Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра математики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: **«ГАУСОВІ СУМІШІ ТА МЕТОД МАКСИМІЗАЦІЇ МАТСПОДІВАННЯ»**

Виконала: студентка 4-го року навчання
Спеціальності
113 Прикладна математика

Єфременко Анастасія Олексіївна

Керівник: Крюкова Г.В.
доцент, кандидат фіз.-мат. наук

Рецензент: _____
(прізвище та ініціали)

Кваліфікаційна робота захищена
з оцінкою _____

Секретар ЕК_____
(підпис)

«_____» _____ 20___р.

Київ - 2025

Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра математики

ЗАТВЕРДЖУЮ
Зав.кафедри математики,
доцент, кандидат фіз.-мат. наук
_____ *Чорней Р.К.*
(підпис)
“____”_____ 2025

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на кваліфікаційну роботу
студентці 4-го курсу факультету інформатики
Єфременко Анастасії Олексіївні

Тема: «Гаусові суміші та метод максимізації матсподівання»

Зміст кваліфікаційної роботи:

Анотація

1. Вступ

2. Огляд основних означень та тверджень, що пов ′язані з Гаусовими сумішами

3. Реалізація алгоритму EM (Expectation-Maximization) для оцінки параметрів Гаусової суміші

4. Експерименти з даними. Порівняння результатів класифікації за допомогою Гаусових сумішей з іншими методами

Висновки

Список літератури

Дата видачі “____”_____ 2025 Керівник _____
(підпис)

Завдання отримав _____
(підпис)

Графік підготовки кваліфікаційної роботи до захисту

Графік узгоджено «_____» _____ 2024р.

№ з/п	Перелік робіт	Термін виконання етапу	Підпис наукового керівника	Дата ознайомлення наукового керівника	Примітка
1.	Отримання теми кваліфікаційної роботи.	06.10.2024			
2.	Ознайомлення з темою кваліфікаційної роботи.	07.10.2024			
3.	Розробка плану та структури роботи.	13.10.2024			
4.	Робота з науковою літературою, опис основних означень. Написання вступу та анотації.	28.10.2024			
5.	Дослідження Гаусових сумішів.	04.11.2024			
6.	Робота над текстовим оформленням теоретичної частини та одержаних результатів.	25.02.2025			
7.	Попередній аналіз кваліфікаційної роботи. Виправлення помилок.	04.04.2025			
8.	Попередній захист кваліфікаційної роботи.	23.05.2025			
9.	Захист кваліфікаційної роботи.	05.06.2025			

Науковий керівник _____
(ПІБ)

Виконавець кваліфікаційної роботи _____
(ПІБ)

Зміст

Анотація

Завдання полягає в розробці ефективного методу кластеризації, який дозволяє ідентифікувати ймовірнісні зв'язки в даних за допомогою моделі суміші Гауса (GMM). Метод передбачає використання параметричного підходу для моделювання кластерів у вигляді гаусіан із заданими середніми, коваріаційними матрицями та ваговими коефіцієнтами. GMM забезпечує м'яке кластеризування, дозволяючи оцінювати належність кожної точки до кількох кластерів одночасно. Це дає змогу аналізувати складні структури даних і виявляти приховані патерни з високою точністю.

1 Вступ

1.1 Актуальність

Гаусові суміші (Gaussian Mixture Models, GMM) та методи максимізації математичного сподівання (Expectation-Maximization, EM) є надзвичайно актуальними в сучасних дослідженнях і застосуваннях через їх універсальність, ефективність і широке застосування у багатьох галузях науки, техніки та економіки.

Гаусові суміші є потужним інструментом для моделювання складних розподілів даних. Завдяки своїй здатності до моделювання нелінійних залежностей, GMM широко використовуються у задачах кластеризації, оцінки щільності розподілу та візуалізації даних.

Методи максимізації математичного сподівання є стандартним підходом до розв′язання задач оптимізації, де дані мають приховані змінні або спостереження є неповними. Алгоритм EM ефективно застосовується у задачах кластеризації, наприклад, для ідентифікації прихованих структур у великих наборах даних, що актуально для сучасних технологій обробки інформації, зокрема:

- Обробка зображень та сигналів.
- Розпізнавання образів і мови.

Також Гаусові суміші використовуються в задачах автоматизації, наприклад, у розробці систем для автономних транспортних засобів, де кластеризація та прогнозування є ключовими компонентами. У медицині GMM допомагають аналізувати медичні зображення для діагностики захворювань. У сфері фінансів вони дозволяють моделювати розподіли ризиків та оцінювати складні інвестиційні портфелі.

Поєднання гаусових сумішей та EM-алгоритму забезпечує міцну математичну основу для вирішення задач, де необхідно працювати із ймовірнісними моделями. Це стимулює подальший розвиток методів нелінійної оптимізації, стохастичних алгоритмів та глибокого навчання.

1.2 Мета, завдання дослідження

Розробка, аналіз і застосування математичних методів та алгоритмів, заснованих на гаусових сумішах і методі максимізації математичного сподівання (ЕМ-алгоритмі), для вирішення прикладних задач кластеризації, оцінки параметрів ймовірнісних моделей та обробки даних.

- Дослідити математичні основи гаусових сумішей, їх властивості та можливості застосування у моделюванні даних.
- Вивчити метод максимізації математичного сподівання, його математичне обґрунтування та умови ефективності.
- Розробити алгоритми для навчання моделей гаусових сумішей із застосуванням ЕМ-методу.
- Спробувати ЕМ-алгоритм у задачах оцінки параметрів для багатомодальних розподілів, що часто зустрічаються у реальних даних.
- Провести порівняльний аналіз запропонованих методів з іншими підходами кластеризації та оцінки параметрів.
- Розглянути приклад, де гаусові суміші та ЕМ-алгоритм можуть продемонструвати свою ефективність.

2 Основні теореми та означення

2.1 Теоретичні основи та приклад використання GMM

Означення 1 . ***Некероване навчання*** *(unsupervised learning) - це підхід у машинному навчанні, де на відміну від навчання з учителем (supervised learning), алгоритми виявляють закономірності виключно на основі немаркованих даних [1].*

Означення 2 . ***Модель суміші Гауса*** *- це техніка м'якого кластерезування, яка визначає ймовірність точки належати кластеру при неконтрольованому навчанні. Ця модель складається з Гаусіан, які позначаються через $k = 1, 2, \ldots, K.$, в наборі даних K - це кількість кластерів. Кожна компонента Гаусіана характеризується такими параметрами для кластеру k:*

1. μ_k - середнє значення, яке визначає центр розподілу.

2. Σ_k - коваріаційна матриця, яка визначає форму та ширину розподілу

(див. [2]).

Означення 3 . ***Математичним сподіванням (середнім значенням)*** *випадкової величини $\xi = f(\omega)$ називають значення інтеграла Лебега:*

$$\mathbb{E}[\xi] = \int f(x)\, P(dx),$$

де P - це імовірнісна міра, яка визначає ймовірність подій на просторі елементарних подій.

Для випадкового вектора $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_n)$, математичне сподівання визначається як вектор:

$$\mathbb{E}[\boldsymbol{\xi}] = (\mathbb{E}[\xi_1], \mathbb{E}[\xi_2], \ldots, \mathbb{E}[\xi_n])\,.$$

Для абсолютно неперервної випадкової величини ξ, яка має функцію густини $f_\xi(x)$, математичне сподівання визначається як:

$$\mathbb{E}[\xi] = \int_{-\infty}^{+\infty} x f_\xi(x)\, dx,$$

за умови збіжності інтеграла:

$$\int_{-\infty}^{+\infty} |x| f_\xi(x)\, dx < \infty.$$

(див. [3]).

Математичне сподівання є ключовою операцією під час E-кроку EM(очікування-максимізації) алгоритму, що використовується для оцінювання параметрів моделі суміші Гауса.

Означення 4 . *Нехай* $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)$ *- випадковий вектор з координатами, що мають скінченну дисперсію. Тоді його* ***коваріаційною матрицею*** *називається квадратна матриця розміру* $n \times n$*, елементи якої визначаються як:*

$$\mathrm{Cov}(\boldsymbol{\xi}) = \left(\mathrm{Cov}(\xi_i, \xi_j)\right)_{i,j=1}^{n},$$

де $\mathrm{Cov}(\xi_i, \xi_j) = \mathbb{E}[(\xi_i - \mathbb{E}[\xi_i])(\xi_j - \mathbb{E}[\xi_j])]$ *- коваріація між змінними* ξ_i *та* ξ_j *(див. [4]).*

Означення 5 . ***Функцією розподілу*** *випадкової величини* ξ *називається функція* $F_\xi(x)$*, яка для кожного дійсного числа* $x \in \mathbb{R}$ *визначається як ймовірність того, що* ξ *набуває значення менше* x*:*

$$F_\xi(x) = \mathbb{P}(\xi < x), \quad x \in \mathbb{R}$$

(див. [4]).

Означення 6 . ***Функція Q*** *- ймовірність того, що нормальна (гауссова) випадкова величина набуде значення, більше ніж* x*.*

$$Q(x) = 1 - \Phi(x), \tag{1}$$

де $\Phi(x)$ *- функція розподілу стандартного нормального закону [5].*

Означення 7 . ***Багатовимірний нормальний розподіл.*** *Нехай* $\vec{\mu} \in \mathbb{R}^m$ *є вектором математичного сподівання, а* $\Sigma \in \mathbb{R}^{m \times m}$ *- симетрична додатно визначена матриця коваріації. Кажуть, що випадковий вектор* $\vec{\xi} \in \mathbb{R}^m$ *має*

багатовимірний нормальний розподіл з параметрами $\vec{\mu}$ та Σ, якщо його щільність ймовірності задається формулою:

$$f_{\vec{\xi}}(\vec{x}) = \frac{1}{(2\pi)^{m/2}|\det \Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\vec{x}-\vec{\mu})^{\top}\Sigma^{-1}(\vec{x}-\vec{\mu})\right\} \tag{2}$$

(див. [6]). де $\det\Sigma$ *- визначник коваріаційної матриці, m - розмірність вектора, а $\vec{x}-\vec{\mu}$ - відхилення від середнього значення.*

Такий розподіл позначається як $\mathcal{N}(\vec{\mu},\Sigma)$ і називається багатовимірним нормальним розподілом з математичним сподіванням $\vec{\mu}$ та коваріацією Σ. Він узагальнює звичайний одномірний нормальний розподіл на випадок векторних величин.

Теорема 1 . *Ймовірність у GMM.* *Ймовірність спостереження x_i в моделі суміші K гаусівських компонент виражається як:*

$$f(x) = \sum_{k=1}^{K} \pi_k \, \varphi(x|\mu_k, \Sigma_k), \tag{3}$$

де $\varphi(x \mid \mu_k, \Sigma_k)$ - багатовимірна нормальна щільність ймовірності для k-ої компоненти, яка описує ймовірність для точки x з параметрами μ_k та Σ_k, а π_k - це ваговий коефіцієнт або пропорція змішування k-го компоненту [2].

Приклад. Суміш двох одномірних нормальних розподілів.

Розглянемо просту модель суміші двох нормальних розподілів:

$$x \sim 0.5 \cdot \mathcal{N}(-2,1) + 0.5 \cdot \mathcal{N}(2,1),$$

що означає: кожне спостереження x з імовірністю 0,5 походить з нормального розподілу $\mathcal{N}(-2,1)$ або з $\mathcal{N}(2,1)$.

На відміну від одного нормального розподілу, така модель дозволяє описувати складніші форми функції густини. У цьому випадку результатом є двомодальна густина, тобто функція з двома вершинами. Вона не є одним гаусівським розподілом, але є прикладом моделі суміші Гауса (GMM) з двома компонентами (див. Рис. 1).

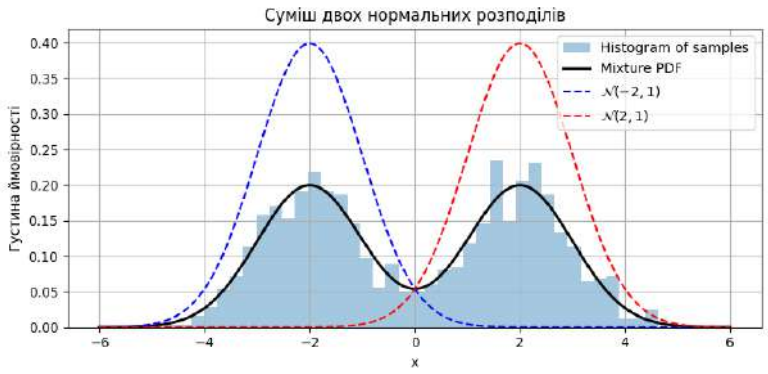

Рис. 1: Суміш двох нормальних розподілів: $\mathcal{N}(-2,1)$ і $\mathcal{N}(2,1)$ з однаковими вагами, використовуючи Python.

2.2 Теоретичні основи EM-алгоритму

Означення 8 . ***Кластеризація*** *- це підхід неконтрольованого машинного навчання, вона автоматично класифікує дані в групи відповідно до подібності критеріїв [1].*

Приклад. У нас є набір блоків різної форми (квадрати, кола, трикутники) і кольору (червоні, сині, зелені). Завдання - розташувати ці блоки на площині за певними правилами. Тоді ми можемо використати такі кластеризації:

- **Варіант 1.** Сортування за кольором Ми вирішуємо розташувати блоки групами одного кольору. Наприклад:
  - У верхньому лівому куті ми розміщуємо всі червоні блоки, незалежно від їх форми.
  - У верхньому правому - сині блоки.
  - У нижній частині - зелені блоки.
- **Варіант 2.** Сортування за формою. Наприклад:

  - У верхній частині ми розміщуємо всі квадрати, незалежно від їх кольору.
  - У нижньому лівому куті - всі кола.
  - У нижньому правому куті - всі трикутники.

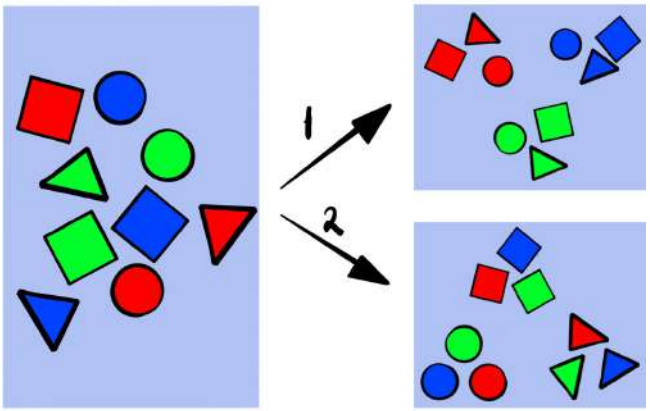

Рис. 2: Цей приклад демонструє, як однаковий набір даних можна кластеризувати за різними критеріями - в залежності від обраної ознаки.

Означення 9 . ***Алгоритм очікування-максимізації (EM)*** *- це алгоритм, що знаходить оцінки параметрів максимальної правдоподібності в задачах, де деякі змінні не спостерігаються (див. означення 12) [2].*

Зауваження 1 . ***Очікування–максимізація (EM) для GMM.***

EM-алгоритм для оцінки параметрів моделі гаусівських сумішей (GMM) складається з двох основних кроків. Його застосування до GMM включає:

- ***E-крок (Expectation step):*** *обчислення ймовірності належності кожного спостереження x_i до k-го кластера на основі поточних параметрів моделі, тобто відповідальності γ_{ik} (детальніше про γ_{ik} див. формулу EM-алгоритму нижче (7)).*
- ***M-крок (Maximization step):*** *оновлення параметрів моделі $\theta = (\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ для максимізації очікуваної лог-правдоподібності:*

$$\pi_k = \frac{1}{N}\sum_{i=1}^{N}\gamma_{ik}, \quad \boldsymbol{\mu}_k = \frac{\sum_{i=1}^{N}\gamma_{ik}\cdot\mathbf{x}_i}{\sum_{i=1}^{N}\gamma_{ik}}, \quad \boldsymbol{\Sigma}_k = \frac{\sum_{i=1}^{N}\gamma_{ik}(\mathbf{x}_i-\boldsymbol{\mu}_k)(\mathbf{x}_i-\boldsymbol{\mu}_k)^\top}{\sum_{i=1}^{N}\gamma_{ik}}$$

де N - загальна кількість спостережень, π_k - апріорні ймовірності (ваги) компонент, $\boldsymbol{\mu}_k$ - середні вектори, а $\boldsymbol{\Sigma}_k$ - коваріаційні матриці k-тої компоненти [2].

Теорема 2 . *EM-алгоритм оцінювання параметрів $\theta = (\pi_k, \mu_k, \Sigma_k)$ визна-*

чається як розв'язок рівняння:

$$Q(\theta|\theta^{(t)}) = \sum_{i=1}^{n}\sum_{k=1}^{K}\gamma(z_{ik})\log\left(\pi_k\,\varphi(x|\mu_k,\Sigma_k)\right), \tag{4}$$

де $\theta = (\pi_k, \mu_k, \Sigma_k)$ - сукупність параметрів гаусівської суміші, які оцінюються в EM-алгоритмі, а $\theta^{(t)}$ - поточне наближення цих параметрів на t-ій ітерації [2].

Означення 10 . *Для оцінювання параметрів статистичної моделі використовується* ***функція правдоподібності*** *$L(\theta;\xi)$, яка відображає ймовірність спостереження вибірки ξ, якщо дані походять з розподілу з параметром θ [7].*

Означення 11 . ***Логарифмічна функція правдоподібності (log-likelihood)*** *- це логарифм функції правдоподібності. Вона часто використовується замість звичайної функції $L(\theta;\xi)$, оскільки:*

- *логарифм переводить добуток ймовірностей у суму (спрощуючи математичні вирази),*
- *дозволяє зручніше шукати максимум за допомогою диференціювання.*

Формально:

$$\ell(\theta) = \log L(\theta;\xi)$$

де $\ell(\theta)$ - логарифм функції правдоподібності [7].

Функція log-likelihood лежить в основі методу максимізації правдоподібності, який використовується в EM-алгоритмі. Загальна форма log-likelihood-функції для моделі суміші гаусівських розподілів для всіх n спостережень має вигляд:

$$\log L(\theta) = \sum_{i=1}^{n}\log\left(\sum_{k=1}^{K}\pi_k\cdot\mathcal{N}(x_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right), \tag{5}$$

Задача полягає у максимізації цієї функції за параметрами: $\theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ (див. [2]).

Теорема 3 . *Теорема Байєса.* *У загальному випадку, теорема Байєса встановлює зв'язок між умовною ймовірністю подій і безумовними ймовірностями. Якщо події $A_1, A_2, \ldots, A_n$ утворюють повну групу (тобто попарно не перетинаються та в сукупності покривають простір елементарних подій), то для будь-якої події B з $P(B) \neq 0$ справджується:*

$$P(A_j \mid B) = \frac{P(A_j) \cdot P(B \mid A_j)}{P(B)}, \tag{6}$$

де знаменник $P(B)$ визначається за формулою повної ймовірності:

$$P(B) = \sum_{j=1}^{n} P(A_j) \cdot P(B \mid A_j)$$

(див. [3]).

Означення 12 . *Латентні змінні* *- це невидимі випадкові змінні, які, як припускається, лежать в основі спостережуваних даних. Вони моделюють приховану структуру, що керує генерацією даних.*

У латентних моделях розподіл спостережуваного вектора x задається через маргіналізацію за латентною змінною t:

$$p(x) = \int p(x \mid t) \cdot p(t)\, dt$$

або у дискретному випадку:

$$p(x) = \sum_{i=1}^{n} p(x \mid t = t_i) \cdot p(t = t_i),$$

де $p(z)$ - апріорний розподіл латентної змінної, а $p(x \mid z)$ - умовний розподіл спостережуваних даних за фіксованого z [8].

У моделі GMM кожне спостереження x_i вважається таким, що згенероване з однієї з K гаусівських компонент. Щоб це формалізувати, вводиться латентна змінна $T_i \in \{1, \ldots, K\}$, яка вказує, з якої компоненти було згенеровано x_i. З цього виходить, що розподіл імовірності для x_i можна записати як:

$$p(x_i) = \sum_{t_i=1}^{K} p(t_i) \cdot p(x_i \mid t_i).$$

У рамках EM-алгоритму значення t_i є невідомим, проте ми можемо оцінити ймовірність того, що x_i належить до k-го кластера. Ця ймовірність називається *"відповідальністю"* (responsibility) та обчислюється:

$$\gamma_{ik} := p(t_i = k \mid x_i) = \frac{\pi_k \cdot \mathcal{N}(x_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \cdot \mathcal{N}(x_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{7}$$

(див. [2]).

Теорема 4 . ***Метод множників Лагранжа.*** *Нехай необхідно знайти екстремум функції $f(x_1, \ldots, x_n)$ за умови виконання m обмежень у вигляді рівнянь:*

$$\Phi_i(x_1, \ldots, x_n) = 0, \quad i = 1, \ldots, m.$$

Вводимо функцію Лагранжа:

$$\mathcal{L}(x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_m) = f(x_1, \ldots, x_n) + \sum_{i=1}^{m} \lambda_i \Phi_i(x_1, \ldots, x_n),$$

де λ_i - множники Лагранжа.

Необхідна умова екстремуму: часткові похідні за всіма змінними та множниками дорівнюють нулю:

$$\frac{\partial \mathcal{L}}{\partial x_j} = 0, \quad j = 1, \ldots, n; \qquad \frac{\partial \mathcal{L}}{\partial \lambda_i} = \Phi_i(x) = 0, \quad i = 1, \ldots, m.$$

Таким чином, ми отримуємо систему з $n + m$ рівнянь для визначення n змінних та m множників [9].

2.3 Метрики для порівняння кластерезуючих алгоритмів

Інформаційні критерії AIC (Akaike Information Criterion) та **BIC** (Bayesian Information Criterion) використовуються для вибору оптимальної моделі з-поміж кількох альтернативних варіантів, забезпечуючи баланс між точністю та складністю [10].

Ці метрики є застосовними переважно в рамках ймовірнісних моделей, зокрема до моделей суміші Гауса (GMM), параметри яких оцінюються за допомогою EM-алгоритму.

AIC накладає менше штрафу за складність моделі, ніж BIC, і є більш гнучким для моделювання.

$$\mathrm{AIC}(K) = \ln\left(\frac{e'e}{n}\right) + \frac{2K}{n} \tag{8}$$

BIC сильніше карає за додаткові параметри, що робить його більш схильним до вибору простіших моделей при великій кількості спостережень.

$$\mathrm{BIC}(K) = \ln\left(\frac{e'e}{n}\right) + \frac{K\ln(n)}{n} \tag{9}$$

- $e'e$ - сума квадратів залишків, тобто $e = y - \hat{y}$,
- K - кількість параметрів у моделі,
- n - обсяг вибірки [10].

Означення 13 . ***Adjusted Rand Index (ARI, скоригований індекс Ренда*** *- це метрика подібності між двома кластеризаціями, яка коригує класичний Rand Index з урахуванням випадкової згоди. Вона оцінює, наскільки дві кластеризації погоджуються в парному поділі елементів, порівнюючи їхню фактичну відповідність із очікуваною при випадковому поділі.*

ARI має значення в інтервалі $[-1, 1]$*:*

- 1 - *означає повну відповідність (ідентичні класифікації),*
- 0 - *відповідність, очікувана випадково,*
- *Значення* < 0 - *гірше, ніж випадкове співпадіння [11].*

Означення 14 . ***Normalized Mutual Information (NMI)*** *- це метрика, що вимірює подібність між двома кластеризаціями та являє собою нормалізоване значення взаємної інформації (MI), масштабоване до інтервалу* $[0, 1]$*. Значення* 0 *означає відсутність взаємної інформації (незалежність кластеризацій), а* 1 *- повну відповідність кластерних структур [12].*

Означення 15 . ***Homogeneity Score*** *- це метрика оцінки кластеризації, яка вважає результат кластеризації однорідним, якщо кожен кластер містить лише об'єкти, що належать до одного справжнього класу.*

Значення метрики лежить в інтервалі $[0, 1]$*, де* 1.0 *означає ідеальну однорідність, а* 0 *показує, те що кластеризація повністю неузгоджена з істинними класами, кожен кластер є "змішаним". Метрика не залежить від значень міток і є нечутливою до перестановок у назвах класів або кластерів [13].*

Означення 16 . ***Silhouette Score*** *- це метрика, що використовується для оцінювання якості кластеризації. Базується на ідеї порівняння схожості кожної точки з точками її власного кластера та відмінності від точок інших кластерів.*

Silhouette-коефіцієнт може набувати значень у межах від -1 *до* $+1$*:*

- *Близькі значення до* $+1$ *вказують, що точка добре віднесена до свого кластера та мають кращу якість кластеризації.*
- *Значення поблизу* 0 *означає, що точка лежить на межі між двома кластерами.*
- *Значення* < 0 *свідчать про потенційне неправильне віднесення точки до кластеру [14].*

Означення 17 . ***Індекс Калінського-Харабаша (Calinski-Harabasz Index, CH Index)*** *- це внутрішня метрика оцінки кластеризації, яка вимірює співвідношення між міжкластерною дисперсією (separation) та внутрішньокластерною дисперсією (cohesion).*

CH Index показує, наскільки щільні кластери всередині та наскільки добре вони відділені один від одного. Чим вище значення, тим краща якість кластеризації. Метрика не має фіксованої верхньої межі та залежить від кількості кластерів, розміру вибірки й характеру розподілу даних. Для аналізу зазвичай аналізують його поведінку при зміні кількості кластерів і шукають точку згину на графіку (elbow plot) [15].

Означення 18 . ***Індекс Девіса-Болдуїна (DBI)*** *- це відносна метрика кластерної кластерної валідності, яка оцінює якість кластеризації шляхом порівняння внутрішньої щільності кластерів із відстанями між ними. Менше значення DBI вказує на кращу кластеризацію.*

Вищі значення DBI свідчать про слабке відокремлення кластерів або їхню розмитість, а нижчі значення означають, що кластери є компактними та добре відокремленими, що зазвичай вважається ознакою успішного кластерного розбиття [16].

Висновок до розділу 2: було розглянуто теоретичні основи моделей сумішей Гауса та EM-алгоритму, а також метрики, що використовуються для оцінки та порівняння якості кластеризації. Це забезпечує теоретичну базу для подальшого практичного застосування.

3 Реалізація алгоритму EM (Expectation - Maximization) для оцінки параметрів Гаусової суміші

Наведене пояснення EM-алгоритму базується на викладеному у працях Оскара Контрерас Карраско та Карла Едварда Расмуссена [2, 17], де докладно описано його математичну основу та застосування до гаусівських сумішей.

Мета EM-алгоритму

Максимізувати лог-правдоподібність:

$$\log p(x|\theta) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_i|\mu_k, \Sigma_k) \right)$$

3.1 Постановка задачі

Нехай маємо множину спостережень $X = \{x_1, x_2, \ldots, x_n\}$, кожне з яких є точкою у d-вимірному евклідовому просторі. Метою є оцінити ймовірність того, що спостереження x_n було згенероване k-ою компонентою моделі. Припускаємо, що ці дані були згенеровані з імовірнісної моделі, яка складається з K компонент нормального розподілу (гаусівських кластерів). Тобто кожна точка походить з одного з кластерів, але ми не знаємо з якого саме.

Наша мета - оцінити параметри моделі:

$$\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K},$$

- π_k - коефіцієнт змішування, або вага k-го гаусівського розподілу в моделі, або апріорна ймовірність компонента k, така, що $\sum_{k=1}^{K} \pi_k = 1$;
- $\mu_k \in \mathbb{R}m$ - вектор середнього значення k-го гаусівського розподілу;
- $\Sigma_k \in \mathbb{R}^{m \times m}$ - коваріаційна матриця k-го кластеру, яка визначає форму, розмір та орієнтацію гаусівського розподілу.

3.2 Формалізація GMM: ймовірності та латентні змінні

Для знаходження цього вводимо латентну змінну t_{nk}, яка дорівнює 1, якщо x_n належить k-ій компоненті, в іншому випадку - 0.

$$p(t_{nk} = 1 \mid x_n) \tag{10}$$

Ця величина інтерпретується як ймовірність того, що x_n було згенеровано з k-го кластеру. Далі вводимо:

$$\pi_k = p(t_k = 1) \tag{11}$$

Позначимо t як вектор усіх латентних змінних:

$$t = \{t_1, \ldots, t_K\}$$

Оскільки латентна змінна t, яка безпосередньо не спостерігається, визначає до якого кластера належить спостереження x_n, нас передусім цікавить спільний розподіл $p(x_n, t)$, тобто ймовірність того, що x_n належить до компонента, який був вибраний через значення латентної змінної t.

Згідно з правилом добутку ймовірностей, маємо:

$$p(x_n, t) = p(x_n \mid t) \cdot p(t)$$

Тепер окремо розглянемо обидва множники цієї формули.

- Перший множник - це ймовірність отримати значення x_n за умови, що воно походить з певного компонента, визначеного t. Оскільки лише одна компонента активна (через те що t - one-hot вектор, у якому лише одна координата дорівнює одиниці, а решта - нулі), отримаємо:

$$p(x_n \mid t) = \prod_{k=1}^{K} \mathcal{N}(x_n \mid \mu_k, \Sigma_k)^{t_k} \tag{12}$$

- Другий множник - це апріорна ймовірність реалізації вектора t, тобто ймовірність того, що активною є саме компонента k, коли $t_k = 1$. Оскільки компоненти незалежні:

$$p(t) = \prod_{k=1}^{K} \pi_k^{t_k} \tag{13}$$

Оскільки t не спостерігається безпосередньо, нас цікавить не умовна, а повна ймовірність $p(x_n)$, що враховує всі можливі значення t. Тому ми виконуємо *marginalizing* за латентною змінною. Якщо t - це вектор типу one-hot, така marginalizing еквівалентна сумуванню ймовірностей для всіх компонент моделі.

Згідно з правилом повної ймовірності, отримаємо:

$$p(x_n) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_n \mid \mu_k, \Sigma_k) \tag{14}$$

Ця формула фактично описує загальну (сумарну) ймовірність того, що точка x_n з'явилася в рамках моделі, яка складається з кількох гаусівських розподілів. Іншими словами, ми беремо взважену суму ймовірностей з усіх компонент, що дозволяє адекватно описати розподіл даних у складних, мультимодальних ситуаціях.

Таким чином, (6) є **основним рівнянням гаусівської суміші**, і на ній базується вся подальша процедура оцінювання параметрів моделі через EM-алгоритм.

Для всіх n спостережень:

$$p(X) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_n \mid \mu_k, \Sigma_k) \tag{15}$$

І, відповідно, логарифм лог-правдоподібності:

$$\log p(X) = \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_n \mid \mu_k, \Sigma_k) \right) \tag{16}$$

Логарифм у виразі правдоподібності використовується для спрощення обчислень, оскільки він перетворює добуток ймовірностей на суму логарифмів, що полегшує оптимізацію, особливо під час виведення градієнтів у EM-алгоритмі.

Крім того, логарифм робить обчислення більш стабільними, бо допомагає уникнути дуже малих або надто великих чисел, які виникають при множенні багатьох ймовірностей.

Оцінка $p(t_k = 1 \mid x_n)$ - ймовірність належності

Згідно з правилом Байєса:

$$p(t_k = 1 \mid x_n) = \frac{p(x_n \mid t_k = 1) \cdot p(t_k = 1)}{\sum_{j=1}^{K} p(x_n \mid t_j = 1) \cdot p(t_j = 1)}$$

- $p(t_k = 1)$ - апріорна ймовірність вибору k-тої компоненти, тобто вага π_k;
- $p(x_n \mid t_k = 1)$ - ймовірність (щільність) генерації x_n цією компонентою, тобто гаусівська щільність з параметрами μ_k і Σ_k.

Оскільки:

$$p(t_k = 1) = \pi_k, \quad p(x_n \mid t_k = 1) = \mathcal{N}(x_n \mid \mu_k, \Sigma_k)$$

Це щільність багатовимірного нормального розподілу (гаусівської компоненти), яка має вигляд:

$$\mathcal{N}(x_n \mid \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{m/2}|\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2}(x_n - \mu_k)^\top \Sigma_k^{-1}(x_n - \mu_k)\right\}$$

Тоді підставимо ці значення у формулу Байєса:

$$p(t_k = 1 \mid x_n) = \frac{\pi_k \cdot \mathcal{N}(x_n \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \cdot \mathcal{N}(x_n \mid \mu_j, \Sigma_j)} = \gamma(t_{nk}) \tag{17}$$

Отримана формула (17) є основою для E-кроку в EM-алгоритму для обчислення "відповідальностей які відображають ймовірність того, що спостереження x_n належить k-ій кластерній компоненті.

3.3 Алгоритм Expectation-Maximization (EM) для GMM

Після побудови ймовірнісної моделі гаусівської суміші та формулювання логарифмічної функції правдоподібності, у нас постає задача оцінювання параметрів моделі: середніх μ_k, коваріаційних матриць Σ_k та вагових коефіцієнтів π_k для кожного кластеру.

Пряме максимізування лог-правдоподібності є складним через присутність латентних змінних. Саме тому використовується ітеративний підхід, а саме алгоритм очікування-максимізації (EM), який дозволяє поступово наближатися до оптимальних параметрів.

Крок 1: Ініціалізація.

Перш за все, необхідно задати початкові значення параметрів моделі. Найчастіше початкові значення обирають випадково, або з використанням результатів попередньої кластеризації (наприклад, методом K-means, див. підрозділ 4.1), що дозволяє ініціалізувати параметри μ_k, Σ_k і π_k для всіх компонент $k = 1, \ldots, K$.

Крок 2: Крок очікування (E-step).

На цьому етапі алгоритму, виходячи з поточних параметрів моделі $\theta = \{\pi_k, \mu_k, \Sigma_k\}$, обчислюються умовні ймовірності того, що кожне спостереження x_n було згенероване з k-тої компоненти гаусівської суміші. Це значення відоме як відповідальність компоненти за спостереження і позначається $\gamma(t_{nk})$. Умовну ймовірність $\gamma(t_{nk})$ відображає наскільки ймовірно, що точка x_n належить до кожного з K кластерів.

Для GMM E-крок зводиться до знаходження $\gamma(t_{nk})$. На основі цих значень, формується функція $Q(\theta^*, \theta)$ - очікуване значення повної логарифмічної правдоподібності, яку потрібно буде максимізувати на наступному кроці.

Очікуване значення повної логарифмічної правдоподібності має вигляд:

$$Q(\theta^*, \theta) = \mathbb{E}_{T \sim p(T|X,\theta)}[\log p(X, T \mid \theta^*)] = \sum_T p(T \mid X, \theta) \log p(X, T \mid \theta^*) \quad (18)$$

Використаємо рівняння (11) та в нього вставимо рівняння (10). Отримаємо:

$$Q(\theta^*, \theta) = \sum_T p(T \mid X, \theta) \log p(X, T \mid \theta^*) \to \sum_T \gamma(t_{nk}) \log p(X, T \mid \theta^*) \quad (19)$$

Для знаходження $p(X, T \mid \theta^*)$ скористаємося припущенням, що всі спостереження x_n є незалежними за умови параметрів θ^* одне від одного. Це дозволяє розкласти спільну ймовірність у вигляді добутку:

$$p(X, T \mid \theta^*) = \prod_{n=1}^{N} p(x_n, t_n \mid \theta^*) \quad (20)$$

Далі, для кожного x_n і компоненти k маємо, що якщо $t_k = 1$, то:

$$p(x_n, t_k \mid \theta^*) = \pi_k \cdot \mathcal{N}(x_n \mid \mu_k, \Sigma_k) \tag{21}$$

де π_k - апріорна ймовірність компоненти, а $\mathcal{N}(x_n \mid \mu_k, \Sigma_k)$ - щільність багатовимірного нормального розподілу з параметрами μ_k та Σ_k для k-тої компоненти.

Візьмемо логарифм добутку умовної ймовірності спостережень і латентних змінних, тоді отримаємо:

$$\log p(x_n, t_k \mid \theta^*) = \log \pi_k + \log \mathcal{N}(x_n \mid \mu_k, \Sigma_k)$$

Для кожного спостереження x_n ми враховуємо внесок від усіх K можливих компонент, до яких ця точка може належати з певною ймовірністю. Тому ми маємо:

$$Q(\theta^*, \theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(t_{nk}) \log p(x_n, t_k \mid \theta^*)$$

Таким чином, внутрішня сума по k охоплює всі компоненти, а зовнішня сума по n - усі спостереження у вибірці.

Підставимо отриманий логарифм ймовірності $p(x_n, t_k \mid \theta^*)$ у вираз функції $Q(\theta^*, \theta)$, отримаємо:

$$Q(\theta^*, \theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(t_{nk}) \left[\log \pi_k + \log \mathcal{N}(x_n \mid \mu_k, \Sigma_k)\right] \tag{22}$$

Таким чином, ми отримали зручний для обчислень вираз, який використовується на наступному M-кроці для оновлення параметрів моделі. Завдяки використанню ймовірнісного підходу через $\gamma(t_{nk})$, ми можемо уникнути жорсткого кластерного приписування і можемо коректно враховувати невизначеність належності точок до кластерів.

Крок 3: Крок максимізації (M-step)

На цьому кроці необхідно знайти такі параметри θ^*, які максимізують функцію очікуваної логарифмічної правдоподібності Q:

$$\theta^* = \arg\max_{\theta} Q(\theta^*, \theta)$$

Оскільки ваги компонент повинні задовольняти обмеження $\sum_{k=1}^{K} \pi_k = 1$, використовуємо метод Лагранжа з множником λ, що дозволяє врахувати це обмеження. Тоді функція набуває вигляду:

$$Q(\theta^*, \theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(t_{nk}) \left[\log \pi_k + \log \mathcal{N}(x_n \mid \mu_k, \Sigma_k) \right] - \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right)$$

Щоб знайти значення π_k, диференціюємо Q за π_k:

$$\frac{\partial Q}{\partial \pi_k} = \sum_{n=1}^{N} \frac{\gamma(t_{nk})}{\pi_k} - \lambda = 0$$

Розв'язавши рівняння відносно π_k, отримаємо:

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \gamma(t_{nk}) = \frac{N_k}{N}$$

Це означає, що вага компоненти k дорівнює відносній частці точок, які ймовірно належать до неї. N_k - так звана "м'яка"кількість точок у кластері k, тобто сумарна відповідальність цієї компоненти за всі точки.

Оновлення параметра μ_k виконується як зважене середнє всіх точок x_n, де вагами виступають відповідальності $\gamma(t_{nk})$:

$$\mu_k = \frac{\sum_{n=1}^{N} \gamma(t_{nk}) x_n}{\sum_{n=1}^{N} \gamma(t_{nk})}$$

Це оновлення зміщує центр компоненти в напрямку точок, які з більшою ймовірністю належать до неї.

Коваріаційна матриця Σ_k оновлюється як зважена оцінка дисперсії:

$$\Sigma_k = \frac{\sum_{n=1}^{N} \gamma(t_{nk})(x_n - \mu_k)(x_n - \mu_k)^\top}{\sum_{n=1}^{N} \gamma(t_{nk})}$$

Ця матриця відображає форму, розмір та орієнтацію k-ї компоненти, просторово описуючи розподіл точок навколо центру μ_k з урахуванням ймовірнісного належності.

3.4 Цикл оновлення та збіжність EM-алгоритму

Ітеративний характер EM-алгоритму.

Алгоритм EM є ітеративним: він починається з початкового наближення параметрів моделі, після чого на кожному кроці виконує два основні етапи - очікування (E-крок), де обчислюються ймовірності належності об'єктів до кластерів, та максимізацію (M-крок), де оновлюються параметри моделі..

Ці кроки повторюються доти, доки зміна логарифмічної правдоподібності не стане меншою за наперед заданий поріг, або доки не буде досягнуто максимальної кількості ітерацій.

Теорема 5 . ***Збіжність EM-алгоритму.*** *Нехай X - спостережувані дані, а θ - параметри моделі. Тоді послідовність $\{\theta^{(t)}\}$, згенерована алгоритмом очікування-максимізації (EM), задовольняє:*

$$\log p(X \mid \theta^{(t+1)}) \geq \log p(X \mid \theta^{(t)}),$$

тобто логарифм правдоподібності не зменшується з кожною ітерацією, що гарантує монотонну збіжність [18].

Наслідок 1 . *EM-алгоритм збігається до стаціонарної точки (локального максимуму або сідлової точки) функції логарифмічної правдоподібності.*

Після кожного M-кроку оновлені параметри π_k, μ_k, Σ_k підставляються у наступному E-кроці. Такий цикл повторюється до досягнення збіжності. Для контролю збіжності використовують логарифмічну правдоподібність даних:

$$\log p(X) = \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_n \mid \mu_k, \Sigma_k) \right)$$

Значення цієї функції зростає, або залишається незмінним у процесі ітерацій, що гарантує збіжність алгоритму до стаціонарної точки, тобто до максимуму функції.

3.5 Практична реалізація EM-алгоритму

Для ілюстрації роботи EM-алгоритму його реалізацію продемонстровано спочатку на синтетичних двовимірних даних, згенерованих як суміш двох нормальних розподілів із наперед заданими параметрами. Такий підхід дозволяє

візуалізувати процес кластеризації, спостерігати збіжність алгоритму, а також перевірити точність оцінювання параметрів моделі.

Також для демонстрації роботи EM-алгоритму на реальних даних було обрано класичний набір *Iris flower data set*. Цей набір був представлений Рональдом Фішером у 1936 році у статті *"The use of multiple measurements in taxonomic problems"*, але також можна почути, що це набір Андерсона, який зібрав ці дані з метою кількісного аналізу морфологічної варіації трьох близьких видів ірисів [19].

Набір складається зі 150 зразків, по 50 від кожного з трьох видів: *Iris Setosa*, *Iris Versicolor* та *Iris Virginica*. Для кожної квітки надано чотири числові ознаки: довжина та ширина чашолистка і пелюстки (у сантиметрах), а також мітка класу (див. також [19]).

Цей датасет було обрано через його широке використання як стандартного тестового прикладу в машинному навчанні, зокрема для оцінки алгоритмів класифікації та кластеризації.

У рамках експерименту дані кластеризуються із використанням моделі гаусівської суміші (GMM) з $K = 3$ компонентами. Алгоритм ініціалізується випадково та поступово наближається до розв'язку за допомогою ітерацій EM. Нижче наведено фрагмент коду, що реалізує цю процедуру.

Опис основних функцій реалізації EM-алгоритму для GMM:

(див. Додаток А 4.6 для реалізації EM-алгоритму)

- `initialize_parameters(X, K)` - ініціалізує параметри моделі: ваги π_k, центри μ_k та матриці коваріації Σ_k.
- `e_step(X, pi, mu, sigma)` - обчислює матрицю відповідальностей γ_{nk}, що відображає ймовірність належності кожної точки до кожної компоненти.
- `m_step(X, gamma)` - оновлює параметри π_k, μ_k, Σ_k на основі поточних відповідальностей.
- `compute_log_likelihood(X, pi, mu, sigma)` - обчислює логарифм правдоподібності для поточних параметрів.

- `em_gmm(X, K)` - основна функція EM-алгоритму: виконує ітерації E- та M-кроків до досягнення збіжності.

Імпортування бібліотек

```
import numpy as np
from scipy.stats import multivariate_normal
```

Генерація синтетичних даних

```
np.random.seed(42)
n_samples = 300
X1 = np.random.multivariate_normal([0, 0], np.eye(2), n_samples)
X2 = np.random.multivariate_normal([5, 5], np.eye(2), n_samples)
X = np.vstack((X1, X2))
```

Цей код створює два набори двовимірних точок із нормального розподілу з центрами в точках $(0, 0)$ та $(5, 5)$ відповідно, кожен по 300 точок. Потім вони об'єднуються в один масив X.

Рис. 3: Результат кластеризації синтетичних даних.

Кластеризація набору Iris

Для кластеризації використовувались лише числові ознаки (довжини та ширини чашолистків і пелюсток). Істинні мітки використовувались лише для

оцінки якості кластеризації.

ЕМ-алгоритм виконував кластеризацію з $K = 3$ кластерами.

```
from sklearn.datasets import load_iris
import numpy as np
import matplotlib.pyplot as plt

iris = load_iris()
X_iris = iris.data[:, :2]
y_iris = iris.target

K = 3
pi_iris, mu_iris, sigma_iris, gamma_iris = em_gmm(X_iris, K=K, max_iter=100,
    tol=1e-4, verbose=True, seed=42)
```

Оцінка якості кластеризації на Iris dataset

Для оцінки результатів кластеризації методом ЕМ використано такі метрики:

- **ARI** = 0,4781 - помірний збіг кластерів з істинними мітками.
- **NMI** = 0,5526 - часткова спільність інформації між кластеризацією та реальними класами.
- **HS** = 0,5522 - кластери частково однорідні, містять переважно точки одного класу.
- **Sil** = 0,2217 - слабка внутрішня згуртованість і роздільність кластерів.
- **CH** = 75,63 - задовільна згуртованість кластерів та відстань між ними.
- **DB** = 2,0674 - наявність помітного перекриття між кластерами.

Отримані результати вказують на помірну відповідність кластерів справжній структурі даних та часткову відокремленість. Найкраще кластеризований клас *Setosa*, тоді як класи *Versicolor* та *Virginica* частково перекриваються, що відповідає відомій особливості цього набору даних.

Для $K = 2$ значення **AIC** = 473,83 та **BIC** = 506,95.

Для $K = 3$ значення **AIC** = 478,14 та **BIC** = 529,32.

Для $K = 4$ значення **AIC** = 469,16 та **BIC** = 538,41.

Отже, **AIC має найкращий показник у моделі з K = 4 (469,16)**, тоді як **BIC - з K = 2 (506,95)**. Вибір між ними залежить від пріоритету: краща якість кластеризації (AIC), чи менша складність моделі (BIC). З огляду на баланс між якістю моделі та її складністю, а також на відомий факт про наявність трьох реальних класів у даних, **оптимальним вибором є** $K = 3$.

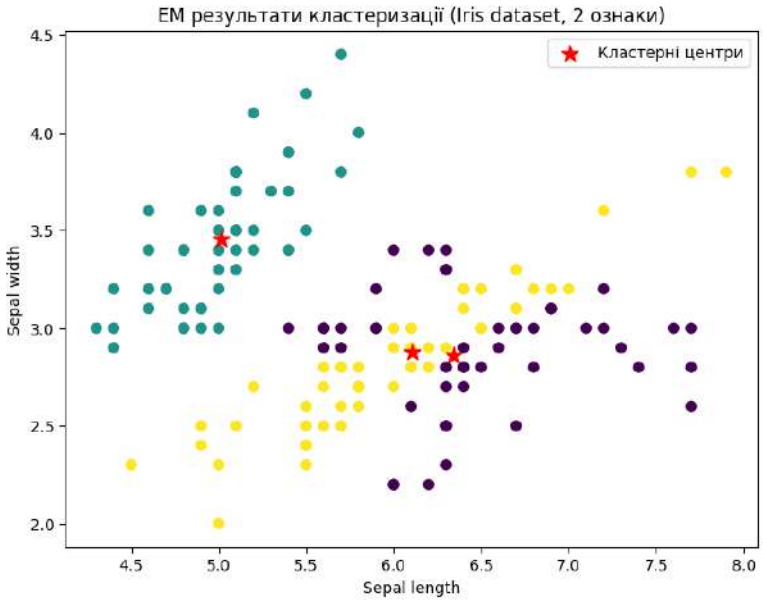

Рис. 4: Результати кластеризації даних Iris за допомогою EM-алгоритму, при $K = 3$.

Висновки до розділу 3: у цьому розділі було реалізовано EM-алгоритм для оцінювання параметрів моделей гаусових сумішей. Проведено покроковий опис ініціалізації, етапів очікування та максимізації, а також формально обгрунтовано збіжність методу. Отримані результати кластеризації на синтетичних та реальних даних (набір Iris) підтвердили ефективність і коректність реалізованого алгоритму.

4 Експерименти з даними. Порівняння результатів класифікації за допомогою Гаусових сумішей з іншими методами

У цьому розділі розглянуто застосування трьох алгоритмів кластеризації: EM, K-means та Agglomerative до реальних наборів даних з різною структурою. Метою є оцінка їх ефективності за допомогою обчислюваних метрик та візуалізації.

Для експериментів використано дані з відкритих джерел:

- **Fashion-MNIST** [20] - зображення предметів одягу, що використовуються як складний приклад для кластеризації. Ці дані не мають вираженої гаусової структури, перед кластеризацією було застосовано зменшення розмірності методом головних компонент (PCA).
- **DOU Salaries (грудень 2024)** [21] - реальний табличний датасет із платформою GitHub, що містить інформацію про зарплати працівників IT-сфери в Україні. Набір використовується для дослідження впливу кількості кластерів K на якість кластеризації, а також для порівняння результатів між трьома алгоритмами.

Аналіз включатиме як візуалізацію кластерів після попередньої обробки (зокрема PCA), так і обчислення метрик якості кластеризації, таких як: ARI, NMI, Silhouette Score тощо.

Означення 19 . ***Метод головних компонент (PCA)*** *- це метод зменшення розмірності, що використовує ортогональні перетворення для відображення вихідних змінних у новий простір з найвищою дисперсією [22].*

Перед застосуванням методу головних компонент змінні необхідно центрувати (віднімати середнє) та масштабувати до однакової дисперсії (стандартного відхилення, рівного одиниці), якщо їх початкові одиниці виміру або варіація суттєво відрізняються.

Без масштабування результати PCA можуть бути спотворені, наприклад, змінна з найбільшою дисперсією домінуватиме у першій головній компоненті, що змістить аналіз у бік технічного масштабу, а не реальної структури даних.

Масштабування забезпечує незалежність головних компонент від одиниць виміру вихідних ознак і дозволяє адекватно інтерпретувати багатовимірну структуру даних [22].

4.1 K-середнє кластеризування

Означення 20 . *K-середніх (K-Means)* *- це алгоритм машинного навчання без учителя, призначений для автоматичного поділу немаркованих даних на кластери, за критерієм схожості. Його широко застосовують для сегментації даних, зокрема у маркетингу, де, наприклад, клієнтів онлайн-магазину можна розділити на категорії на основі частоти покупок та витрат: "економні покупці", "активні клієнти", "великі витратники" тощо [23].*

Алгоритм кластеризації K-Means

Для опису алгоритму було використано [22, 23] . Метод K-середніх передбачає розбиття множини спостережень на K невзаємоперетинних кластерів $C_1, \ldots, C_K$, таких що:

1. $\bigcup_{k=1}^{K} C_k = \{1, \ldots, n\}$ - кожне спостереження належить до щонайменше одного кластеру;

2. $C_k \cap C_{k'} = \emptyset$ для всіх $k \neq k'$ - кластери не перетинаються.

Завдання полягає в мінімізації загальної внутрішньокластерної дисперсії, яка визначається за формулою:

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k),$$

де $W(C_k)$ - міра відстані між точками в кластері C_k. Для евклідової відстані в квадраті вона має вигляд:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2,$$

де x_{ij} - j-та ознака i-го спостереження.

Підставивши це у загальну формулу, отримуємо повну цільову функцію, яку мінімізує алгоритм:

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2.$$

де $\bar{x}_{kj} = \frac{1}{|C_k|} \sum\limits_{i \in C_k} x_{ij}$ - середнє значення ознаки j в k-му кластері.

Алгоритм працює ітеративно:

1. Ініціалізація: обираються K початкових центрів кластерів (наприклад, випадково або методом *K-Means++*).

2. Повторюється:

   (а) Кожне спостереження x_i призначається до найближчого центру:

   $$\text{cluster}(x_i) = \arg\min_k \|x_i - c_k\|^2.$$

   (б) Центроїди оновлюються:

   $$c_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i.$$

3. Алгоритм зупиняється, коли кластерні мітки перестають змінюватися або досягнуто максимум ітерацій.

Зауваження. Алгоритм K-середніх знаходить лише локальний мінімум, тому його рекомендується запускати кілька разів з різними початковими умовами. Найкраще кластеризування вибирається за мінімальним значенням цільової функції [22].

4.2 Агломеративне ієрархічне кластеризування

Означення 21 . ***Ієрархічна кластеризація*** *- це розділення даних на різні групи з ієрархії кластерів на основі певної міри подібності [25].*

Означення 22 ***Агломеративне кластеризування (Agglomerative)*** *- це метод ієрархічної кластеризації, що починається з того, що кожне спостереження є окремим кластером, після чого відбувається поетапне злиття найближчих кластерів до досягнення бажаної кількості. Цей підхід реалізує стратегію знизу вгору (bottom-up), поступово будуючи ієрархію від індивідуальних елементів до одного загального кластеру [26].*

Алгоритм агломеративного кластеризування

Агломеративне кластеризування починається з того, що кожне спостереження вважається окремим кластером, після чого ітеративно виконується об′єднання найближчих кластерів доти, доки не залишиться лише один кластер або не буде досягнуто заздалегідь визначеної кількості кластерів K. Для опису Agglomerative алгоритму було використано: [22, 26].

1. **Ініціалізація:** Нехай $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ - набір даних з n елементів. Кожне спостереження x_i початково вважається окремим кластером:

$$\mathcal{C} = \{\{x_1\}, \{x_2\}, \ldots, \{x_n\}\}.$$

2. **Обчислення відстаней:** для кожної пари кластерів обчислюється міжкластерна відстань $d(C_i, C_j)$ (наприклад, євклідова).

$$d(C_i, C_j) = \min_{x \in C_i,\, y \in C_j} \|x - y\|_2.$$

3. **Об′єднання найближчих кластерів:** Знаходимо пару кластерів (C_p, C_q) з найменшою відстанню $d(C_p, C_q)$ і об'єднуємо їх:

$$C_{\text{new}} = C_p \cup C_q.$$

4. **Оновлення відстаней:** переобчислюємо відстані між новим кластером C_{new} і всіма іншими кластерами відповідно до обраного зв'язування:

   - **Single linkage:** найменша попарна відстань.
   - **Complete linkage:** найбільша попарна відстань.

- **Average linkage:** середнє всіх попарних відстаней.
- **Centroid linkage:** відстань між центроїдами кластерів.

5. **Повторення:** кроки 2-4 повторюються до досягнення K кластерів або одного.

6. **Побудова дендрограми:** Візуалізуємо процес злиття кластерів у вигляді дендрограми.

7. **Визначення кластерів:** обирається рівень "обрізання"дендрограми на основі найбільшого розриву у висоті.

4.3 Практична реалізація K-Means та Agglomerative алгоритмів

У рамках даної роботи алгоритми кластеризації K-середніх та агломеративного ієрархічного кластеризування реалізовано за допомогою бібліотеки `scikit-learn` [24, 27]. Для K-means використано реалізацію на основі класичного алгоритму Lloyd's з підтримкою ініціалізації методом *K-Means++*, а для Agglomerative Clustering - реалізацію з можливістю вибору метрики відстані та типу зв'язування (*linkage*).

Код реалізації (див. Додаток Б 4.6 для реалізації коду обчислення метрик).

```
from sklearn.cluster import KMeans, AgglomerativeClustering

models = {
    "K-Means": KMeans(n_clusters=n_clusters, random_state=42),
    "Agglomerative": AgglomerativeClustering(n_clusters=n_clusters, linkage=
        'ward')
}
```

У наведеному прикладі створено два об'єкти моделей кластеризації з однаковою кількістю кластерів *n_clusters*. Алгоритм K-середніх використовує параметр `random_state` для відтворюваності результатів, а агломеративна кластеризація реалізована з використанням методу `ward`, який мінімізує дисперсію всередині кластерів.

4.4 Порівняння EM, K-Means та Agglomerative алгоритмів

У таблицях 1 та 2 представлено порівняння основних кластеризаційних алгоритмів, що розглядалися в цій роботі: Expectation-Maximization (GMM), K-means та агломеративного кластеризування. Першу таблицю присвячено їхнім ключовим перевагам, тоді як друга узагальнює основні недоліки та обмеження кожного з підходів.

Алгоритми **K-Means** та **Expectation-Maximization** мають схожу ітеративну структуру з двома кроками: оновлення належностей та параметрів кластерів. GMM забезпечує м'яке кластеризування і краще підходить у разі перекриття кластерів, проте потребує задання K і є чутливим до початкових умов. Метод K-Means - простий і швидкий, однак малоефективний для складних структур і не гарантує глобального оптимуму.

Натомість агломеративне кластеризування забезпечує високу гнучкість, дозволяє працювати без наперед заданого K, добре справляється з довільною формою кластерів, але має високу обчислювальну складність та обмежену здатність до адаптації після злиття.

Таким чином, вибір алгоритму залежить від характеристик даних, обсягу вибірки та поставлених цілей аналізу.

Табл. 1: Переваги кластеризаційних алгоритмів

Алгоритм	**Переваги**
Expectation-Maximization (GMM)	• Дає м'яке кластеризування (кожна точка має розподіл належності); • Добре працює, якщо кластери мають форму еліпсів; • Має ймовірнісну інтерпретацію.
K-Means	• Простота реалізації та інтерпретації. • Висока швидкодія на невеликих і середніх обсягах даних. • Підходить для кластерів, що мають сферичну форму та однаковий розмір. • Дає змогу ефективно масштабуватися на великі обсяги даних (при оптимізації реалізації).
Agglomerative Clustering	• Не потребує попереднього задання кількості кластерів; • Побудова дендрограми дає гнучкий контроль над рівнем кластеризації; • Добре працює для кластерів довільної форми; • Єдиний запуск дає повну ієрархію вкладених кластерів; • Можна використовувати різні метрики відстані та методи звязування (single, complete, average).

Табл. 2: Недоліки кластеризаційних алгоритмів

Алгоритм	**Недоліки**
Expectation-Maximization (GMM)	• Потрібно задавати K наперед; • Може зійтися до локального максимуму; • Погано працює, якщо дані мають нерівномірну щільність або сильно перекриваються.
K-Means	• Не гарантує глобального оптимуму; результат залежить від початкової ініціалізації центроїдів. • Не працює добре з витягнутими або перекритими кластерами. • Чутливий до викидів, які можуть змістити центроїд. • Потрібно заздалегідь знати або вказати кількість кластерів K.
Agglomerative Clustering	• Висока обчислювальна складність - непридатний для великих наборів даних без оптимізацій; • Не підтримує перегрупування: об′єкти не можуть переходити між кластерами після злиття; • Чутливий до шуму та викидів; • Результати можуть змінюватись при виборі різної метрики або методу зв′язування.

Часова складність однієї ітерації

Для EM-алгоритму:

$$\mathcal{O}(N \cdot K \cdot m^2)$$

- N - кількість точок у вибірці,
- K - кількість гаусівських компонент,
- m - кількість ознак (розмірність простору).

Пояснення:

- **E-крок.** Для кожної з N точок обчислюється ймовірність належності до кожної з K компонент. Для цього потрібно обчислити значення багатовимірної нормальної щільності, що включає інверсію та множення матриць розмірності $m \times m$. Це дає складність $\mathcal{O}(K \cdot m^2)$ на одну точку.
- **M-крок.** Оновлення параметрів кожної компоненти (ваг π_k, середніх μ_k та коваріаційних матриць Σ_k) вимагає проходження по всіх N точках для кожної з K компонент, з аналогічною складністю на матричні операції. Загальна складність також становить $\mathcal{O}(N \cdot K \cdot m^2)$.

Для K-Means:

$$\mathcal{O}(N \cdot K \cdot m)$$

- N - кількість точок у вибірці,
- K - кількість кластерів,
- m - кількість ознак (розмірність простору).

Пояснення:

- **Крок призначення.** Для кожної з N точок обчислюється відстань до кожного з K центроїдів. Обчислення евклідової відстані у просторі розмірності m потребує $\mathcal{O}(m)$ операцій. Загальна складність цього кроку - $\mathcal{O}(N \cdot K \cdot m)$.

- **Крок оновлення центроїдів.** Для кожного з K кластерів обчислюється новий центроїд як середнє значення по всіх точках у цьому кластері. Сума по N точках у просторі m має складність $\mathcal{O}(N \cdot m)$.

- **Загальна складність.** Основну частину витрат формує саме крок призначення, тому загальна оцінка складності однієї ітерації - $\mathcal{O}(N \cdot K \cdot m)$.

Для Agglomerative:

Агломеративне кластеризування включає обчислення та оновлення попарних відстаней між кластерами. При наївній реалізації алгоритму складність має вигляд:

$$\mathcal{O}(n^3)$$

n - кількість об′єктів.

Пояснення:

- Початкове обчислення матриці відстаней між n об′єктами: $\mathcal{O}(n^2)$;

- Кількість ітерацій - $n - 1$ (злиттів);

- На кожному кроці потрібно оновлювати $\mathcal{O}(n)$ відстаней до нового кластера.

Найменшу обчислювальну складність має K-Means, що робить його ефективним для великих наборів даних. Expectation-Maximization (GMM) є складнішим через операції з коваріаційними матрицями, а Agglomerative Clustering є найменш масштабованим через кубічну складність, що обмежує його використання лише на малих вибірках.

Умови завершення алгоритмів кластеризації

Expectation-Maximization (EM) для GMM. Алгоритм завершується, якщо виконується одна з умов:

- Приріст лог-правдоподібності між ітераціями стає меншим за заданий поріг;

- Досягнуто максимальну кількість ітерацій.

Алгоритм K-середніх. Кластеризація зупиняється при виконанні однієї з наступних умов:

- Центроїди не змінюються після ітерації (кластеризація стабілізувалася);
- Жодне спостереження не змінило кластер;
- Досягнуто встановленої максимальної кількості ітерацій;
- Зменшення значення цільової функції менше за поріг ε:

$$|J^{(t)} - J^{(t-1)}| < \varepsilon,$$

де $J^{(t)}$ - значення функції на ітерації t.

Агломеративне кластеризування. Є детермінованим процесом і зупиняється при:

- Досягненні заданої кількості кластерів K (тобто $|\mathcal{C}| = K$);
- Об'єднанні всіх об'єктів у один кластер;
- При побудові дендрограми - на рівні заданої висоти "обрізки"(cut height), яка визначає рівень деталізації кластерів.

Умови завершення кластеризаційних алгоритмів демонструють різну гнучкість і специфіку кожного методу. EM та K-Means мають гнучкі критерії, наприклад, якщо зміни стали дуже малими або досягнуто максимум ітерацій. Це дозволяє краще контролювати процес. Натомість Agglomerative Clustering є жорстко детермінованим і завершується при досягненні заздалегідь заданої кількості кластерів або рівня ієрархії, що робить його менш адаптивним.

4.5 Огляд та попередня підготовка наборів данних

Про датасет Fashion-MNIST

Fashion-MNIST - це набір даних зображення предметів одягу, що зібрані компанією Zalando, які використовуються як альтернативний датасет до класичного MNIST. У наборі є 60 000 зображень для навчання та 10 000 для

тестування. Кожне зображення - це монохромна картинка розміром 28×28 пікселів, що відноситься до одного з десяти попередньо визначених класів.

Розмірність одного зображення становить $28 \times 28 = 784$ пікселі. Піксельне значення - це ціле число від 0 до 255, що відображає рівень яскравості: 0 відповідає білому кольору, а 255 - майже чорному. Кожен об'єкт у наборі представлений як один рядок із 785 значень: перше - це мітка класу, решта показує інтенсивність пікселів, які згорнуті в одномірний вектор.

Щоб відновити розташування пікселя на початковій сітці 28×28, можна скористатися формулою $x = i \cdot 28 + j$, де i - номер рядка, а j - номер стовпця (обидва в межах від 0 до 27) [20].

Наприклад, `pixel150` має індекс $x = 150$, тоді $x = 5 \cdot 28 + 10 = 150$. Отже, піксель розташований у 6-му рядку ($i = 5$) та 11-му стовпці ($j = 10$).

Кожному прикладу присвоєно одну з таких міток:

0. **T-shirt/top** - футболка
1. **Trouser** - штани
2. **Pullover** - пуловер
3. **Dress** - сукня
4. **Coat** - пальто
5. **Sandal** - сандалі
6. **Shirt** - сорочка
7. **Sneaker** - кросівки
8. **Bag** - сумка
9. **Ankle boot** - чоботи до щиколотки

Попереднє очищення даних

Для забезпечення коректного застосування кластеризуючих методів, які будуть розглянуті далі з метою порівняння до EM-алгоритму, до датасету Fashion-

MNIST попередньо було застосовано PCA для зменшення розмірності з 785 до 50 компонент.

Опис структури DOU Salaries

Для дослідження кластеризації соціально-економічних даних було використано відкритий датасет *DOU Salaries* за грудень 2024 року [21]. Цей набір містить анонімні відповіді працівників IT-сфери в Україні щодо заробітної плати, професійного досвіду, зайнятості та місця проживання тощо.

Найважливіші колонки, присутні в наборі:

- **Ваша основна зайнятість в IT зараз...** - тип зайнятості (фултайм, втратив роботу тощо);
- **Зарплата / дохід в IT у $ за місяць, лише ставка ЧИСТИМИ** - місячний дохід після податків;
- **Оберіть ваш тайтл або роль у компанії** - позиція (Junior, Middle, Senior, Team Lead тощо);
- **Categories / Position / Розробники - спеціалізація** - категорія та профільна роль;
- **Основна мова програмування** - мова, з якою працює респондент;
- **Основний напрям роботи компанії** - галузь (сервіс, продукт тощо);
- **Загальний стаж роботи за нинішньою IT-спеціальністю** - досвід у роках;
- **Знання англійської мови** - рівень володіння (Upper-Intermediate, Advanced, тощо, тобто A1-C2);
- **Де ви зараз живете? / В якій області ви зараз живете?** - географічне розташування в Україні, чи поза межами;
- **Ваша стать** - гендер опитуваного.

Обробка датасету DOU Salaries

Під час попереднього аналізу датасету *DOU Salaries* (грудень 2024) було виявлено значну кількість пропущених, або неінформативних значень у змінних. З метою підвищення точності та стабільності обчислень було видалено рядки:

- Ознаки, які не несуть цінності для кластеризації (*Submitted at*) були виключені з подальшого аналізу.
- Записи, у яких значення представлені як “NaN“, “-“, або порожні рядки, що не можуть бути однозначно інтерпретовані або коректно закодовані.
- пропущені значення в ключових категоріальних змінних:
  - *Categories* - відсутнє у 36 записах;
  - *Розробники - основна спеціалізація* - пропущено у 8 327 випадках;
  - *Основна мова програмування* - відсутня у 5 364 записах;
  - *В якій області ви зараз живете?* - порожнє у 2 622 записах.
- Також було видалено записи з $PC2 < -10$ після застосування PCA, щоб усунути крайні значення та покращити якість кластеризації.
- **Кількість записів до очищення:** 14 148
- **Після очищення:** 4 742
- **Частка видалених записів:** 66,5%

Незважаючи на суттєве зменшення обсягу вибірки, очищення було необхідним для забезпечення стабільної та коректної роботи кластеризуючих алгоритмів для мінімізації впливу шуму.

4.6 Оцінка та порівняння метрик кластеризації

Порівняння для Fashion MNIST

У результаті кластеризації з використанням трьох алгоритмів - K-середніх, агломеративного та EM кластерезування на датасеті **Fashion MNIST** бу-

ло проведено порівняння за низкою якісних метрик: ARI, NMI, Homogeneity Score, Silhouette Score, Calinski-Harabasz Index та Davies-Bouldin Index, яке можна побачити в (табл. 3), також матриць невідповідностей на рис. 6 (див. код в Додаток В **??** для реалізації порівнянь).

Табл. 3: Порівняння кластеризаційних алгоритмів за якісними метриками для датасету Fashion MNIST.

Метрика	**K-Means**	**Agglomerative**	**EM**	**EM (sklearn)**
ARI	0.00228	-0.00094	0.00032	**0.00508**
NMI	0.02359	0.01850	0.01753	**0.02972**
Homogeneity Score	0.02312	0.01819	0.01665	**0.02914**
Silhouette Score	**0.18797**	0.17923	0.09425	0.15432
CH Index	**149.09**	133.98	62.76	122.43
DBI	**1.709**	1.746	3.416	1.914
Час (с)	**0.0135**	0.0634	1.3646	1.0004

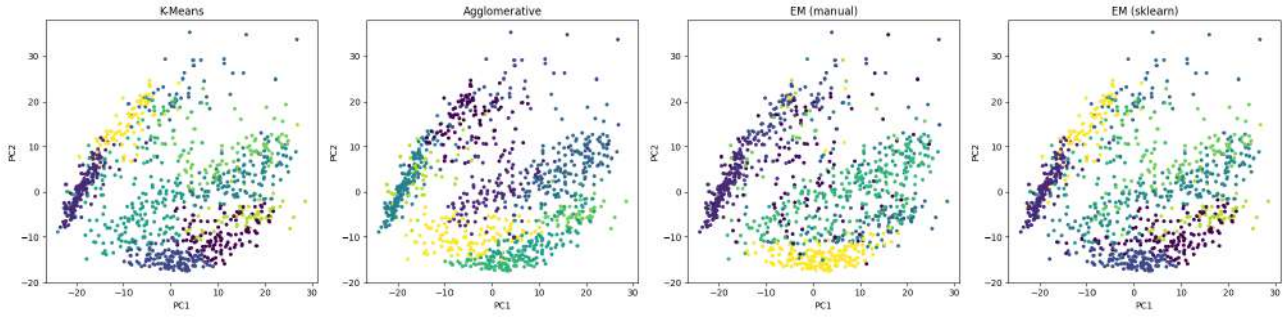

Рис. 5: Візуалізація результатів кластеризації на датасеті Fashion MNIST трьома алгоритмами (K-Means, Agglomerative, EM (GMM)) у просторі перших двох головних компонент (PC1 та PC2).

На основі результатів, наведених у табл. 3, можна зробити такі висновки:

- **K-Means** показав найкращі геометричні метрики кластеризації, зокрема:
  - найвище значення Silhouette Score (0.18797);
  - найвищий Calinski-Harabasz Index (149.09);
  - найнижчий Davies-Bouldin Index (1.709);

– а також найменший час виконання (0.0135 с).

Проте, значення ARI, NMI та Homogeneity є дуже низькими, що вказує на слабкий збіг із реальними класами.

- **Agglomerative Clustering** має схожі геометричні метрики, але ще гірші результати за ARI, NMI та Homogeneity.
- **EM (ручна реалізація)** продемонструвала найгірші результати за більшістю метрик. Зокрема, Silhouette Score = 0.09425 та DBI = 3.416 свідчать про низьку якість кластеризації. Також ця реалізація є найповільнішою.
- **EM (sklearn)** має найкращі значення **ARI (0.00508)**, **NMI (0.02972)** та **Homogeneity Score (0.02914)** серед усіх моделей, що означає найкращу відповідність реальним міткам. За геометричними метриками поступається K-Means, однак забезпечує баланс між якістю кластеризації та обгрунтованістю розподілу.

Слід також враховувати, що **датасет Fashion MNIST не є гаусівським за своєю природою**: зображення одягу в піксельному просторі не підпорядковуються нормальному розподілу. Це частково пояснює нижчі результати алгоритму EM (GMM), який припускає, що дані походять із суміші гаусівських компонент. У таких випадках моделі, що не роблять ймовірнісних припущень, як Agglomerative, або K-Means можуть виявлятися більш ефективними.

Аналіз матриць невідповідностей

Оскільки у датасеті Fashion MNIST наявні справжні мітки класів, це дозволяє не лише використовувати зовнішні метрики якості кластеризації, але й провести детальний аналіз результатів кластеризації за допомогою **матриць невідповідностей**, які відображають відповідність між істинними класами та отриманими кластерами [28].

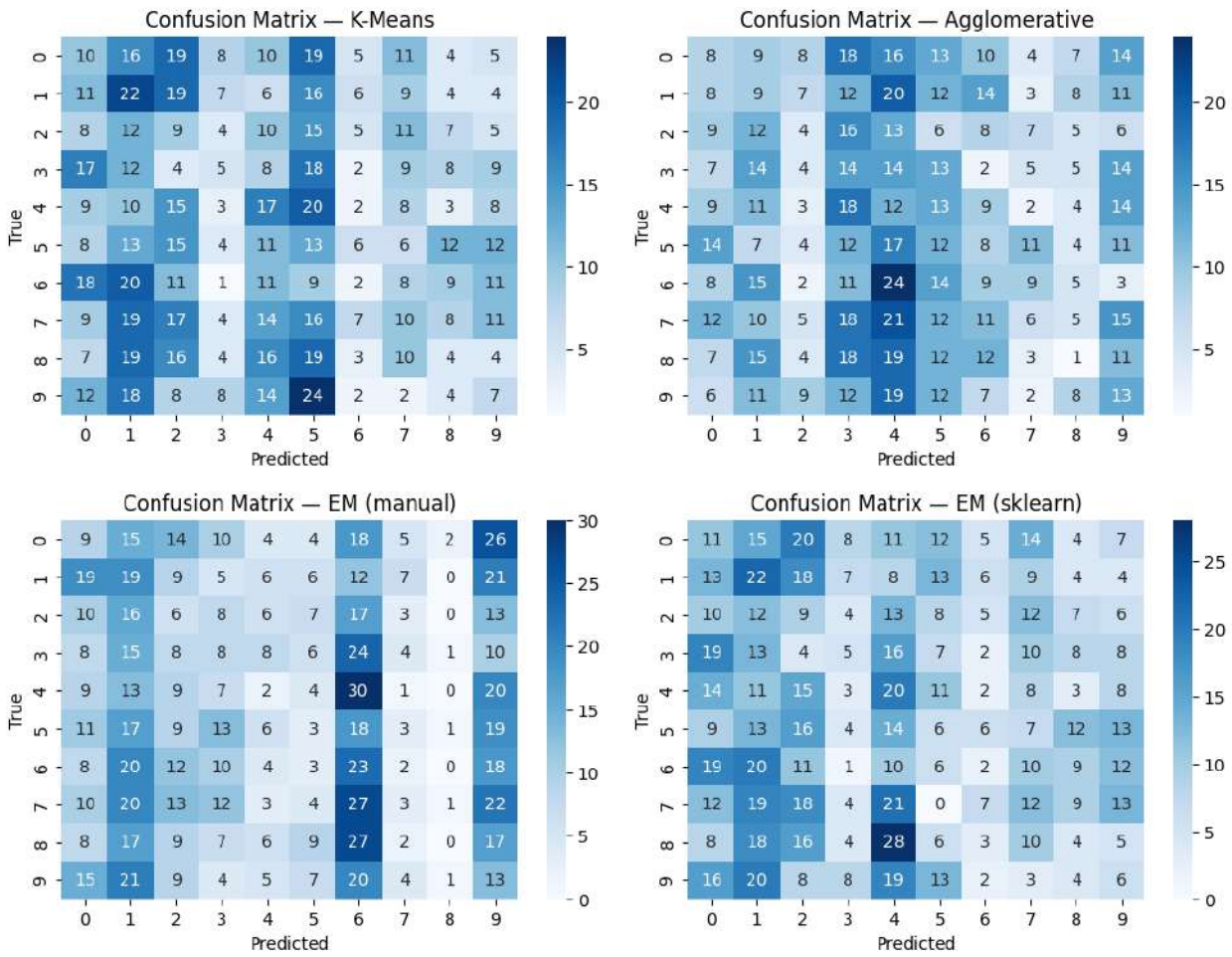

Рис. 6: Матриці неточностей (Confusion Matrices) для алгоритмів кластеризації на Fashion MNIST.

На рис. 6 зображено матриці невідповідностей для кластеризації з $K = 10$. Основні спостереження:

- **K-Means** демонструє відносно рівномірний розподіл кластерів. Проте спостерігається змішування класів: одна і та ж група об'єднує об'єкти різних типів. Наприклад, 5-й кластер (сандалі) містить зразки з інших класів, зокрема кросівки (7) та чоботи (9).

- **Agglomerative Clustering** має ще більший рівень змішування. Помітно, що майже кожен справжній клас розпорошений по багатьох кластерах. Це знижує інтерпретованість результату. Наприклад, клас сорочка (6) потрапляють майже в усі кластери, що ускладнює їх виділення.

- **EM (ручна реалізація)** формує сильніше виражені кластери, наприклад, 7-й (кросівки) і 8-й (сумки), але має проблеми з точністю, бо ча-

стина об'єктів класифікується неправильно через накладання гаусівських розподілів.

- **EM (sklearn)** - незважаючи на нижчу геометричну якість, класифікаційна структура дещо ближча до істинної. Деякі кластери чітко відповідають окремим класам, зокрема, 4-й кластер (пальто) здебільшого відповідає класу 8 (сумки), а тим часом 8-й кластер - класу 9 (чоботи), що може частково пояснити вищі значення ARI та Homogeneity.

Висновок: якщо важлива швидкість та компактність кластерів - найкраще підходить **K-Means**. Якщо ж ключовим є наближення до справжніх класів - доцільно використовувати **EM (sklearn)**.

Порівняння результатів кластеризації на датасеті DUO Salary (грудень 2024)

Для аналізу заробітних плат за грудень 2024 року з датасету DUO було застосовано три алгоритми кластеризації: K-Means, Agglomerative Clustering та EM (GMM). Якість кластеризації оцінювалась за внутрішніми метриками: Silhouette Score, Calinski-Harabasz Index та Davies-Bouldin Index (див. код в Додаток Г 4.6).

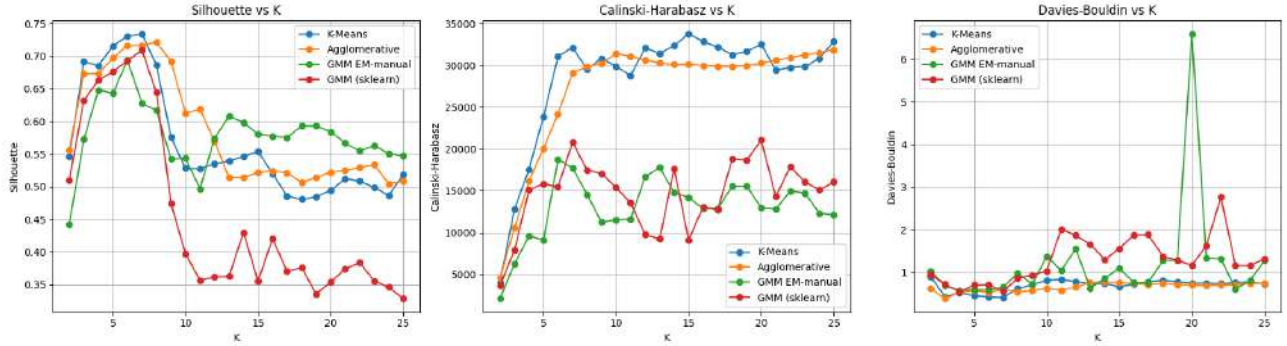

Рис. 7: Залежність метрик кластеризації від кількості кластерів K для різних алгоритмів. Зображено значення трьох метрик: Silhouette Score, Calinski-Harabasz Index, Davies-Bouldin Index.

На основі графіка (рис. 7) можна зробити такі спостереження:

- **Silhouette Score:** для всіх алгоритмів показник досягає максимуму в діапазоні $K = 5 - 8$, після чого починає спадати. Найвищі значення має **K-Means**, що вказує на чітке внутрішнє згрупування кластерів. У **GMM (sklearn)** та **ручного GMM** спостерігається різкий спад якості після $K > 9$.

- **Calinski-Harabasz Index (CH):** значення цього індексу зростають до певного моменту і стабілізуються. **K-Means** та **Agglomerative** демонструють стабільно вищі значення індексу CH, що вказує на хорошу глобальну відокремленість кластерів у порівнянні з GMM.

- **Davies-Bouldin Index (DBI):** чим нижче значення, тим краще. Найнижчі значення спостерігаються у **Agglomerative** та **K-Means**, особливо в діапазоні $K = 5 - 10$. **GMM (manual)** має сильні коливання і сплески DBI при $K > 15$, що вказує на нестабільну кластеризацію.

Для вибору кількості кластерів було обрано діапазон $K = 6 - 8$, у якому спостерігається оптимальне співвідношення між основними метриками якості кластеризації.

Табл. 4: Якісні метрики кластеризації для датасету DOU Salary при $K = 7$.

Метрика	**K-Means**	**Agglomerative**	**GMM**	**GMM (sklearn)**
Silhouette Score	**0.7342**	0.7166	0.6277	0.7095
CH Index	**32131.95**	29082.45	17711.89	20802.38
DBI	**0.4135**	0.6013	0.6562	0.5543
Час виконання	**0.01**	0.71	0.63	0.11

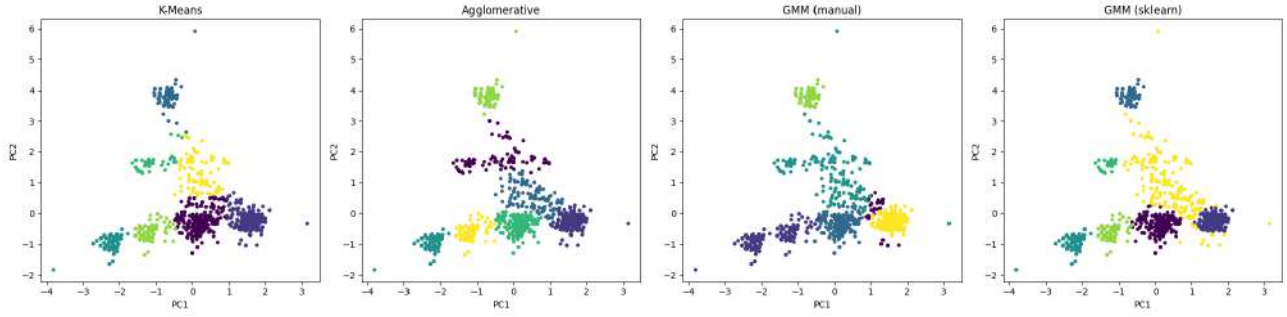

Рис. 8: Візуалізація результатів кластеризації на датасеті DUO Salary у просторі перших двох головних компонент (PC1 та PC2) при $K = 7$.

Порівняння алгоритмів проводилось для кількості кластерів $K = 6, 7, 8$. Було виявлено, що, найкращі результати були для $K = 7$. Найбільш стабільні результати в цьому діапазоні демонструють алгоритми **K-Means** та **Agglomerative**.

K-Means досяг найкращих результатів за всіма метриками: найвище значення Silhouette Score (0.7342), найвище CH Index (32131.95) та найнижчий Davies-Bouldin Index (0.4135), що свідчить про чітке розділення кластерів з високою щільністю всередині;

Agglomerative Clustering показав другу за якістю кластеризацію: високі значення Silhouette (0.7166) і CH Index (29082.45), але дещо гірший DBI (0.6013), що вказує на дещо більший рівень перекриття між кластерами;

GMM (manual) та **GMM (sklearn)** дали нижчі результати: Silhouette близько 0.63-0.71 і DBI від 0.55 до 0.65. Ймовірнісна природа цих моделей забезпечує гнучке кластеризування, але водночас - менш чітке порівняно з K-Means.

Таким чином, для кластеризації заробітних плат у грудні 2024 року:

- якщо головним пріоритетом є чітке розділення кластерів та їхня внутрішня однорідність - оптимальним вибором є **K-Means**;
- якщо необхідно врахувати ієрархічну будову даних або структуру підкластерів - доцільно застосувати **Agglomerative Clustering**;
- якщо важливо отримати ймовірнісну інтерпретацію кластерів або враховувати розмитість меж між групами, то варто використовувати **EM (GMM)**.

Висновок до розділу 4: K-Means виявився найстабільнішим і найефективнішим методом кластеризації як для зображень Fashion-MNIST, так і для табличних даних DOU salaries, демонструючи кращу узгодженість кластерів, високу чіткість поділу та ефективність за обчислювальними ресурсами.

Усі результати було підтверджено як числовими метриками (Silhouette, CH, DB і тд.), так і візуальними скатерплотами у просторі перших двох компонент PCA.

Висновки

Кваліфікаційна робота була присвячена GMM та EM-алгоритму. Було досліджено математичні та практичні аспекти моделей сумішей Гауса та EM-алгоритму. Побудовано теоретичний базис, який охоплює імовірнісну інтерпретацію, формалізм латентних змінних та принципи оцінювання параметрів методом максимізації математичного сподівання. Реалізовано власну версію EM-алгоритму, що пройшла перевірку на синтетичних та реальних даних.

У практичній частині було проведено аналіз на двох різних типах даних. Результати підтвердили доцільність застосування EM-алгоритму в задачах кластеризації, особливо коли важлива ймовірнісна інтерпретація. Водночас експерименти засвідчили, що класичні методи, як K-Means у багатьох випадках залишаються кращими за критеріями швидкості та геометричної якості кластерів.

Аналіз результатів на реальних даних показав, що:

- Для високо-структурованих, негаусівських даних (зображення Fashion-MNIST) більш ефективними є простіші геометричні методи, як K-Means.
- Для гаусоподібних або змішаних структур (у зарплатних даних) EM-алгоритм демонструє стабільні результати та коректну ймовірнісну інтерпретацію.
- Вибір алгоритму кластеризації має грунтуватися не лише на метриках якості, а й на природі даних та цілі аналізу.

Таким чином, поставлену мету та завдання дослідження було досягнуто повністю. Робота має як теоретичну, так і прикладну цінність, а результати можуть бути використані в задачах аналізу складних багатовимірних даних без наявиних міток.

Список літератури

Література

[1] Wu, Wei. (14 April 2024) *Unsupervised Learning* (PDF).
Доступний за посиланням: https://na.uni-tuebingen.de/ex/ml_seminar_ss2022/Unsupervised_Learning%20Final.pdf

[2] Oscar Contreras Carrasco *Gaussian Mixture Model Explained* 30 вересня 2024.
Доступний за посиланням: https://builtin.com/articles/gaussian-mixture-model

[3] Чорней Р. К. *Теорія ймовірностей і випадкові процеси.* Навчальний посібник. Київ, 2020 ст. 22-23 і ст.62-63.

[4] Карташов М.В. *Імовірність, процеси, статистика: Посібник.* КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА. Київ: Видавничо-поліграфічний центр «Київський університет», 2008. ст. 88

[5] Clayton Scott, Robert Nowak. *The Q-function*
Доступний за посиланням: https://web.archive.org/web/20120229030808/http://cnx.org/content/m11537/latest/

[6] Шевченко Г. *Лекції з теорії ймовірностей.* - Київ: КНУ, 3 листопада 2018 р.
Доступний за посиланням: https://probability.knu.ua/userfiles/zhoraster/probability.pdf

[7] Taboga, Marco (2021), PhD. *"Log-likelihood"*. StatLect, Lectures on probability theory and mathematical statistics. Kindle Direct Publishing.
Доступний за посиланням: https://www.statlect.com/glossary/log-likelihood

[8] Jonathan Pillow. *Lecture 18: Latent Variable Models and Expectation Maximization.* Statistical Modeling and Analysis of Neural Data, NEU 560, Princeton

University, Fall 3 листопада 2020.
Доступний за посиланням: https://pillowlab.princeton.edu/teaching/statneuro2020/notes/notes18_LatentVariableModels.pdf

[9] Фіхтенгольц Г. М. *Курс диференціального та інтегрального числення. Том 2.* Переклад українською: С. Зіновєв та інші. - 2022–2025.
Доступний за посиланням: https://nebayduzhi-math.azurewebsites.net/%D0%A4%D1%96%D1%85%D1%82%D0%B5%D0%BD%D0%B3%D0%BE%D0%BB%D1%8C%D1%86

[10] *Світлана Дрінь презентація до лекції з Big Data: "Вибір моделі, розширення та діагностика Національний університет "Києво-Могилянська академія"*, січень 2025, слайди 6–7.

[11] OECD AI. *Adjusted Rand Index (ARI).* OECD.AI Metrics Catalogue.
Доступно за посиланням: https://oecd.ai/en/catalogue/metrics/adjusted-rand-index-ari

[12] OECD AI. *Normalized Mutual Information (NMI).* OECD.AI Metrics Catalogue.
Доступно за посиланням: https://oecd.ai/en/catalogue/metrics/normalized-mutual-information-nmi

[13] Scikit-learn developers. *sklearn.metrics.homogeneity_score*, User Guide, Version 1.6.1.
Доступно за посиланням: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_score.html.

[14] *Hazal Gültekin. What is Silhouette Score?. Medium* 7 вересня 2023.
Доступний за посиланням: https://medium.com/@hazallgultekin/what-is-silhouette-score-f428fb39bf9a.

[15] GeeksforGeeks. *Calinski-Harabasz Index – Cluster Validity Indices | Set 3*, 25 квітня 2022.
Доступно за посиланням: https://www.geeksforgeeks.org/calinski-harabasz-index-cluster-validity-indices-set-3/

[16] GeeksforGeeks. *Davies-Bouldin Index*, 5 листопада 2023.
Доступно за посиланням: https://www.geeksforgeeks.org/davies-bouldin-index/

[17] Carl Edward Rasmussen. *The Expectation Maximization or EM algorithm* 18 листопада 2016.
Доступний за посиланням: https://mlg.eng.cam.ac.uk/teaching/4f13/1617/expectation%20maximization.pdf

[18] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. Розділ 9.4, ст. 451.

[19] Arshid. *Iris Flower Dataset Kaggle*.
Доступний за посиланням: https://www.kaggle.com/datasets/arshid/iris-flower-dataset

[20] dataset of Zalando. *Fashion MNIST* 2017.
Доступний за посиланням: https://www.kaggle.com/datasets/zalando-research/fashionmnist

[21] *DOU IT Salaries Dataset (грудень 2024)*.
Доступний за посиланням: https://github.com/devua/csv/tree/master/salaries

[22] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, Jonathan Taylor. *An Introduction to Statistical Learning with Applications in Python* 5 липня 2023.
Доступний за посиланням: https://www.statlearning.com/

[23] GeeksforGeeks. *K-Means Clustering - Introduction*. Доступний за посиланням: https://www.geeksforgeeks.org/k-means-clustering-introduction/

[24] *Scikit-learn K-means* Доступний за посиланням: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[25] Amit Ranjan. *Hierarchical Clustering (Agglomerative)* 30 листопада 2020. Доступний за посиланням: https://medium.com/analytics-vidhya/hierarchical-clustering-agglomerative-f6906d440981

[26] Abubakar Auwal Khalid. *Agglomerative Hierarchical Clustering: A Study and Implementation in Python* 5 липня 2023. Доступний за посиланням: https://medium.com/@khalidassalafy/agglomerative-hierarchical-clustering-a-study-and-implementation-in-pyth

[27] Scikit-learn Agglomerative Clustering. Доступний за посиланням: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

[28] GeeksforGeeks. *Understanding the Confusion Matrix in Machine Learning.* Доступний за посиланням: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/.

Додатки

Додаток А. Реалізація EM-алгоритму на Python

```python
import numpy as np
from scipy.stats import multivariate_normal

def initialize_parameters(X, K, seed=None):
    np.random.seed(seed)
    n_samples, n_features = X.shape
    pi = np.full(K, 1 / K)
    mu = X[np.random.choice(n_samples, K, replace=False)]
    sigma = [np.cov(X.T) + np.eye(n_features) * 1e-6 for _ in range(K)]
    return pi, mu, sigma

def e_step(X, pi, mu, sigma):
    K = len(pi)
    n = X.shape[0]
    gamma = np.zeros((n, K))

    for k in range(K):
        gamma[:, k] = pi[k] * multivariate_normal.pdf(X, mean=mu[k], cov=
            sigma[k], allow_singular=True)

    gamma_sum = np.sum(gamma, axis=1, keepdims=True)
    gamma = np.divide(gamma, gamma_sum, where=gamma_sum!=0)  # avoid NaNs
    return gamma

def m_step(X, gamma):
    n_samples, n_features = X.shape
    K = gamma.shape[1]

    N_k = gamma.sum(axis=0)
    pi = N_k / n_samples
    mu = np.dot(gamma.T, X) / N_k[:, np.newaxis]
    sigma = []

    for k in range(K):
        diff = X - mu[k]
        weighted_cov = np.dot((gamma[:, k][:, np.newaxis] * diff).T, diff) /
            N_k[k]
        sigma.append(weighted_cov + np.eye(n_features) * 1e-6)  #
            regularization

    return pi, mu, sigma

```

```
def compute_log_likelihood(X, pi, mu, sigma):
    n_samples = X.shape[0]
    K = len(pi)
    likelihood = np.zeros((n_samples, K))

    for k in range(K):
        likelihood[:, k] = pi[k] * multivariate_normal.pdf(X, mean=mu[k],
            cov=sigma[k], allow_singular=True)

    return np.sum(np.log(np.sum(likelihood, axis=1) + 1e-10))  # avoid log
        (0)

def em_gmm(X, K, max_iter=100, tol=1e-4, verbose=False, seed=None):
    pi, mu, sigma = initialize_parameters(X, K, seed)
    log_likelihood = -np.inf

    for i in range(max_iter):
        gamma = e_step(X, pi, mu, sigma)
        pi, mu, sigma = m_step(X, gamma)

        new_log_likelihood = compute_log_likelihood(X, pi, mu, sigma)

        if verbose and (i == 0 or i == max_iter - 1 or np.abs(
            new_log_likelihood - log_likelihood) < tol):
            print(f"Iteration {i+1}, Log-likelihood: {new_log_likelihood:.4f
                }")

        if np.abs(new_log_likelihood - log_likelihood) < tol:
            break

        log_likelihood = new_log_likelihood

    return pi, mu, sigma, gamma
```

Демонстрація кластеризації за допомогою EM-алгоритму на синтетичних даних.

```
import matplotlib.pyplot as plt

np.random.seed(42)
n_samples = 300
X1 = np.random.multivariate_normal([0, 0], np.eye(2), n_samples)
X2 = np.random.multivariate_normal([5, 5], np.eye(2), n_samples)
X = np.vstack((X1, X2))

K = 2
pi, mu, sigma, gamma = em_gmm(X, K=K, max_iter=100, tol=1e-4, verbose=True,
```

```
    seed=42)

clusters = np.argmax(gamma, axis=1)

plt.scatter(X[:, 0], X[:, 1], c=clusters, cmap='coolwarm', s=30)
plt.scatter(mu[:, 0], mu[:, 1], c='orange', marker='*', s=100, label='
                                    ')
plt.title("EM
                                                          ")
plt.legend()
plt.show()
```

Оцінка якості кластеризації GMM за такими метриками: AIC, BIC, ARI, NMI, Homogeneity, Silhouette, Calinski-Harabasz та Davies-Bouldin.

```
from sklearn.metrics import adjusted_rand_score,
    normalized_mutual_info_score, homogeneity_score, silhouette_score,
    calinski_harabasz_score, davies_bouldin_score

def evaluate_gmm(X, y_true, pi, mu, sigma, gamma):
    y_pred = np.argmax(gamma, axis=1)
    n_clusters = gamma.shape[1]
    n_samples, n_features = X.shape

    log_likelihood = compute_log_likelihood(X, pi, mu, sigma)

    d = n_features
    K = n_clusters
    n_params = K * d + K * d * (d + 1) / 2 + (K - 1)
    aic = 2 * n_params - 2 * log_likelihood
    bic = np.log(n_samples) * n_params - 2 * log_likelihood

    ari = adjusted_rand_score(y_true, y_pred)
    nmi = normalized_mutual_info_score(y_true, y_pred)
    homogeneity = homogeneity_score(y_true, y_pred)

    silhouette = silhouette_score(X, y_pred)
    ch_score = calinski_harabasz_score(X, y_pred)
    db_score = davies_bouldin_score(X, y_pred)

    print(f"  Log-likelihood: {log_likelihood:.4f}")
    print(f"  AIC: {aic:.2f}")
    print(f"  BIC: {bic:.2f}")
    print(f"  Adjusted Rand Index: {ari:.4f}")
    print(f"  Normalized Mutual Info: {nmi:.4f}")
    print(f"  Homogeneity Score: {homogeneity:.4f}")
    print(f"  Silhouette Score: {silhouette:.4f}")
```

```
    print(f"  Calinski-Harabasz: {ch_score:.2f}")
    print(f"  Davies-Bouldin Index: {db_score:.4f}")
```

Цикл кластеризації методом Gaussian Mixture Model (EM-алгоритм) для трьох варіантів кількості кластерів: K = 2, 3, 4 на датасеті Iris.

```
from sklearn.datasets import load_iris
import numpy as np
import matplotlib.pyplot as plt

iris = load_iris()
X_iris = iris.data[:, :2]
y_iris = iris.target

for K in [2, 3, 4]:
    print(f"\n------------                                                    K = {K}
        ------------")

    pi_iris, mu_iris, sigma_iris, gamma_iris = em_gmm(X_iris, K=K, max_iter
        =100, tol=1e-4, verbose=True, seed=42)
    clusters_iris = np.argmax(gamma_iris, axis=1)

    plt.figure(figsize=(8, 6))
    plt.scatter(X_iris[:, 0], X_iris[:, 1], c=clusters_iris, cmap='viridis',
         s=40)
    plt.scatter(mu_iris[:, 0], mu_iris[:, 1], c='red', marker='*', s=120,
        label='                                  ')
    plt.title(f"EM                                Iris (K = {K})")
    plt.xlabel("Sepal length")
    plt.ylabel("Sepal width")
    plt.legend()
    plt.show()

    print("              GMM                                :")
    evaluate_gmm(X_iris, y_iris, pi_iris, mu_iris, sigma_iris, gamma_iris)
```

Додаток Б. Реалізація K-Means, Agglomerative Clustering для реалізації обчислення метрик кластеризації на Python

Оцінка якості кластеризації моделями: K-Means, Agglomerative Clustering, EM-алгоритм (ручна реалізація), EM-алгоритм з sklearn. Повертає таблицю з метриками та будує візуалізацію й матрицю неточностей.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.mixture import GaussianMixture
from sklearn.metrics import (
    adjusted_rand_score, normalized_mutual_info_score, silhouette_score,
    calinski_harabasz_score, davies_bouldin_score, homogeneity_score,
    confusion_matrix
)
from sklearn.decomposition import PCA
import seaborn as sns
import time

def evaluate_clustering_models_mnist(X, y_true, n_clusters=10, visualize=
    True,
                           save_csv_path=None, show_conf_matrix=True):
    results = []
    labels_dict = {}
    timings = {}

    models = {
        "K-Means": KMeans(n_clusters=n_clusters, random_state=42),
        "Agglomerative": AgglomerativeClustering(n_clusters=n_clusters,
            linkage='ward'),
        "EM (manual)": manual_em_wrapper,
        "EM (sklearn)": GaussianMixture(n_components=n_clusters,
            random_state=42)
    }

    for name, model_func in models.items():
        if model_func is None:
            continue

        start_time = time.time()

        if callable(model_func):
          labels = model_func(X, n_clusters)
        else:
          labels = model_func.fit_predict(X)

        elapsed_time = time.time() - start_time
        timings[name] = elapsed_time
        labels_dict[name] = labels

        ari = adjusted_rand_score(y_true, labels)
```

```
        nmi = normalized_mutual_info_score(y_true, labels)
        homo = homogeneity_score(y_true, labels)
        sil = silhouette_score(X, labels)
        ch = calinski_harabasz_score(X, labels)
        db = davies_bouldin_score(X, labels)

        results.append({
            "Model": name,
            "ARI": round(ari, 5),
            "NMI": round(nmi, 5),
            "Homogeneity": round(homo, 5),
            "Silhouette": round(sil, 5),
            "Calinski-Harabasz": round(ch, 5),
            "Davies-Bouldin": round(db, 5),
            "Time (s)": round(elapsed_time, 5)
        })

    df_results = pd.DataFrame(results).set_index("Model")
    print(df_results)

    if visualize:
        pca_vis = PCA(n_components=2).fit_transform(X)
        fig, axes = plt.subplots(1, len(labels_dict), figsize=(5 * len(
            labels_dict), 5))
        if len(labels_dict) == 1:
            axes = [axes]

        for i, (name, labels) in enumerate(labels_dict.items()):
            axes[i].scatter(
                pca_vis[:, 0],
                pca_vis[:, 1],
                c=labels,
                s=10,
                cmap='viridis'
            )
            axes[i].set_title(name)
            axes[i].set_xlabel("PC1")
            axes[i].set_ylabel("PC2")
        plt.tight_layout()
        plt.show()


    if show_conf_matrix:
        for name, labels in labels_dict.items():
            cm = confusion_matrix(y_true, labels)
            plt.figure(figsize=(6, 4))
```

```
            sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
            plt.title(f"Confusion Matrix     {name}")
            plt.xlabel("Predicted")
            plt.ylabel("True")
            plt.show()

    return df_results
```

Додаток В. Порівняння кластеризаційних алгоритмів за якісними метриками для датасету Fashion MNIST. Візуалізація результатів кластеризаці на Python

Покращена версія $\mathrm{em}_{g}mm_{B}igi4.6.manual_{e}m_{w}rapperiiiargmaxii(gamma)$.

```

import numpy as np
from scipy.stats import multivariate_normal

def initialize_parameters(X, K, seed=None):
    np.random.seed(seed)
    n_samples, n_features = X.shape
    pi = np.full(K, 1 / K)
    mu = X[np.random.choice(n_samples, K, replace=False)]
    sigma = [np.cov(X.T) + np.eye(n_features) * 1e-2 for _ in range(K)]
    return pi, mu, sigma

def e_step(X, pi, mu, sigma):
    K = len(pi)
    n = X.shape[0]
    gamma = np.zeros((n, K))

    for k in range(K):
        gamma[:, k] = pi[k] * multivariate_normal.pdf(
            X, mean=mu[k], cov=sigma[k], allow_singular=True
        )

    gamma_sum = np.sum(gamma, axis=1, keepdims=True)
    gamma = np.divide(gamma, gamma_sum, where=gamma_sum!=0)
    return gamma

def m_step(X, gamma):
    n_samples, n_features = X.shape
    K = gamma.shape[1]
    N_k = gamma.sum(axis=0)
```

```
    pi = N_k / n_samples
    mu = np.dot(gamma.T, X) / N_k[:, np.newaxis]
    sigma = []

    for k in range(K):
        diff = X - mu[k]
        weighted_cov = np.dot((gamma[:, k][:, np.newaxis] * diff).T, diff) /
            N_k[k]

        cov_k = weighted_cov + np.eye(n_features) * 1e-2
        cov_k = (cov_k + cov_k.T) / 2
        sigma.append(cov_k)

    return pi, mu, sigma

def compute_log_likelihood(X, pi, mu, sigma):
    n_samples = X.shape[0]
    K = len(pi)
    likelihood = np.zeros((n_samples, K))

    for k in range(K):
        likelihood[:, k] = pi[k] * multivariate_normal.pdf(
            X, mean=mu[k], cov=sigma[k], allow_singular=True
        )

    return np.sum(np.log(np.sum(likelihood, axis=1) + 1e-10))

def em_gmm_Big(X, K, max_iter=100, tol=1e-4, verbose=False, seed=None):
    pi, mu, sigma = initialize_parameters(X, K, seed)
    log_likelihood = -np.inf

    for i in range(max_iter):
        gamma = e_step(X, pi, mu, sigma)
        pi, mu, sigma = m_step(X, gamma)
        new_log_likelihood = compute_log_likelihood(X, pi, mu, sigma)

        if verbose:
            print(f"Iteration {i+1}, Log-likelihood: {new_log_likelihood:.4f
                }")
        if np.abs(new_log_likelihood - log_likelihood) < tol:
            break

        log_likelihood = new_log_likelihood

    return pi, mu, sigma, gamma

```

```
75
76 def manual_em_wrapper(X, n_clusters):
77     pi, mu, sigma, gamma = em_gmm_Big(X, K=n_clusters, max_iter=100, tol=1e
       -4, verbose=False, seed=42)
78     return np.argmax(gamma, axis=1)
```

Завантаження даних Fashion MNIST, їх обробка, використання кластеризація трьома методами та обчислення основних метрик.

```
1  import numpy as np
2  import pandas as pd
3  from sklearn.decomposition import PCA
4  from sklearn.preprocessing import StandardScaler
5  from tensorflow.keras.datasets import fashion_mnist
6
7  (X_train, y_train), (_, _) = fashion_mnist.load_data()
8  X_flat = X_train.reshape(X_train.shape[0], -1)
9
10 X_scaled = StandardScaler().fit_transform(X_flat)
11
12 pca = PCA(n_components=50)
13 X_pca = pca.fit_transform(X_scaled)
14
15 df_pca = pd.DataFrame(X_pca)
16 df_pca["label"] = y_train
17
18 df_sampled = (
19     df_pca.groupby("label", group_keys=False)
20     .apply(lambda x: x.sample(n=100, random_state=42), include_groups=False)
21     .reset_index(drop=True)
22 )
23
24 df_sampled["label"] = df_pca.loc[df_sampled.index, "label"].values
25
26 X = df_sampled.drop("label", axis=1).values
27 y_true = df_sampled["label"].values
28
29 results_df = evaluate_clustering_models_mnist(X, y_true, n_clusters=10)
```

Додаток Г. Порівняння кластеризаційних алгоритмів за якісними метриками для датасету DUO Salary (грудень 2024). Візуалізація результатів кластеризаці на Python

Завантаження даних DUO Salary, їх обробка, розділення на 2 компоненти та збереження в новий csv. файл.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

url = "https://raw.githubusercontent.com/devua/csv/master/salaries/2024
    _dec_raw.csv"
df = pd.read_csv(url)
```

Для зручності подальшої обробки даних було виконано перейменування колонок у датафреймі за допомогою методу df.rename(), наприклад, 'В якій області ви зараз живете?': 'region'. Через особливості LaTeX деякі символи, пробіли та переноси рядків у початкових назвах в датасеті не відображаються коректно при вставці коду. Тому код із перейменуванням колонок подано нижче в узагальненому вигляді:

```
df.rename(columns={...}, inplace=True)
```

```
selected_columns = ['specialization', 'category', 'main_language', 'region']

df_cleaned = df.dropna(subset=selected_columns).copy()
df_encoded = pd.get_dummies(df_cleaned[selected_columns])

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_encoded)

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

df_cleaned.loc[:, 'PC1'] = X_pca[:, 0]
df_cleaned.loc[:, 'PC2'] = X_pca[:, 1]

mask = X_pca[:, 1] > -20
df_cleaned = df_cleaned[mask]

pca_components = pd.DataFrame(
    pca.components_,
    columns=df_encoded.columns,
```

```
    index=['PC1', 'PC2']
).T.sort_values(by='PC1', ascending=False)

df_cleaned.to_csv("dou_salaries_cleaned.csv", index=False)
```

Візуальне визначення оптимальне значення K для подальшого аналізу.

```
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score, calinski_harabasz_score,
    davies_bouldin_score
import matplotlib.pyplot as plt
import numpy as np

def plot_all_models_metrics_vs_k(X, k_min=2, k_max=15):
    K_range = range(k_min, k_max + 1)

    models = {
        "K-Means": lambda k: KMeans(n_clusters=k, random_state=42),
        "Agglomerative": lambda k: AgglomerativeClustering(n_clusters=k,
            linkage='ward'),
        "GMM EM-manual": lambda k: lambda X: manual_em_wrapper(X, k),
        "GMM (sklearn)": lambda k: GaussianMixture(n_components=k,
            random_state=42)
    }

    metrics = {
        "Silhouette": [],
        "Calinski-Harabasz": [],
        "Davies-Bouldin": []
    }

    for model_name, model_func in models.items():
        silhouette_scores = []
        ch_scores = []
        db_scores = []

        for k in K_range:
            model = model_func(k)
            try:
                if model_name == "GMM EM-manual":
                    labels = model(X)
                elif model_name == "GMM (sklearn)":
                    labels = model.fit(X).predict(X)
                else:
                    labels = model.fit_predict(X)

```

```
                silhouette_scores.append(silhouette_score(X, labels))
                ch_scores.append(calinski_harabasz_score(X, labels))
                db_scores.append(davies_bouldin_score(X, labels))

            except Exception as e:
                print(f"[{model_name} - K={k}] Error: {e}")
                silhouette_scores.append(np.nan)
                ch_scores.append(np.nan)
                db_scores.append(np.nan)

        metrics["Silhouette"].append((model_name, silhouette_scores))
        metrics["Calinski-Harabasz"].append((model_name, ch_scores))
        metrics["Davies-Bouldin"].append((model_name, db_scores))

    fig, axes = plt.subplots(1, 3, figsize=(18, 5))

    for idx, (metric_name, results) in enumerate(metrics.items()):
        ax = axes[idx]
        for model_name, scores in results:
            ax.plot(K_range, scores, marker='o', label=model_name)
        ax.set_title(f"{metric_name} vs K")
        ax.set_xlabel("K")
        ax.set_ylabel(metric_name)
        ax.legend()
        ax.grid(True)

    plt.tight_layout()
    plt.show()

X_dou = df_cleaned[['PC1', 'PC2']].values
plot_all_models_metrics_vs_k(X_dou, k_min=2, k_max=25)
```

Оцінка кластеризації без наявних міток.

```
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score, calinski_harabasz_score,
    davies_bouldin_score
import matplotlib.pyplot as plt
import pandas as pd
import time

def evaluate_unsupervised_clustering(X, n_clusters=10, visualize=True):

    models = {
        "K-Means": KMeans(n_clusters=n_clusters, random_state=42),
        "Agglomerative": AgglomerativeClustering(n_clusters=n_clusters,
```

```
        linkage='ward'),
    "GMM (manual)": manual_em_wrapper,
    "GMM (sklearn)": GaussianMixture(n_components=n_clusters,
        random_state=42)
}

results = []
labels_dict = {}
timings = {}

for name, model in models.items():
    start = time.time()
    if callable(model):
        labels = model(X, n_clusters)
    else:
        if name == "GMM (sklearn)":
            labels = model.fit(X).predict(X)
        else:
            labels = model.fit_predict(X)
    elapsed = time.time() - start

    silhouette = silhouette_score(X, labels)
    ch = calinski_harabasz_score(X, labels)
    db = davies_bouldin_score(X, labels)

    results.append({
        "Model": name,
        "Silhouette": round(silhouette, 4),
        "Calinski-Harabasz": round(ch, 2),
        "Davies-Bouldin": round(db, 4),
        "Time (s)": round(elapsed, 2)
    })

    labels_dict[name] = labels

df_results = pd.DataFrame(results).set_index("Model")
print(df_results)

if visualize:
    fig, axes = plt.subplots(1, len(labels_dict), figsize=(5 * len(
        labels_dict), 5))
    if len(labels_dict) == 1:
        axes = [axes]
    for i, (name, labels) in enumerate(labels_dict.items()):
        axes[i].scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s
            =10)
```

```
            axes[i].set_title(name)
            axes[i].set_xlabel("PC1")
            axes[i].set_ylabel("PC2")
        plt.tight_layout()
        plt.show()

    return df_results
```

Знаходження найкращого K по метрикам та вивід кластерів

```
all_results = []

for k in range(6, 9):
    print(f"\n-------------                                                    : K = {k}
        ------------- ")
    df_result = evaluate_unsupervised_clustering(X_dou, n_clusters=k,
        visualize=True)
    df_result["K"] = k
    all_results.append(df_result)

results_df_dou = pd.concat(all_results)
results_df_dou = results_df_dou.reset_index().set_index(["K", "Model"])
print("\ n                                                                :")
print(results_df_dou)
```