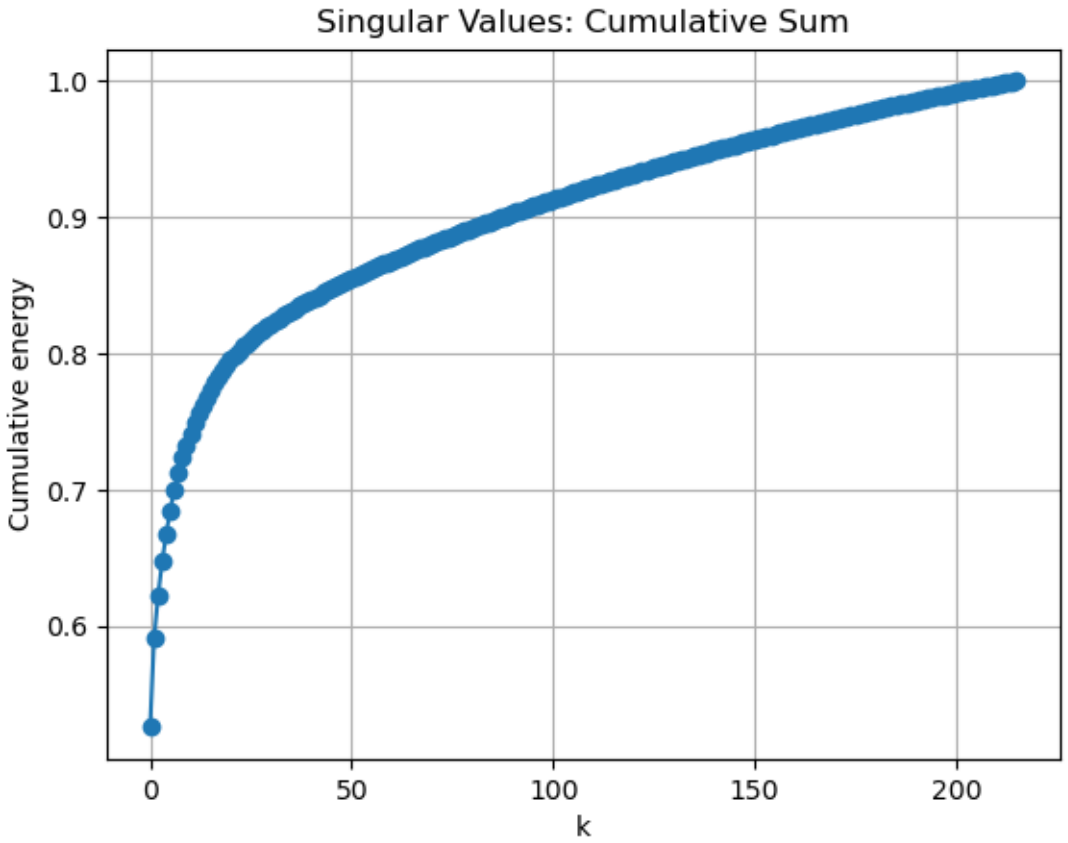


Рис. 3.9 Кумулятивна сума перших k значень сингулярних чисел

Крок 3. Отже, щоб обрати осі головних компонент та головні компоненти ми розрахуємо їх за формулою (11). Ми будемо будувати трьох вимірний графік, тому візьмемо перші три рядки матриці V^T, вектори – v_1, v_2, v_3 і помножимо їх на кожний рядок матриці X. На виході ми отримаємо три значення, які ми візуалізуємо на графіку головних компонент.

Крок 4. Діаграма показує, що червоними хрестиками позначені пацієнти з раком, а синіми кругами – здорові (рис. 3.10). Спостерігається доволі чітке розділення на групи, або кластери. Можна зробити висновок, що взявши більше головних компонент, можна буде чітко сказати, до якого класу відносити пацієнта, але це неможливо буде візуалізувати.

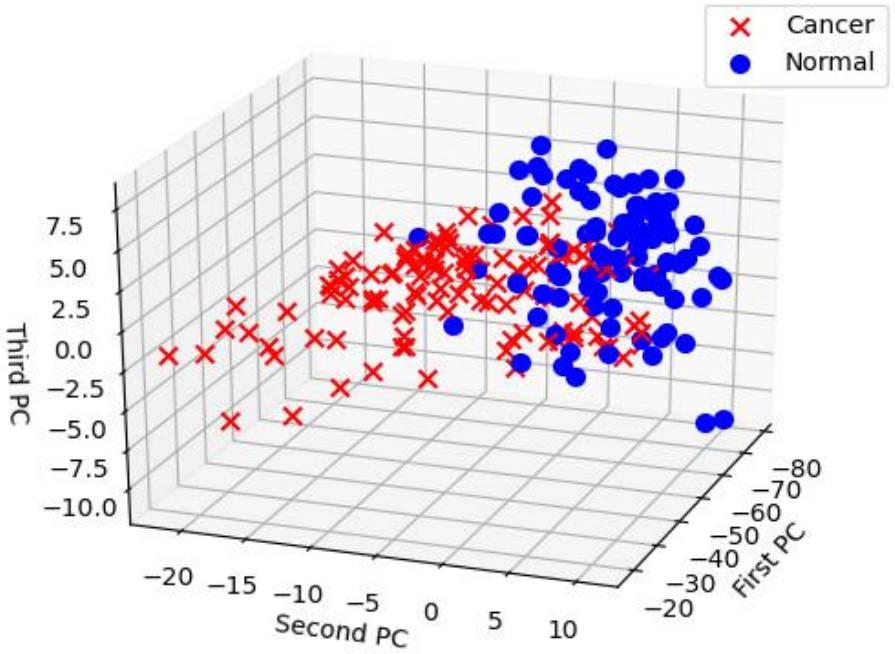


Рис. 3.10 Трьох вимірна діаграма головних компонент

3.3 Множинна лінійна регресія

Множинна лінійна регресія - це як звичайна лінійна регресія, але з більш ніж одним незалежним значенням, тобто ми намагаємось передбачити значення на основі двох або більше змінних.

У цьому прикладі застосування сингулярного розкладання ми будемо аналізувати набір даних про ринок житла в Бостоні, та спробуємо спрогнозувати ціну на майно за допомогою множинної лінійної регресії. Цей дата-сет доступний в бібліотеці *sklearn* в Python.

Для реалізації та дослідження множинної лінійної регресії нам потрібно:

1. Зробити SVD-розкладання
2. Обчислити коефіцієнти регресії
3. Проаналізувати результати

Крок 1. Дата-сет складається з 12 атрибутів, які корелюють із ціною на будинок, наприклад рівень злочинності на душу населення та податок на майно. Виконаємо його сингулярне розкладання по формулі (1).

Крок 2. Після розкладу обчислимо коефіцієнти лінійної регресії за допомогою псевдооберненої матриці за формулою (8). Вхідною матрицею A в нас буде матриця розміром 506×12 елементів, де 506 – це кількість районів, а 12 – це атрибути нашого набору даних. Вектором b буде слугувати середня ціна будинків в кожному районі.

Крок 3. Для прогнозування ми поділили дата-сет на 400 тренувальних та 106 тестових елементів. Множина атрибутів регресує на дані про ціни, а прогноз найкращої ціни відповідає графіку справжньої ціни будинку (рис. 3.11), але цю

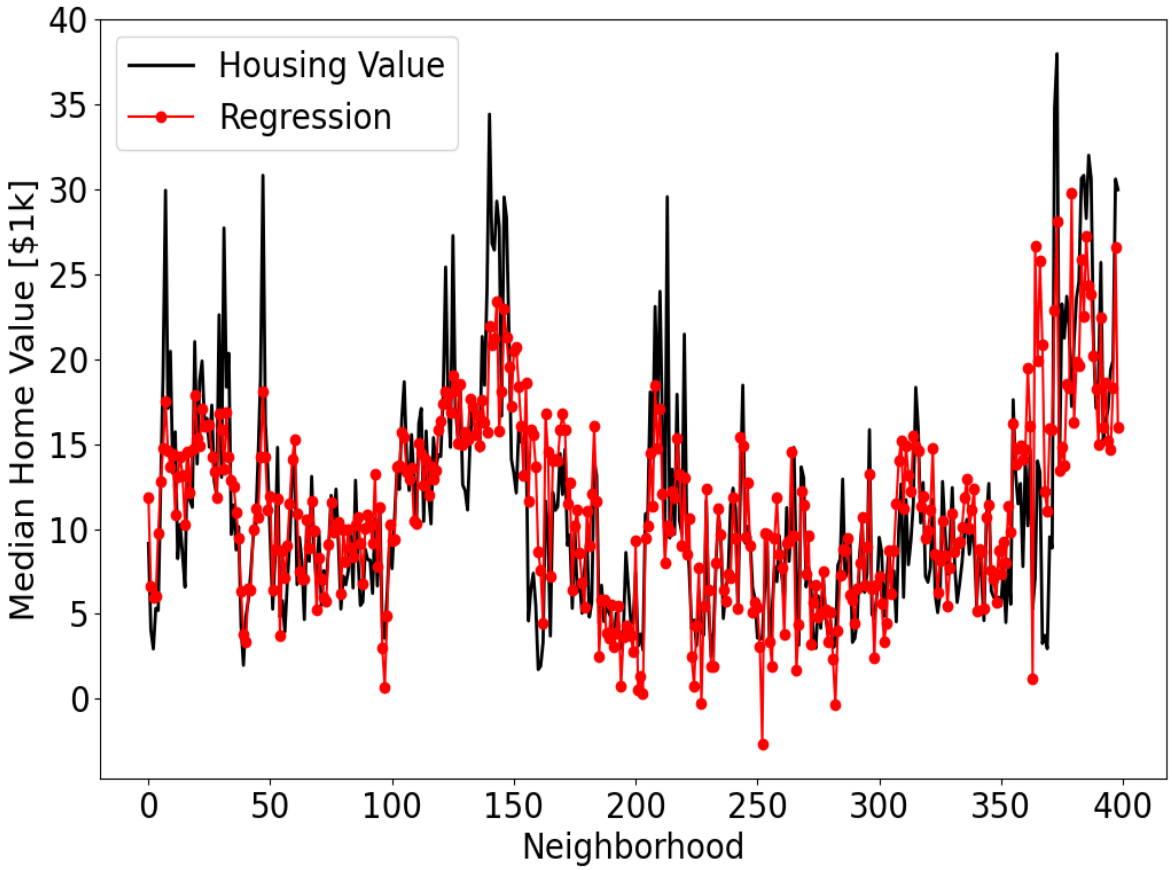


Рис. 3.11 Графік результатів тренування

модель ми отримали під час тренування. На графіку вісь ординат відповідає за середню ціну будинку (одна поділка є 1000$), а вісь абсцис відповідає за район. Часто буває так, що викиди з найвищою цінністю недостатньо добре фіксуються, як спостерігається в цьому прикладі.

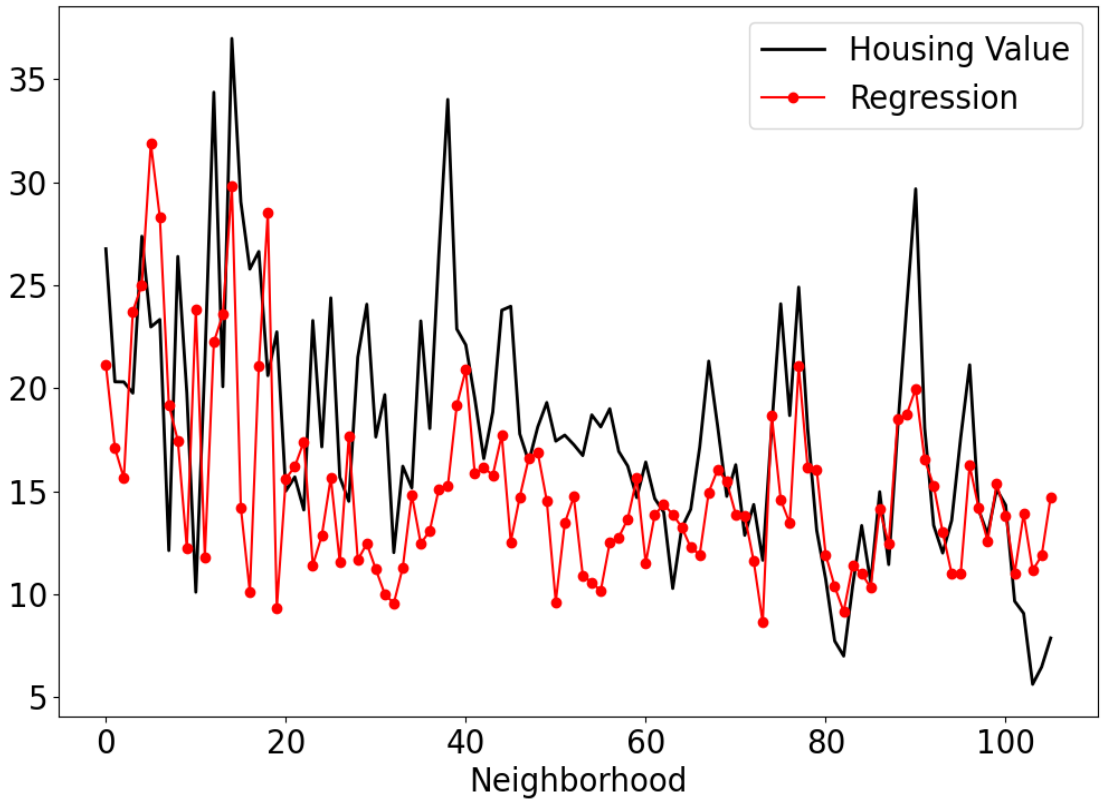


Рис. 3.12 Графік результатів тестування

На рисунку 3.12 зображено графік результатів тестування. У деяких частинах графіку передбачення показало досить хороші результати, але в більшості випадків передбачення невдале. Звідси можна зробити висновок, що модель, яку ми розробили під час тренування, не зовсім підходить для прогнозування. Можливо, це зумовлено тим, що данні з дата-сету збирались поступово, тобто окремо спочатку з одного району, а потім з іншого. Тому, коли ми спробували побудувати тестову модель, то дані тестової вибірки містять інформацію про райони, які відрізняються від районів тренувального дата-сету.

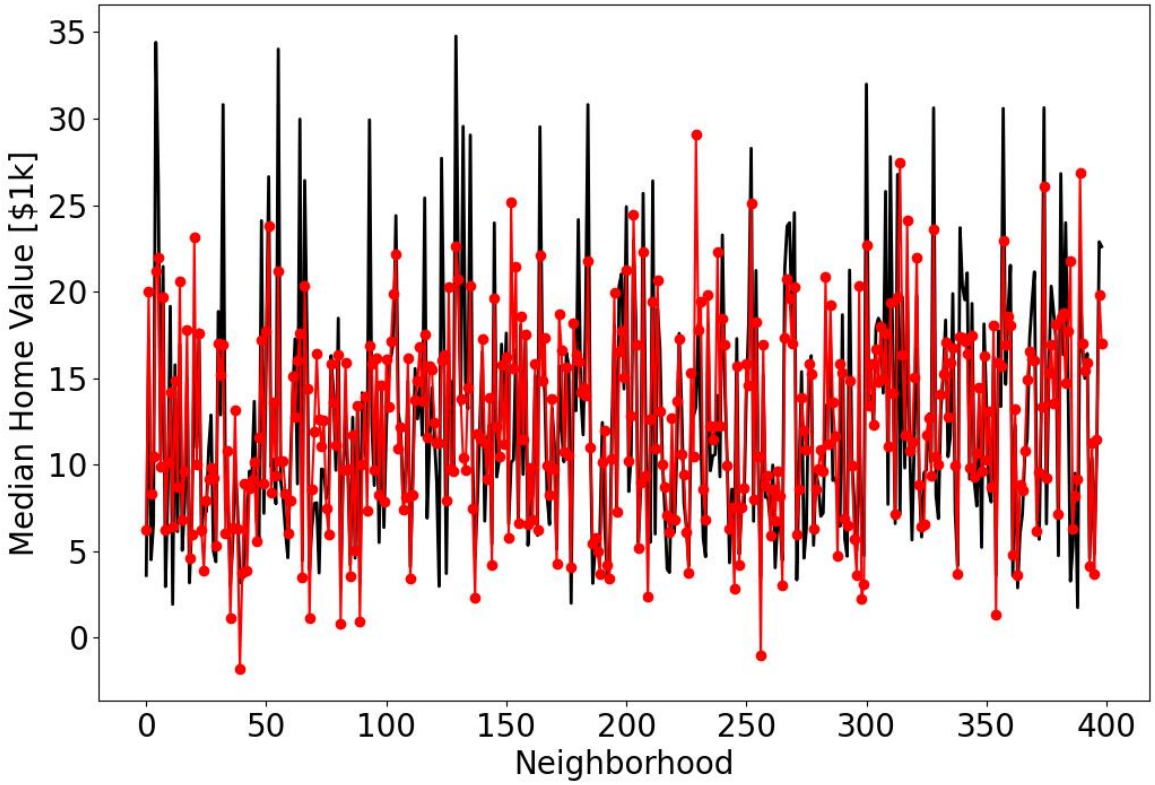


Рис. 3.13 Графік результатів тренування після перестановки даних

Для вирішення цієї проблеми спробуємо зробити випадкову перестановку наших даних. Розподіл наборів на тестові та тренувальні залишимо без змін.

Загалом тренування після перестановки даних (рис. 3.13) не показало відмінних результатів від попереднього тренування, так само викиди з добре фіксуються, але в цілому спостерігаються хороші результати. А ось тестування (рис. 3.14) навпаки показало кращі результати, ніж тестування без перестановки даних. Навіть деякі викиди доволі точно спрогнозовані.

Отже, можна зробити висновок, що даний дата-сет складався поступово, районом за районом, тож краще, для більш точного аналізу, зробити перестановку його елементів.

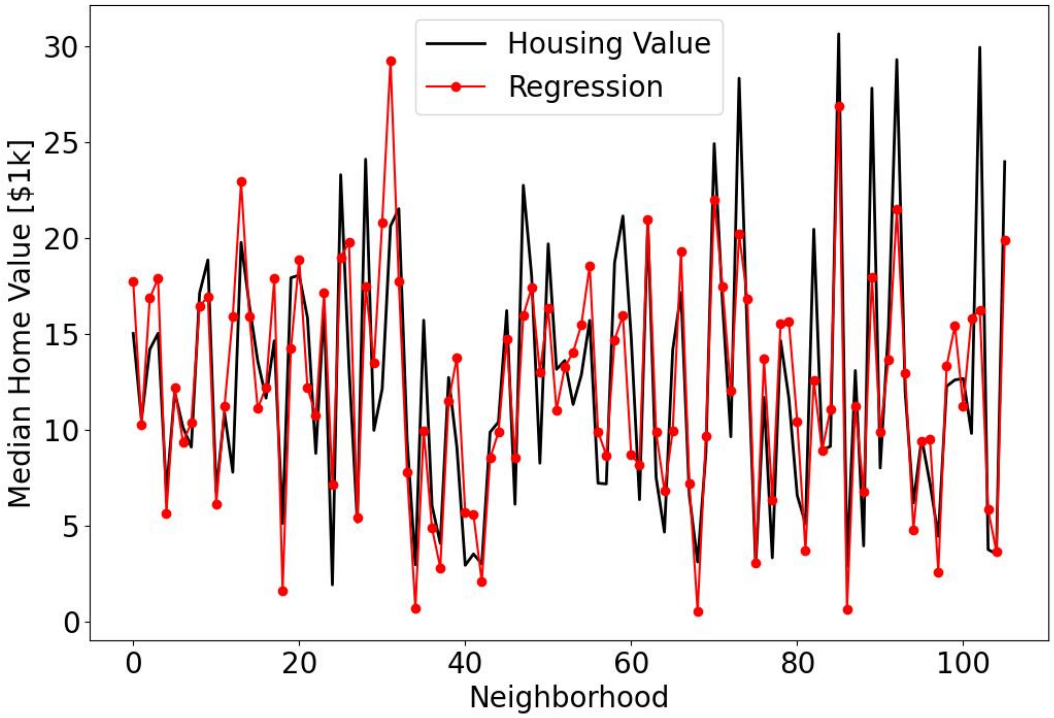


Рис. 3.14 Графік результатів тестування після перестановки даних

ВИСНОВОКИ

Отже, сингулярне розкладання – це не лише класична теорія обчислення та аналізу матриць, але також є потужним інструментом у машинному навчанні та сучасному аналізі даних. У даному курсовому проекті ми спочатку вивчаємо основне поняття SVD, а потім показуємо центральну роль SVD в розкладанні матриць. Використовуючи матричне наближення, ми розглядаємо роль сингулярних значень та власних векторів у матричних перетвореннях. Обговорюємо унітарно інваріантні норми, які потім використовуються для формулювання загальних результатів для оптимального рангового наближення матриці.

В курсовій роботі були програмно реалізовані методи застосування сингулярного розкладання, а саме: у стисненні зображення, методі головних компонент та регресійному аналізі. Завдяки візуалізації досліджень та проведення аналізу довели доречність використання методу SVD-розкладання матриці.

Перспективним напрямком продовження дослідження є застосування SVD-розкладання в рандомізованій числовій лінійній алгебрі для збільшення ефективності розкладання матриць великих даних, що є одним з основних досліджень обчислювальної математики та науки про дані. Оскільки так звані рандомізовані чисельні методи мають потенціал змінити обчислювальну лінійну алгебру, забезпечуючи точне матричне розкладання при менших затратах детермінованих методів. Більше того, із усе більш масштабними вимірюваннями, обчислювальна ефективність рандомізованих методів стане важливою в найближчі роки та десятиліття зі зростанням кількості даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гантмахер Ф. Р. Теория матриц Москва: Наука, 1966. 576с.
2. Andrews, H. Singular value decomposition (SVD) image coding / H. Andrews, C. Patterson // IEEE transactions on Communications. – 1976. – Vol. 24, no. 4. – P. 425–432 c.
3. C. Eckart, G. Young. The approximation of one matrix by another of lower rank. Psychometrika, 1(3):211–218, 1936.
4. E. Schmidt. Zur theorie der linearen und nichtlinearen integralgleichungen. i teil. entwicklung willkurlichen funktionen nach system vorgeschriebener. Mathematische Annalen. Німеччина, Лейпциг : Deutschland, Leipzig 433– 476c., 1907
5. Gene H. Holub, Charles F. Van Loan. Matrix Computations // JHU Press, 2013. 756 c.
6. Gerbrands, J.J. On the relationships between SVD, KLT and PCA / J.J. Gerbrands // Pattern Recognition. – 1981. – Vol. 14, no. 1–6. – P. 375–381 c.
7. Hotelling H. Analysis of a complex of statistical variables into principal components : наукове видання. 1933. 24с.
8. Jha, S.K. Denoising by singular value decomposition and its application to electronic nose data processing / S.K. Jha, R.D.S. Yadava // IEEE Sensors Journal. – 2011. – Vol. 11, no. 1. – P. 35–44 c.
9. Karl Pearson. On lines and planes of closest fit to systems of points in space: наукове видання. Лондон, Великобританія, 1901, 559–572 с.
10. Liu, R. An SVD-based watermarking scheme for protecting rightful ownership / R. Liu, T. Tan // IEEE transactions on multimedia. – 2002. – Vol. 4, no. 1. – P. 121–128 c.
11. Moonen, M. SVD and signal processing: algorithms, applications and architectures / M. Moonen, B. De Moor. – Elsevier Science, 1995. – 485 c.
12. No-reference image blur index based on singular value curve / Q. Sang [et al.] //

Journal of Visual Communication and Image Representation. – 2014. – Vol. 25, no. 7. – P. 1625–1630 c.

13. Singular value decomposition based sample diversity and adaptive weighted fusion for face recognition / G. Zhang [et al.] // Digital Signal Processing. – 2017. – Vol. 62, no. 3. – P. 150–156 c.
14. Targhi, A.T. Clustering of singular value decomposition of image data with applications to texture classification / A.T. Targhi, A. Shademan // Proc. of Intern. Conf. on Visual Communications and Image Processing. – Lugano, Switzerland, 2003. – Vol. 5150. – P. 972–979 c.
15. Vinita Vasudevan, M.Ramakrishna. A Hierarchical Singular Value Decomposition Algorithm for Low Rank Matrices : наук. доповідь. Ченай, Індія : Indian Institute of Technology-Madras, 2019
16. William H. Press, Saul A.Teukolsky, William T. Vetterling, Brian P. Flannery. Numerical Recepies in C: The Art of Scientific Computing. Великобританія, Кембридж : Cambridge University Press, 2007. 1262c.
17. Zhang, D. A new face recognition method based on svd perturbation for single example image per person / D. Zhang, S. Chen, Z.-H. Zhou // Applied Mathematics and Computation. – 2005. – Vol. 163, no. 2. – P. 95–907 c.
18. Псевдообернена матриця // Вікіпедія : веб-сайт. URL : https://uk.wikipedia.org/wiki/%D0%9F%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D0%BE%D0%B1%D0%B5%D1%80%D0%BD%D0%B5%D0%BD%D0%B0_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8F
19. Eigendecomposition of a matrix // Wikipedia : веб-сайт. URL: https://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix
20. Eigenvalues and eigenvectors // Wikipedia : веб-сайт. URL: https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors
21. Linear Regression // Wikipedia : веб-сайт. URL: https://en.wikipedia.org/wiki/Linear_regression

22. Matrix Decomposition // Wikipedia : веб-сайт. URL: https://en.wikipedia.org/wiki/Matrix_decomposition (дата звернення: 04.04.2020)
23. Schur decomposition // Wikipedia : веб-сайт. URL: https://en.wikipedia.org/wiki/Schur_decomposition
24. Singular Value // Wikipedia : веб-сайт. URL: https://en.wikipedia.org/wiki/Singular_value (дата звернення: 04.04.2020)
25. Numpy svd // Numpy : веб-сайт. URL: https://numpy.org/doc/stable/reference/generated/numpy.linalg.svd.html

ДОДАТКИ
ДОДАТОК А

```
from matplotlib.image import imread
import matplotlib.pyplot as plt
import numpy as np

A = imread('flower.jpg')
X = np.mean(A, -1) # Convert RGB to grayscale

img = plt.imshow(X)
img.set_cmap('gray')
plt.axis('off')
plt.title('Original image')
#plt.savefig('output/original_image.png')
plt.show()

U, S, VT = np.linalg.svd(X,full_matrices=False)
S = np.diag(S)

j = 0
rank = [5, 20, 100]
for r in (5, 20, 100):
    # Construct approximate image
    Xapprox = U[:,:r] @ S[0:r,:r] @ VT[:r,:]
    plt.figure(j+1)
    j += 1
    img = plt.imshow(Xapprox)
    img.set_cmap('gray')
    plt.axis('off')
    plt.title('r = ' + str(r))
    #plt.savefig('output/singular_values{0}.png'.format(r))
    plt.show()

plt.figure(1)
plt.semilogy(np.diag(S))
plt.title('Singular Values')
plt.ylabel('Singular value')
plt.xlabel('k')
plt.grid()
#plt.savefig('output/singular_values_plot.png')
plt.show()

plt.figure(2)
plt.plot(np.cumsum(np.diag(S))/np.sum(np.diag(S)))
plt.title('Singular Values: Cumulative Sum')
plt.ylabel('Cumulative energy')
plt.xlabel('k')
plt.grid()
#plt.savefig('output/cumulative_sum_plot.png')
plt.show()
```

ДОДАТОК Б

```
import matplotlib.pyplot as plt
import numpy as np
import os
from mpl_toolkits.mplot3d import Axes3D

obs = np.loadtxt(os.path.join('ovariancancer_obs.csv'), delimiter=',')

f = open(os.path.join('ovariancancer_grp.csv'), "r")
grp = f.read().split("\n")
print(obs)
U, S, VT = np.linalg.svd(obs, full_matrices=0)
print(U)
print(S)
print(VT)

plt.semilogy(S, '-o', color='k')
plt.grid(True)
plt.title('Singular Values')
plt.ylabel('Singular value')
plt.xlabel('k')
plt.savefig('output/pca1.png')
plt.show()
plt.plot(np.cumsum(S) / np.sum(S), '-o', color='k')
plt.grid(True)
plt.title('Singular Values: Cumulative Sum')
plt.ylabel('Cumulative energy')
plt.xlabel('k')
plt.savefig('output/pca2.png')
plt.show()

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

for j in range(obs.shape[0]):
    x = VT[0, :] @ obs[j, :].T
    y = VT[1, :] @ obs[j, :].T
    z = VT[2, :] @ obs[j, :].T

    if grp[j] == 'Cancer':
        cancer = ax.scatter(x, y, z, marker='x', color='r', s=50, label="Cancer")
    else:
        normal = ax.scatter(x, y, z, marker='o', color='b', s=50, label="Normal")

plt.legend(handles=[cancer, normal])
ax.view_init(25, 50)
ax.set_xlabel('First PC')
ax.set_ylabel('Second PC')
ax.set_zlabel('Third PC')

plt.savefig('output/3dpca.png')
plt.show()
```

ДОДАТОК В

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.datasets import load_boston
plt.rcParams['figure.figsize'] = [12, 8]
plt.rcParams.update({'font.size': 20})

# Load dataset
H, y = load_boston(return_X_y=True)
b = H[:,-1] # housing values in $1000s
A = H[:,:-1] # other factors

# Pad with ones for nonzero offset
A = np.pad(A,[(0,0),(0,1)],mode='constant',constant_values=1)

n = 400
btrain = b[1:n]
Atrain = A[1:n]
btest = b[n:]
Atest = A[n:]

# Solve Ax=b using SVD
U, S, VT = np.linalg.svd(Atrain,full_matrices=0)
x = VT.T @ np.linalg.inv(np.diag(S)) @ U.T @ btrain


plt.plot(btrain, Color='k', LineWidth=2, label='Housing Value') # True
relationship
plt.plot(Atrain@x, '-o', Color='r', LineWidth=1.5, MarkerSize=6,
label='Regression')
plt.xlabel('Neighborhood')
plt.ylabel('Median Home Value [$1k]')
plt.legend()
plt.savefig('output/regressionris.png')
plt.show()

# sort_ind = np.argsort(H[:,n])
# btest = btest[sort_ind] # sorted values
plt.plot(btest, Color='k', LineWidth=2, label='Housing Value') # True relationship
plt.plot(Atest@x, '-o', Color='r', LineWidth=1.5, MarkerSize=6,
label='Regression')
plt.xlabel('Neighborhood')
plt.legend()
plt.savefig('output/regression2.png')
plt.show()
```

ДОДАТОК Г

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.datasets import load_boston
plt.rcParams['figure.figsize'] = [12, 8]
plt.rcParams.update({'font.size': 20})

# Load dataset
H, y = load_boston(return_X_y=True)
b = H[:,-1] # housing values in $1000s
A = H[:,:-1] # other factors

# Pad with ones for nonzero offset
A = np.pad(A,[(0,0),(0,1)],mode='constant',constant_values=1)

n = 400
p = np.random.permutation(506)
A = A[p, :]
b = b[p]
btrain = b[1:n]
Atrain = A[1:n]
btest = b[n:]
Atest = A[n:]

# Solve Ax=b using SVD
U, S, VT = np.linalg.svd(Atrain,full_matrices=0)
x = VT.T @ np.linalg.inv(np.diag(S)) @ U.T @ btrain


plt.plot(btrain, Color='k', LineWidth=2, label='Housing Value') # True
relationship
plt.plot(Atrain@x, '-o', Color='r', LineWidth=1.5, MarkerSize=6,
label='Regression')
plt.xlabel('Neighborhood')
plt.ylabel('Median Home Value [$1k]')
plt.savefig('output/regression_train_shuffle.png')
plt.legend()
plt.show()

# sort_ind = np.argsort(H[:,n])
# btest = btest[sort_ind] # sorted values
plt.plot(btest, Color='k', LineWidth=2, label='Housing Value') # True relationship
plt.plot(Atest@x, '-o', Color='r', LineWidth=1.5, MarkerSize=6,
label='Regression')
plt.xlabel('Neighborhood')
plt.ylabel('Median Home Value [$1k]')
plt.legend()
plt.savefig('output/regression_test_shuffle.png')
plt.show()
```