КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю 121 Інженерія програмного забезпечення
на тему:

МЕТОДИ НЕГЛАДКОЇ ОПТИМІЗАЦІЇ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ КЛАСТЕРИЗАЦІЇ

Виконала студентка 4-го курсу
Софія ЦУБІН _____
(підпис)

Науковий керівник:
асистент, PhD з прикладної математики
Віктор СТОВБА _____
(підпис)

Засвідчую, що в цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Роботу розглянуто й допущено до захисту на засідання кафедри інтелектуальних програмних систем
«_25_» _травня_____ 2022 р.,
протокол № 10
Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи 54 сторінки, 24 ілюстрації, 5 таблиць, 26 джерел посилання.

Ключові слова: КЛАСТЕРИЗАЦІЯ, МЕТОД ЕЛІПСОЇДІВ, МЕТОД ШТРАФІВ, ОПУКЛІСТЬ, СУБГРАДІЄНТ, ЯРУЖНІСТЬ, K-MEANS, P-MEDIANS, R-АЛГОРИТМ.

Об’єктом дослідження є задача кластеризації, поставлена у вигляді оптимізаційної задачі задля розв’язання за допомогою субградієнтних методів.

Метою кваліфікаційної роботи є дослідження придатності методів негладкої оптимізації для вирішення задачі кластеризації.

Мовою для імплементації досліджуваних алгоритмів і цільових функцій було обрано Python. Використане середовище розробки — Google Colab. Були використані бібліотека для наукових обчислень NumPy, бібліотека Matplotlib для візуалізації графіків, бібліотека scikit-learn для порівняння з аналогами.

В роботі показано, що метод еліпсоїдів та r-алгоритм придатні для розв’язання поставленої задачі, оскільки цільові функції є квазіопуклими. Однак через порушення строгої опуклості і яружність функцій робота алгоритмів не завжди є максимально ефективною.

Застосування методів негладкої оптимізації представляє інтерес в контексті задач машинного навчання, навчання нейронних мереж, глибокому навчанні, а також певний науковий інтерес. Результати цієї роботи можуть стати в нагоді при вирішенні інших задач з цієї галузі, де постає необхідність мінімізувати негладкі і не строго опуклі функції.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта досліджень. Після кількох десятиліть галузь машинного навчання не перестає розвиватися. Перед науковцями весь час постає величезна кількість різних задач, великих і малих, кількісних і якісних, через що необхідним є створення цілої низки різноманітних методів аналізу та обробки даних. Зокрема, було запропоновано кілька підходів до розв’язання задачі кластеризації, таких як ієрархічна кластеризація, оптимізаційна кластеризація, кластеризація на основі щільності розподілу даних. Дослідження щодо покращення якості та ефективності роботи цих алгоритмів тривають і нині.

Актуальність роботи та підстави для її виконання. Кластерний аналіз — це типова задача машинного навчання, з якою дослідники та інші фахівці стикаються щоденно. Можливість отримання нових результатів для задачі кластеризації, а також дослідження роботи субградієнтних алгоритмів з квазіопуклими яружними функціями зумовлюють актуальність цієї роботи.

Мета й завдання роботи. Метою кваліфікаційної роботи є дослідження придатності методів негладкої оптимізації для вирішення задачі кластеризації.

Для досягнення поставленої мети було виконано такі завдання:

– сформулювати задачу кластеризації як оптимізаційну;

– розв’язати методом еліпсоїдів;

– розв’язати за допомогою r-алгоритму;

– імплементувати алгоритми мовою Python;

– порівняти з класичними бібліотечними реалізаціями.

Об’єкт, методи й засоби дослідження. Об’єктом дослідження є задача кластеризації, поставлена у вигляді оптимізаційної задачі задля розв’язання за допомогою субградієнтних методів.

Мовою для імплементації досліджуваних алгоритмів і цільових функцій було обрано Python, бо ця мова пропонує широкий набір зручних інструментів для

розв’язання задач машинного навчання. Використане середовище розробки — Google Colab.

Можливі сфери застосування. Застосування методів негладкої оптимізації представляє інтерес в контексті багатьох задач машинного навчання, наприклад при навчанні нейронних мереж чи глибокому навчанні, де можуть зустрічатися схожі на розглянуті у цій роботі цільові функції. Часто такі функції не є диференційовними чи хоча б опуклими, і як показано в цій роботі, навіть в таких випадках можливо підібрати достатньо ефективний оптимізаційний метод.

1 ОСНОВНІ МЕТОДИ КЛАСТЕРИЗАЦІЇ. ПОНЯТТЯ СУБГРАДІЄНТА

1.1 Кластерний аналіз

Останніми десятиліттями людство приділяє велику увагу обробці і аналізу даних. Це дозволяє вченим робити нові висновки, створювати і перевіряти нові гіпотези, а бізнес-сектору — покращувати свої послуги і приваблювати більше клієнтів.

Завдяки розвитку сучасних технологій з’явилася можливість збирати і зберігати небачений досі об’єм інформації абсолютно різного характеру з багатьох різних сфер. Вручну обробити такі масиви даних стало просто неможливо, тому наступним закономірним кроком стала розробка методів для автоматизації цього процесу. Людство занурилося в концепт штучного інтелекту, що має формалізувати і вирішувати задачі, притаманні людині, такі як розпізнавання образів, природної мови, генерація осмислених текстів, покращення результатів роботи алгоритмів, що вже існують тощо.

Машинне навчання — розділ штучного інтелекту, наука, що надає комп’ютерам можливість вчитись без необхідності бути явно на це запрограмованими.

Машинне навчання здатне вирішувати широке коло різноманітних задач. Системи машинного навчання можна поділити на кілька великих категорій за низкою параметрів:

а) чи необхідний вчитель для тренування: системи навчання зі вчителем, без учителя, змішані та навчання з підкріпленням;

б) динамічність навчання, тобто чи можливо навчання “на льоту” на противагу так званому batch learning;

в) чи порівнюють вони нові дані з вже відомими, чи виділяють шаблони в навчальній вибірці і будують передбачувальну модель, прямо як вчені (так звані instance-based та model-based навчання) [1].

У цій роботі розглядатиметься один з розділів машинного навчання, що базується на навчанні без учителя, тобто вхідні дані не мають жодних класифікацій. Власне, нашим завданням і є створити цю класифікацію.

Кластерний аналіз або кластеризація — розділ машинного навчання, який вирішує задачу групування набору об’єктів таким чином, щоб об’єкти в одній групі (так званому кластері) за певними параметрами були більш схожі один на одного, ніж на об’єкти в інших групах (кластерах)[2].

Кластеризація знайшла застосування в багатьох галузях людської діяльності. Так, кластерний аналіз став невід’ємним інструментом в маркетингових дослідженнях, категоризації товарів, аналізі соціальних мереж. Він застосовується для сегментації зображень, еволюційних алгоритмів, виявлення аномалій. Навіть в біології, хімії, соціології і багатьох інших науках кластеризація допомагає виявляти групи генів і топологічних індексів речовин й проводити кримінальний аналіз тощо.

1.2 Підходи до розв'язання задачі кластеризації

Існує велика кількість алгоритмів, що вирішують задачу кластеризації. Різні алгоритми дають різні результати на одних і тих самих даних через різні підходи до визначення положення кластерів. Для певних даних деякі алгоритми непридатні, бо їхні припущення щодо форми кластерів не збігаються з реальним характером розподілу точок. Є підходи, які візуально дозволяють розділити масив даних, однак не визначають кластери математично, залишаючи це завдання досліднику.

1.2.1 Графічне виявлення кластерів

Наявність кластерів в спостережуваних даних часто може бути виявлена за допомогою простих графічних представлень даних, можливо, із застосуванням

відповідних оцінок щільності [3]. Так, популярними засобами є гістограми та точкові діаграми.

Гістограми можуть бути корисним першим кроком для попереднього аналізу даних. Так, наприклад, дивлячись на гістограму (рис. 1.1), дослідник може явно бачити два піки, що наводить на думку про існування двох відносно чітко окреслених груп даних.

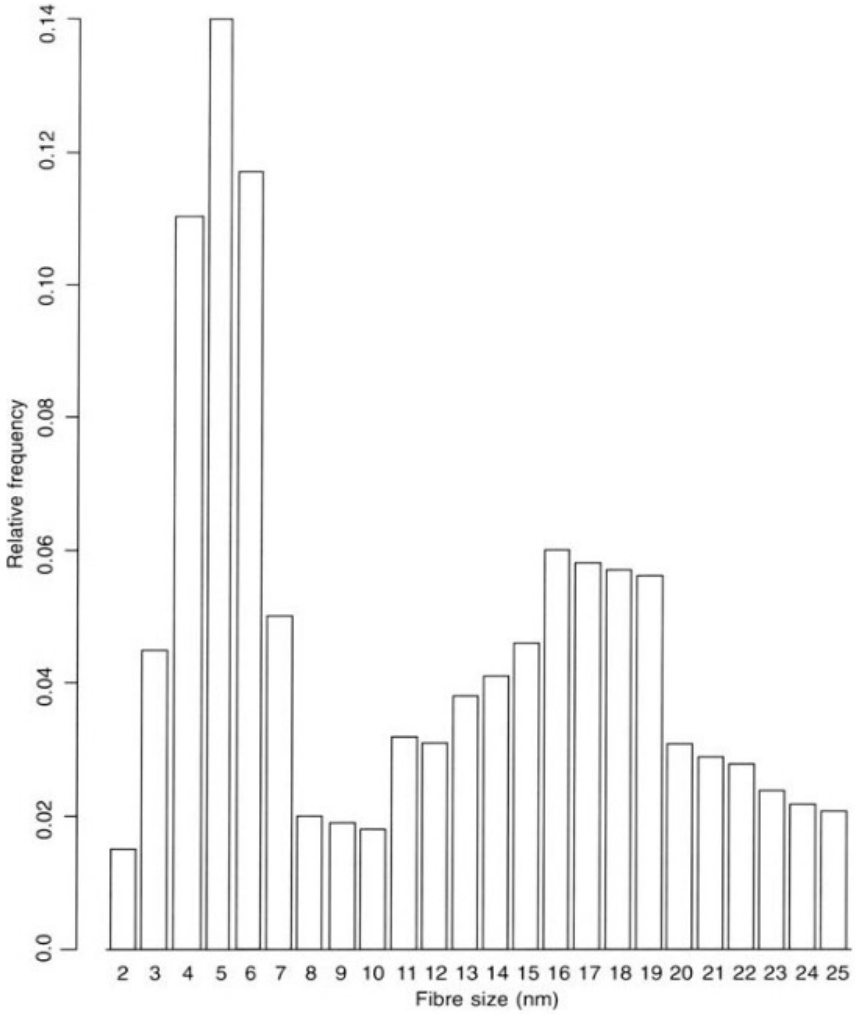


Рисунок 1.1 — Гістограма розмірів волокон мієлінізованого попереково-крижового вентрального корінця кошенят фіксованого віку

Розгляньмо також гістограму швидкостей вісімдесяти двох галактик із шести добре розділених конічних ділянок космосу. Аналіз цих даних має пролити світло на те, чи є в спостережуваному Всесвіті суперкластери галактик, оточені великими порожнечами. Доказом існування таких суперкластерів була б

мультимодальность розподілу швидкостей. Наведена гістограма демонструє наявність такої мультимодальності – див. рис. 1.2.

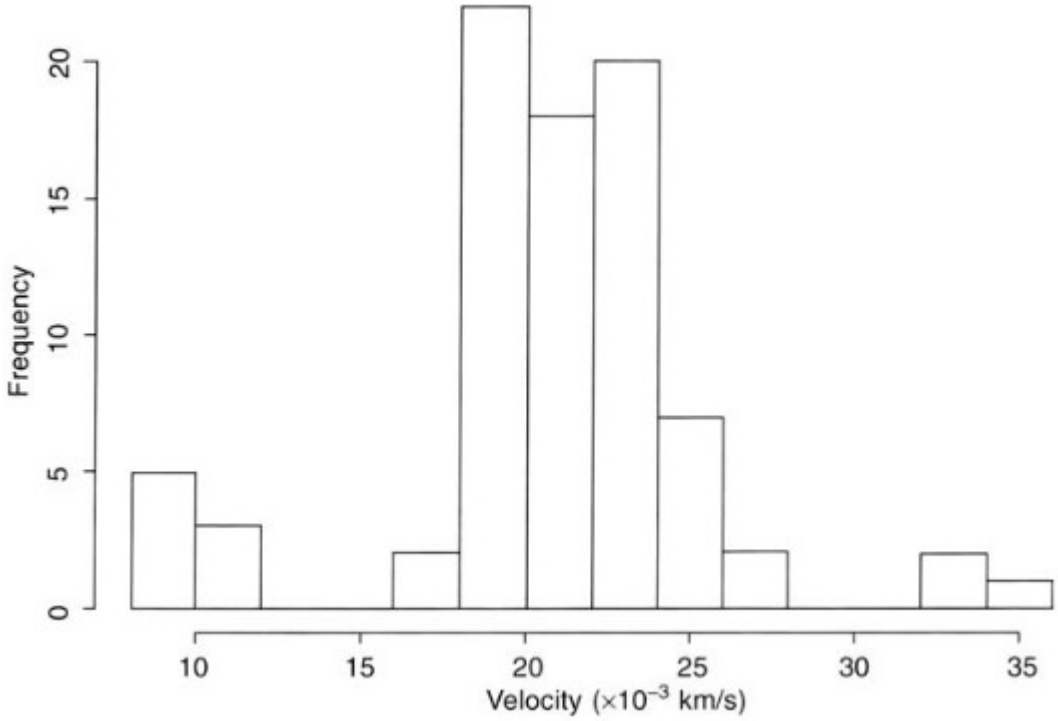


Рисунок 1.2 — Гістограма швидкостей 82 галактик

Точкові діаграми представляють дані у простому для сприйняття двовимірному *xy*-вигляді. Цей метод застосовувався як мінімум з вісімнадцятого століття. Подивившись на такі графіки, людина здатна з першого погляду виділити угрупування точок, як наприклад на рис. 1.3.

Однак часто зустрічаються випадки, коли не всі кластери одразу видно неозброєним оком. Через неоднорідну або надмірну щільність розподілу даних два або більше кластерів можуть візуально зливатися в один, що заважає об’єктивно оцінити структуру спостережуваних даних (рис. 1.4).

В таких випадках на допомогу приходить чисельна оцінка щільності даних. Існує багато різноманітних оцінок, що володіють різними властивостями і використовуються для багатьох типів даних. Ця тема висвітлюється у численних працях, наприклад, таких як [4], [5].

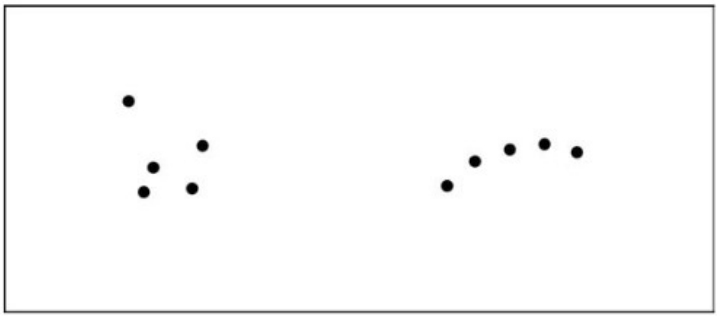


Рисунок 1.3 — Точкова діаграма з двома чіткими групами точок

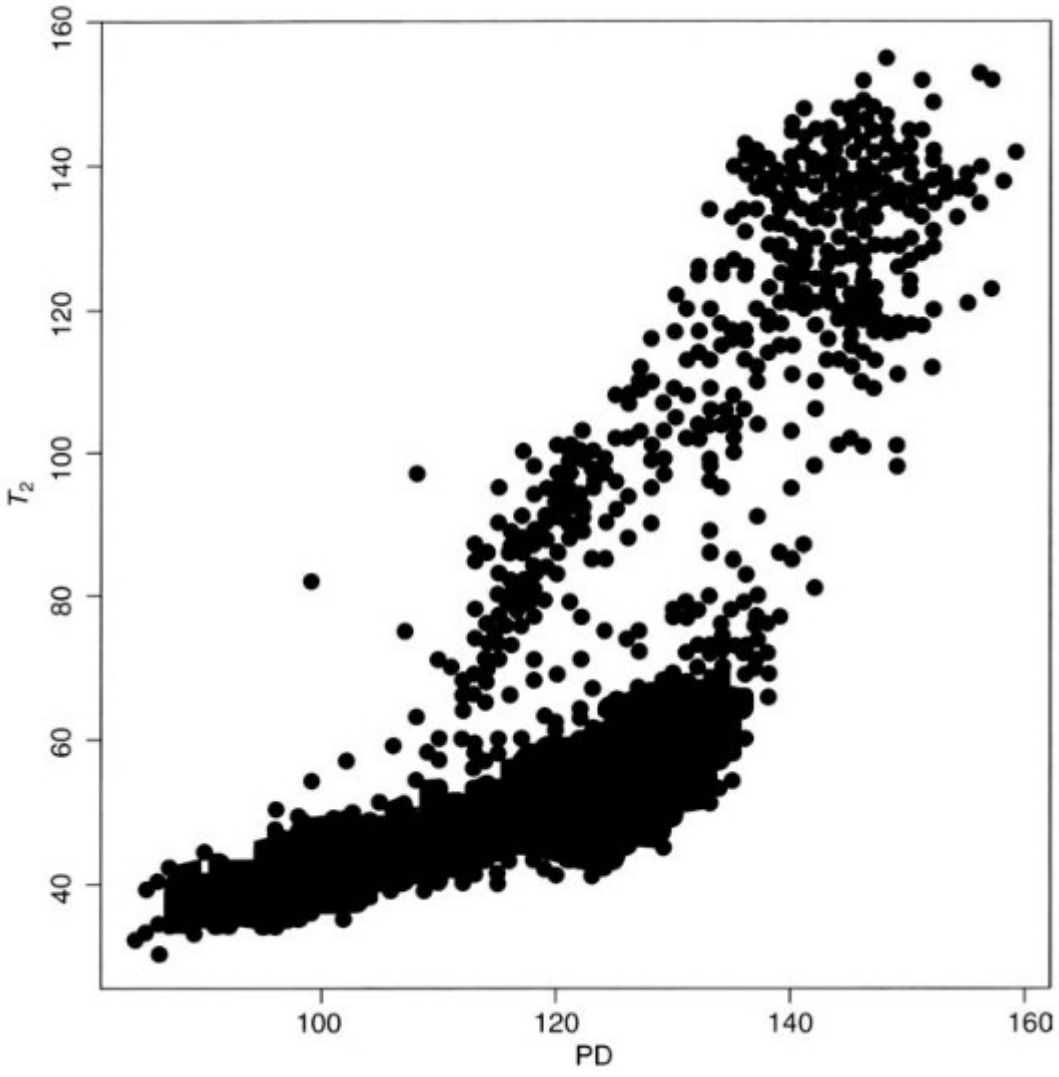


Рисунок 1.4 — Точкова діаграма 2836 вокселів магнітно-резонансної томографії (МРТ)

Застосувавши двовимірну оцінку щільності до даних, точкова діаграма яких наведена на рис. 1.4, можемо бачити, що насправді є аж три чітко виділені кластери, а не два, як здавалось на перший погляд (рис. 1.5).

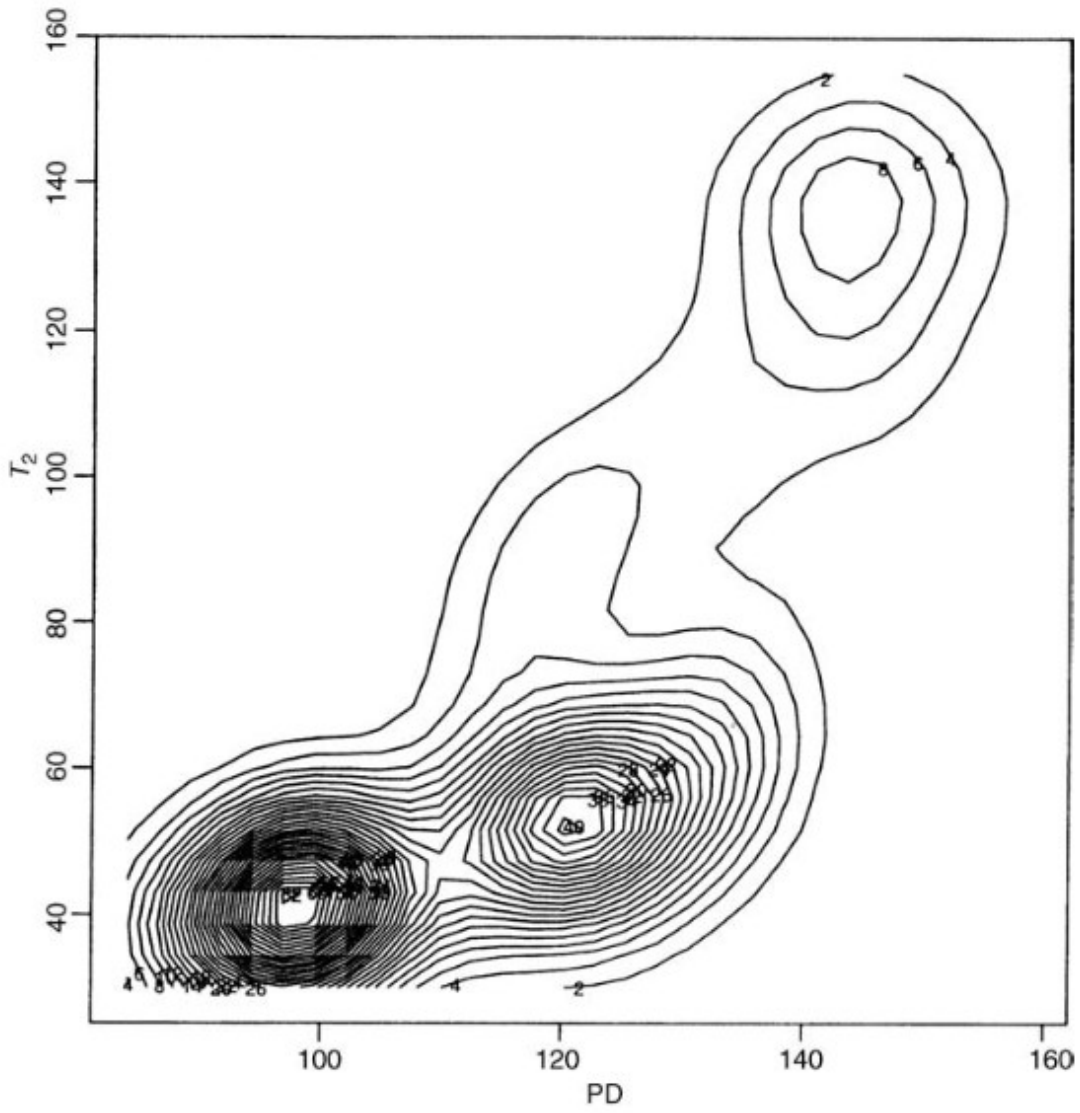


Рисунок 1.5 — Точкова діаграма МРТ, покращена двовимірною оцінкою щільності

Зазвичай досліджувані дані є багатовимірними, що унеможливлює їхнє пряме представлення на площині. Існує декілька варіантів представлення таких даних:

а) матриці точкових діаграм, що на кожному з графіків відображають розподіл тільки двох характеристик (рис. 1.6);

б) тривимірні графіки, діаграми Треліса;

в) метод головних компонент (англ. principal component analysis, PCA), пошук найкращої проекції — методи, що дозволяють зменшити розмірність досліджуваних даних. Зокрема завдяки цим алгоритмам можна зменшити розмірність до двох, що дозволить представити дані на площині у вигляді звичайної точкової діаграми [3].

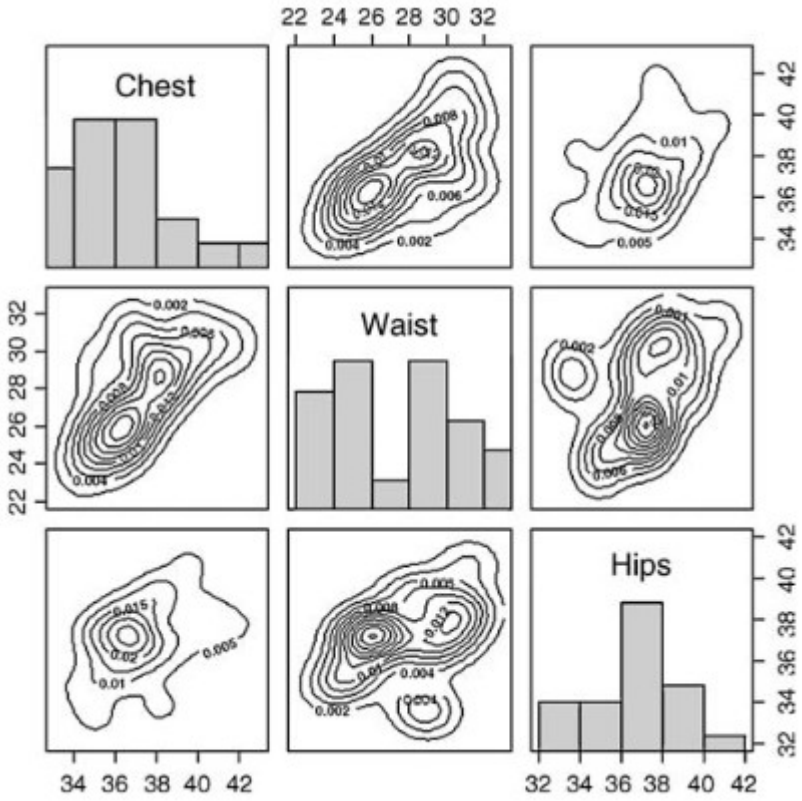


Рисунок 1.6 — Матриця точкових діаграм вимірів параметрів тіла, покращена двовимірною оцінкою щільності

1.2.2 Ієрархічна кластеризація

На відміну від багатьох інших методів кластеризації, в ієрархічній кластеризації дані не поділяються на конкретне число класів або кластерів за один крок. Натомість класифікація складається з ряду розділів, які можуть починатись з одного кластера, що містить усі точки, до *n* кластерів, кожен з яких містить одну точку. Ієрархічні методи кластеризації можна розділити на агломеративні методи,

які проводять послідовні злиття розрізнених класів у більші групи, і розділові методи, які розділяють великі збірні групи на менші кластери. Обидва види ієрархічної кластеризації можна розглядати як спробу знайти оптимальний за певними параметрами крок на кожному етапі розділення або об'єднання даних.

Важливим поняттям для ієрархічної кластеризації є поняття коефіцієнта подібності і матриці подібності. Завдяки порівнянню чисельної міри подібності між двома змінними алгоритм ієрархічної кластеризації визначає, які кластери зливати або ж роз'єднувати далі. Існує величезна кількість мір подібності. Часто використовують звичайні міри відстані, такі як евклідова метрика чи мангеттенська метрика [3]. Закономірно, матриця подібності складається з попарних коефіцієнтів подібності даних між собою.

Ієрархічні класифікації, створені або за агломеративним, або за розділовим принципом, можуть бути представлені у вигляді бінарного дерева, відомого також як дендрограма, яка ілюструє злиття або поділи, зроблені на кожному етапі аналізу. Приклад такої діаграми наведено на рис. 1.7.

Головною проблемою ієрархічних алгоритмів кластеризації є те, що оптимальний на певному етапі крок може виявитися глобально неоптимальним, через що всі наступні кроки призведуть до невдалого розподілу, адже базуються на помилковому рішенні. Деякі властивості дендрограм і те, як їх можна використовувати для інтерпретації результатів роботи ієрархічного методу кластеризації обговорюються в розділі 4 в [d3].

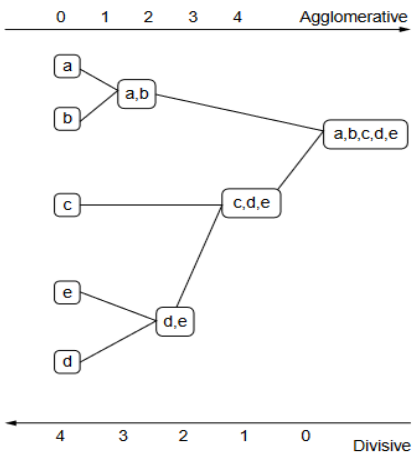


Рисунок 1.7 — Приклад дендограми, отриманої при ієрархічній кластеризації

1.2.3 Кластеризація на основі графів

Кластеризація на основі графів базується на представленні досліджуваних даних у вигляді графа, де вершинами є точки, а ребра, наприклад, мають вагу, пропорційну до міри подібності пари точок, які зв’язує це ребро.

Методи кластерного аналізу на графах поділяються на дві великі групи:

а) міжграфові — розподіляють набір графів у різні кластери;

б) в межах одного графа — розділяють вершини одного графа у кластери.

Алгоритми типу б) часто базуються на відомих алгоритмах на графах. Так, наприклад метод k-кістякових дерев шукає мінімальне кістякове дерево заданого графа, після чого видаляє з нього k найдовших ребер, що результує у k різних кластерів.

Ще один метод, що має назву Shared Nearest Neighbor Clustering, передбачає побудову нового графа, де вага ребра між двома вершинами — це кількість вершин, що знаходяться досить близько до цих вершин і пов’язані з ними, тобто є спільними сусідами цих вершин. Розділ на кластери реалізується через видалення ребер у побудованому графі, вага яких менша деякого порогового τ, тобто коли вершини не є достатньо близькими.

Серед інших алгоритмів кластеризації на графах можна згадати методи, що застосовують міру посередництва, сильну зв’язність компонент та інші. Більше інформації про них можна знайти в [6].

1.2.4 Оптимізаційна кластеризація

Ще одна техніка, яка використовується для кластеризації, — це пошук розподілу по кластерах, що мінімізує (максимізує) певний числовий критерій. Ідея оптимізаційних методів кластеризації полягає в тому, що для кожного можливого розподілу n точок на g кластерів можна порахувати індекс $c(n, g)$, що певним

чином характеризує якість даного розподілу і дає можливість порівнювати його з іншими.

За останні десятиліття було запропоновано велику кількість різноманітних критеріїв, що треба оптимізувати. Одні оперують попарними мірами подібності, інші працюють з повною матрицею досліджуваних даних.

Наприклад, одним з можливих критеріїв може бути певна характеристика матриці відмінностей (аналогічно до матриці подібностей, містить в собі інформацію про міру відмінності між парою екземплярів з запропонованого масиву даних). Ось деякі з них:

а) критерій гомогенності, що мінімізує відмінності всередині *m*-ного кластера

$$h_1(m)=\sum_{l=1}^{n_m}\sum_{v=1,\,v\neq l}^{n_m}\left(\delta_{ml,mv}\right)^r,$$

де n_m – кількість точок у *m*-ному кластері,

$\delta_{ml,mv}$ — міра відмінності між *l*-ною і *v*-ною точкою *m*-ного кластеру,

r – фіксована велична, зазвичай 1 чи 2;

б) критерій гомогенності, схожий на попередній, однак зосереджується тільки на максимальній відмінності всередині кластера. При *r* = 2 обидва вони, по суті, обмежують діаметр кластера:

$$h_2(m)=\max_{\substack{l=1,\dots,n_m\\ v=1,\dots n_m\\ v\neq l}}\left[\left(\delta_{ml,mv}\right)^r\right]\ ;$$

в) Критерій розділення, що оцінює, наскільки знайдені кластери відрізняються один від одного

$$i_1(m)=\sum_{l=1}^{n_m}\sum_{k\neq m}\sum_{v=1}^{n_k}\left(\delta_{ml,kv}\right)^r\ .$$

Коли критерій обрано, то цільову функцію можна визначити таким чином:

$$c_1(n,g)=\sum_{m=1}^{g}h(m)\ ,$$

$$c_2(n,g)=\max_{m=1,\dots,g}\left[h(m)\right]$$

або

$$c_2(n, g) = \min_{m=1,\dots,g} \left[h(m) \right] \ ,$$

де g – кількість кластерів.

Інший клас критеріїв опирається безпосередньо на наявні координати даних. Нехай матриця X розміром $n \times p$ містить дані, що треба кластеризувати. Тоді можна на її основі можна обчислити матрицю дисперсій T розмірністю $p \times p$ за формулою:

$$T = \sum_{m=1}^{g} \sum_{l=1}^{n_m} \left(x_{ml} - \bar{x} \right) \left(x_{ml} - \bar{x} \right)' \ ,$$

де x_{ml} – p-вимірний вектор спостереження l-ного об'єкта в m-ному кластері,

$\bar{x}$ – p-вимірний вектор середніх арифметичних кожної змінної по всьому набору даних.

Ця дисперсійна матриця може бути розділена на дисперійні матриці для одного кластера:

$$W = \sum_{m=1}^{g} \sum_{l=1}^{n_m} \left(x_{ml} - \bar{x_m} \right) \left(x_{ml} - \bar{x_m} \right)' \ ,$$

де – $\bar{x_m}$ p-вимірний вектор середніх арифметичних кожної змінної в m-ному кластері. Тоді міжкластерна дисперсійна матриця визначається за формулою:

$$B = \sum_{m=1}^{g} n_m \left(\bar{x_m} - \bar{x} \right) \left(\bar{x_m} - \bar{x} \right)' \ ,$$

причому справедлива рівність:

$$T = W + B \ . \qquad (1.1)$$

Для одновимірних даних (p = 1) рівняння (1.1) являє собою розподіл загальної сума квадратів змінної на суму квадратів усередині та між групами. В одновимірному випадку природним критерієм для групування буде вибір розділу, такого, щоб значення внутрішньогрупової суми квадратів було мінімальне або, що еквівалентно, щоб значення міжкластерної суми квадратів було максимальне [3].

Можливими критеріями якості можуть бути мінімізація trace(W) або det(W), що відповідає мінімізації розрізненості елементів в рамках одного кластера,

а також максимізація trace(BW^{-1}), що вказує на значну відмінність центрів кластерів.

Однак, найвідомішими представниками цієї категорії алгоритмів кластеризації є алгоритми сімейства k-means, в основі яких лежить мінімізація цільової функції, яку можна представити як

$$c(x)=\sum_{i=1}^{n}\sum_{j=1}^{k} u_{ij}\left\|a_i - x_j\right\|_p^r \ ,$$

де n – кількість елементів у вибірці;

k – кількість кластерів;

u_{ij} – функція, що визначає приналежність i-того елемента до j-того кластера, набуває значення 0 або 1;

a_i – вектор координат i-того елемента вибірки;

x_j – координати центра j-того кластера, так званого центроїда;

r – степінь, до якої підноситься норма, набуває значень 1 або 2;

p – вказує на те, яка норма використовується — евклідова (p = 2) чи мангеттенська (p = 1).

Так, для класичного алгоритму k-means, також відомого як алгоритм Ллойда [7], застосовується евклідова норма, що підноситься до квадрату. Можна показати, що мінімізація цільової функції k-means еквівалентна мінімізації trace(W) [3]. Алгоритм p-meadians використовує мангеттенську норму в першій степені, як і d-medoids, який накладає додаткову умову на x_j — центроїди кластерів мають бути елементами вибірки, в той час як інші два алгоритми цього не вимагають.

Згадані алгоритми цієї категорії об’єднує низка спільних недоліків. По-перше, всі вони збігаються лише до локальних мінімумів, не гарантуючи глобально оптимальний розв’язок. По-друге, кластери тут вважаються гіперсферами однакових радіусів з центрами у центроїдах, що не завжди є правдою.

Однак, незважаючи на це, роками ці методи були популярні серед науковців, що намагалися покращити їхню ефективність, збіжність тощо. Дослідження на цю тему можна знайти, зокрема в [8] та [9].

1.2.5 Кластеризація на основі щільності розподілу даних

У кластеризації на основі щільності кластери визначаються як області з високою щільністю даних. Об’єкти в розріджених зонах, що відокремлюють кластери, зазвичай вважаються шумовими та прикордонними точками [10].

Найпопулярнішим методом кластеризації на основі щільності є DBSCAN [11]. На відміну від багатьох новіших методів, він має чітко визначену кластерну модель під назвою «досяжність щільності». Вона заснована на з’єднанні точок у межах певних порогових відстаней. Однак вона пов’язує лише точки, які задовольняють критерію щільності, визначеному як мінімальна кількість інших об’єктів у цьому радіусі. Кластер складається з усіх об’єктів, пов’язаних достатньо щільно, а також усіх об’єктів, які знаходяться в межах діапазону цих об’єктів. При цьому утворені кластери набувають довільної форми, на відміну від багатьох інших методів. Іншою цікавою властивістю DBSCAN є те, що його складність досить низька – він вимагає лінійної кількості запитів діапазону до бази даних. Також варто зазначити, що результати роботи цього алгоритму є детермінованими для точок ядра та шуму, але не для точок кордону, однак в цілому результати після декількох запусків будуть дуже схожі, на відміну від, наприклад, k-means.

Алгоритм OPTICS [12] — це узагальнення DBSCAN, яке усуває необхідність вибору відповідного значення для параметра діапазону ε і видає ієрархічний результат, пов'язаний з кластеризацією зв'язків.

Ключовим недоліком DBSCAN і OPTICS є те, що вони очікують певного падіння щільності для виявлення меж кластерів. Для наборів даних із, наприклад, гаусовими розподілами, що перетинаються, отримані межі кластерів часто розмиті і приймають випадковий вигляд, оскільки щільність кластерів безперервно зменшується. В таких випадках алгоритми на основі щільності майже завжди перевершують такі методи, як кластеризація EM, які здатні точно моделювати дані такого роду. [2]

1.3 Опукла функція. Субградієнт. Застосування субградієнтних методів для мінімізації опуклих функцій

Функція $f(x)$ називається опуклою на множині X, якщо для будь-яких точок $x^{(1)}, x^{(2)} \in X$ виконується нерівність

$$f\left(\lambda x^{(1)} + (1-\lambda) x^{(2)}\right) \le \lambda f\left(x^{(1)}\right) + (1-\lambda) f\left(x^{(2)}\right) \qquad (1.2)$$

для всіх $\lambda \in [0,1]$.

Графіком функції $f(x)$, яка визначена на множині X, називається множина

$$G_f = \left\{(x, \alpha) \in \mathbb{R}^{n+1} \vee x \in X, \alpha \in \mathbb{R}^1, \alpha = f(x)\right\} .$$

Надграфіком функції $f(x)$, яка визначена на множині X, називається множина

$$epi\ f = \left\{(x, \alpha) \in \mathbb{R}^{n+1} \vee x \in X, \alpha \in \mathbb{R}^1, \alpha \ge f(x)\right\} .$$

Підграфіком функції $f(x)$, яка визначена на множині X, називається множина

$$hypo\ f = \left\{(x, \alpha) \in \mathbb{R}^{n+1} \vee x \in X, \alpha \in \mathbb{R}^1, \alpha \le f(x)\right\}$$ [13].

Нехай g — вектор із $\mathbb{R}^n$, $\|g\| \le K \le +\infty$, X — непорожня опукла множина, $x^{(0)} \in X$.

Функція $f(x)$, визначена на множині X такій, що $int\ X \ne \varnothing$ називається диференційовною в точці $x^{(0)} \in int\ X$, якщо існує такий вектор $g\left(x^{(0)}\right) \in \mathbb{R}^n$, що

$$f\left(x^{(0)} + \alpha g\right) = f\left(x^{(0)}\right) + \alpha \left\langle v, g\left(x^{(0)}\right)\right\rangle + o\left(x^{(0)}, v, \alpha\right)$$

при будь-яких $v \in \mathbb{R}^n, \|v\| = 1$, де α при $\frac{o\left(x^{(0)}, v, \alpha\right)}{\alpha} \to 0$, прямуючому до 0, рівномірно по v.

Якщо функція $f(x)$ диференційовна в точці $x^{(0)}$, то вона має частинні похідні першого порядку в цій точці, а $g(x^{(0)})$ буде градієнтом функції:

$$g\left(x^{(0)}\right) = grad\ f\left(x^{(0)}\right) = \left(\frac{\partial f\left(x^{(0)}\right)}{\partial x_1}, \frac{\partial f\left(x^{(0)}\right)}{\partial x_2}, \ldots, \frac{\partial f\left(x^{(0)}\right)}{\partial x_n}\right) .$$

Також градієнт функції $f(x)$ в точці $x^{(0)}$ позначають як $f'(x^{(0)})$.

Однак важливим для опуклого аналізу є вивчення властивостей опуклих функцій в точках, в яких вони недиференційовні, бо часто саме такі точки виявляються екстремальними [13].

Тому було введено поняття субградієнта, що узагальнює поняття градієнта.

Нехай функція $f(x)$ визначена і скінченна на $\mathbb{R}^n$.

Вектор $g\left(x^{(0)}\right) \in \mathbb{R}^n$ називають субградієнтом функції $f(x)$ в точці $x^{(0)}$, якщо при будь-яких $x \in \mathbb{R}^n$

$$f(x) - f\left(x^{(0)}\right) \geq \left\langle g\left(x^{(0)}\right), x - x^{(0)} \right\rangle \ .$$

Множину всіх субградієнтів функції $f(x)$ в точці $x^{(0)}$ називають субдиференціалом функції $f(x)$ в точці $x^{(0)}$ і позначають як $\partial f\left(x^{(0)}\right)$.

З геометричної точки зору, аналогічно до двохвимірного випадку, де похідна в точці $x^{(0)}$ задавала кут нахилу дотичної прямої до функції в цій точці, в $(n+1)$-вимірному випадку градієнт визначає вектор нормалі дотичної гіперплощини, що проходить через $x^{(0)}$.

Якщо опукла функція $f(x)$ не має неперервної похідної, то не в усіх точках до графіка $f(x)$ функції існує дотична гіперплощина, тобто множина *epi f* має зломи (рис. 1.8) [13]. В таких точках існує одразу багато гіперплощин, що проходять через ці точки функції $f(x)$ та більше ніде не перетинаються з її надграфіком.

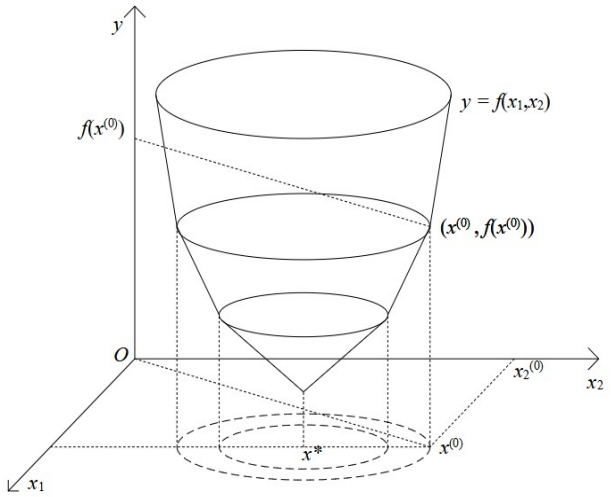


Рисунок 1.8 — Приклад опуклої функції, що не є диференційовною в усіх точках

Якщо функція $f(x)$ не диференційовна і не опукла, то субградієнт може не існувати в деяких точках, як наприклад у точці $x^{(2)}$ на рис. 1.9.

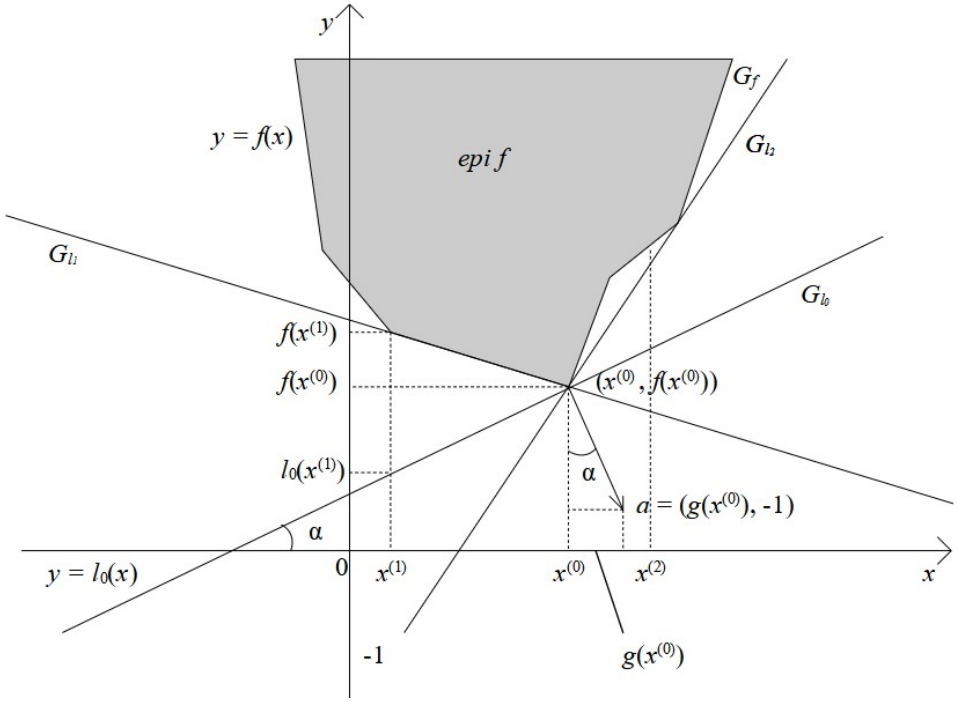


Рисунок 1.9 — Субградієнти неопуклої функції

Субградієнт функції $f(x)$ також має такі властивості:

а) якщо $f(x)=\alpha_1 f_1(x)+\alpha_2 f_2(x), \forall \alpha_1, \alpha_2 \geq 0$:

$$\partial f(x)=\alpha_1 \partial f_1(x)+\alpha_2 \partial f_2(x) \ ;$$

б) якщо $f(x)=h(Ax+b)$, тоді

$$\partial f(x)=A^T \partial h(Ax+b) \ ;$$

в) якщо $f(x)=h(f_1(x),\ldots,f_k(x))$, причому h – опукла і неспадна функція, а f_i – опуклі для $i = 1,\ldots, k$, то субградієнт функції f у точці x_0 можна знайти за формулою:

$$g=z_1 g_1+\ldots+z_k g_k \ ,$$

де $g \in \partial f(x_0)$,

$z_i \in \partial h(f_1(x_0),\ldots,f_k(x_0)), i=1,\ldots,k$,

$g_i \in \partial f_i(x_0), i=1,\ldots,k$ [14].

В [13] наведені теореми, що встановлюють певні зв’язки між опуклістю функціїї та існуванням субградієнтів:

а) для того, щоб функція $f(x)$, визначена на $\mathbb{R}^n$, була опуклою, необхідно і достатньо, щоб вона мала непорожній субдиференціал в усіх точках $\mathbb{R}^n$;

б) Для опуклої функції $f(x)$, визначеної на $\mathbb{R}^n$, субдиференціал $\partial f\left(x^{(0)}\right)$ для будь-якої точкиє $x^{(0)} \in \mathbb{R}^n$ непорожньою опуклою замкненою і обмеженою множиною.

Саме завдяки теоремі б), що гаранутє існування хоча б одного субградієнта в будь-якій точці з області визначення опуклої функції $f(x)$, стає можливим застосування субградієнтних методів оптимізації для опуклих функцій.

2 МЕТОД ЕЛІПСОЇДІВ. ПЕРЕВІРКА ПРИДАТНОСТІ МЕТОДУ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ КЛАСТЕРИЗАЦІЇ

2.1 Постановка задачі

Отже, сформулюймо постановку задачі: дослідити використання методів негладкої оптимізації для розв’язання задачі кластеризації та з’ясувати, чи є вони придатними для вирішення цього типу задач.

Задача кластеризації виникає в багатьох сферах діяльності людини, таких як маркетинг, дослідження соціальних мереж, біологія, хімія, соціологія тощо. Схожі за формулюванням задачі та цільові функції у дещо видозміненому вигляді зустрічаються ще в ширшому колі проблем, серед них транспортні задачі та дискретна оптимізація. Завдяки цьому це дослідження є актуальним не тільки для кластерного аналізу, бо його результати можуть стати в нагоді в інших галузях машинного навчання та інших прикладних задач, де необхідно оптимізувати недиференційовні функції.

В першу чергу буде розглянутий метод еліпсоїдів — відомий, гарно досліджений субградієнтний метод, що має доведену збіжність для опуклих функцій та низку деяких інших позитивних якостей, що обговорюватимуться у розділі 2.3.

2.2 Формулювання задачі кластеризації як оптимізаційної. Дослідження цільової функції

Як видно з огляду методів кластеризації, наведеного в розділі 1.2, найприроднішим є спробувати застосувати вищезазначений метод субградієнтної оптимізації для алгоритмів оптимізаційної кластеризації, наприклад, до алгоритмів сімейства k-means.

Як було зазначено в пункті 1.2.4, їхню цільову функцію можна представити в такому загальному вигляді, можливо з додатковими обмеженнями на аргумент функції:

$$c(x)=\sum_{i=1}^{n}\sum_{j=1}^{k} u_{ij}\left\|a_i - x_j\right\|_p^r \ ,$$

де n – кількість елементів у вибірці;

k – кількість кластерів;

u_{ij} – функція, що визначає приналежність i-того елемента до j-того кластера, набуває значення 0 або 1;

a_i – вектор координат i-того елемента вибірки;

x_j – координати центра j-того кластера, так званого центроїда;

r – степінь, до якої підноситься норма, набуває значень 1 або 2;

p – вказує на те, яка L_p-норма використовується — евклідова ($p = 2$) чи мангеттенська ($p = 1$).

Таким чином, конкретно для класичного алгоритму k-means цільова функція набуває вигляду

$$c(x)=\sum_{i=1}^{n}\sum_{j=1}^{k} u_{ij}\left\|a_i - x_j\right\|_2^2 \ , \qquad (2.1)$$

де u_{ij} дорівнює одиниці, якщо j-тий центроїд є найближчим до i-того екземпляру вибірки (якщо таких центроїдів декілька, то найближчим вважатиметься тільки один будь-який з них), в іншому випадку — нулю.

Оскільки метод еліпсоїдів — це, передусім, метод оптимізації опуклих функцій, доцільно дослідити функцію (2.1) на опуклість.

Як відомо, евклідова норма є опуклою функцією, що неважко показати за допомогою означення опуклої функції та застосування деяких властивостей норми. Виявляється, що квадрат евклідової норми теж є опуклою функцією.

Щоб це підтвердити, досить довести, що для будь-яких векторів $u,v \in V$, де V - векторний простір над полем K, $\|\cdot\|: V \to \mathbb{R}$, $0 \le \alpha \le 1$ справедлива нерівність

$$\left\|\alpha u+(1-\alpha)v\right\|^2 \le \alpha\left\|u\right\|^2+(1-\alpha)\left\|v\right\|^2 \ . \qquad (2.2)$$

Розглянемо ліву частину нерівності. За властивістю норми:

$$\|\alpha u+(1-\alpha)v\|^2 \le (\alpha\|u\|+(1-\alpha)\|v\|)^2 \quad .$$

Розкривши дужки у правій частині нерівності, маємо

$$\|\alpha u+(1-\alpha)v\|^2 \le \alpha^2\|u\|^2+2\alpha(1-\alpha)\|u\|\|v\|+(1-\alpha)^2\|v\|^2 \quad . \tag{2.3}$$

Повертаючись до нерівності (2.2), замінимо її ліву частину на праву частину нерівності (2.3). Тоді, достатньо довести, що

$$\alpha^2\|u\|^2+2\alpha(1-\alpha)\|u\|\|v\|+(1-\alpha)^2\|v\|^2 \le \alpha\|u\|^2+(1-\alpha)\|v\|^2 \quad .$$

Перенісши всі доданки у праву частину нерівності, отримуємо

$$0 \le (\alpha-\alpha^2)\|u\|^2-2\alpha(1-\alpha)\|u\|\|v\|+((1-\alpha)-(1-\alpha)^2)\|v\|^2 \quad .$$

В процесі спрощення правої частини нерівності маємо

$$0 \le (\alpha-\alpha^2)\|u\|^2-2(\alpha-\alpha^2)\|u\|\|v\|+(\alpha-\alpha^2)\|v\|^2 \quad .$$

Винісши за дужки $\alpha-\alpha^2$ бачимо повний квадрат різниці норм векторів *u* та *v*:

$$0 \le (\alpha-\alpha^2)(\|u\|^2-2\|u\|\|v\|+\|v\|^2) \quad ,$$

$$0 \le (\alpha-\alpha^2)(\|u\|-\|v\|)^2 \quad . \tag{2.4}$$

Якщо проаналізувати, які значення можуть набувати дужки у правій частині нерівності (2.4), то легко бачити, що обидві вони невід'ємні. З цього випливає, що нерівність (2.4) правильна, а тому виконуватиметься і нерівність (2.2), що означає, що квадрат евклідової норми теж є опуклою функцією.

Однак це лише складова цільової функції. Як наявність множника u_{ij} впливає на опуклість цієї функції? Для з'ясування цього питання розглянемо такий приклад.

Припустимо, що дані, які потребують кластеризації, одновимірні, а кластерів лише два. Тоді цільова функція залежатиме лише від двох змінних, і ми зможемо побудувати графік цільової функції у тривимірному просторі, аби оцінити, з чим маємо справу.

У таблиці 2.1 наведені координати точок згенерованої експериментальної вибірки. Їхній розподіл на прямій зображено на рис. 2.1.

Таблиця 2.1 — Координати точок експериментальної вибірки

Координата
3.5902
2.8215
7.3460
8.5301
3.3625
3.3972
8.0192
8.4488
2.6530
3.5183

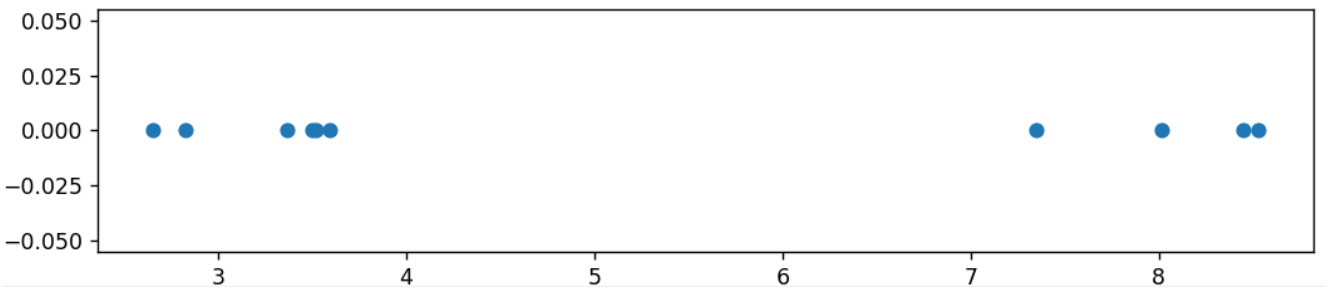


Рисунок 2.1 — Точкова діаграма даних з табл. 2.1

Неозброєним оком видно два чіткі кластери з центроїдами $c_1 \approx 3, c_2 \approx 8$. Погляньмо тепер на графік цільової функції на рис. 2.2.

Досліджувана функція не є опуклою. Зокрема, опуклість явно змінюється при переході між півпросторами, на які простір розділяється площиною-бісектрисою кута між осями C1 та C2. Функція насправді виявляється симетричною відносно неї, і це не дивно, адже значення цільової функції ніяким чином не залежить від нумерації центроїдів — як $c_1 \approx 3, c_2 \approx 8$, так і $c_1 \approx 8, c_2 \approx 3$ мінімізуватимуть цю функцію.

Незважаючи на це, є ділянки, де функція дійсно є опуклою. Так, легко показати, що при сталих коефіцієнтах u_{ij} функція (2.1) буде опуклою, як лінійна комбінація опуклих функцій з невід'ємними коефіцієнтами.

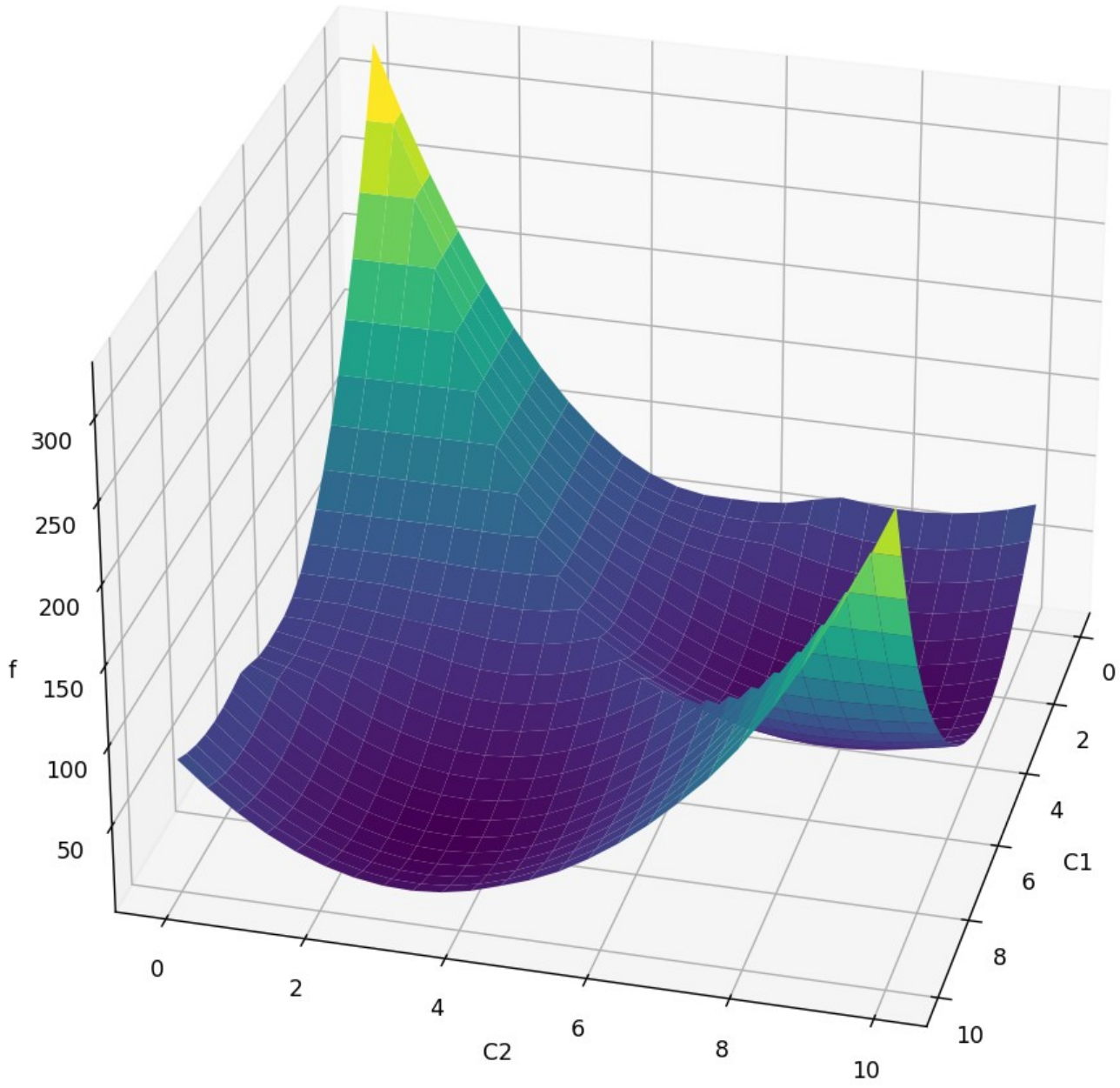


Рисунок 2.2 — Графік цільової функції k-means для вибірки, наведеної у табл. 2.1

Зміна опуклості може статися при переході точки із кластера в кластер, однак виявляється, що такі переходи не завжди так сильно впливають на опуклість цільової функції.

Так, на графіку можна бачити набагато меншу зміну опуклості вздовж іншої лінії в межах одного симетричного півпростору. Цікаво, що така лінія тільки одна, хоча декілька точок змінили приналежність до кластерів. Навіть для такого простого прикладу можна виділити три різні ситуації, на межі яких міг виникнути

перегин: коли всі точки належать одному кластеру і обидва центроїди знаходяться зліва; коли точки розподілені між двома кластерами; коли два центроїди розташовані справа, і всі точки належать одному з них. Незважаючи на це, спостерігаємо лише один перегин. Загалом цю функцію можна розглядати як квазіопуклу.

Саме з цієї причини застосування субградієнтного метода для оптимізації такої функції видається можливим. Однак, через нестрогу опуклість зберігається шанс потрапити в локальний мінімум, через що результат оптимізації значною мірою залежить від обраного початкового наближення, що, в принципі, справедливо і для звичайної реалізації k-means.

2.3 Метод еліпсоїдів

Метод еліпсоїдів — це ітераційний метод мінімізації опуклих функцій, окремий випадок субградієнтних методів з розтягненням простору в напрямі субградієнта, запропонованих Н.З. Шором [15].

Метод еліпсоїдів має довгу історію. Як субградієнтний метод з перетворенням простору в напрямку субградієнта, він був запропонований Шором [16], і майже одночасно з ним він був запропонований Аркадієм Немировським та Давидом Юдіним як метод послідовних відсічень [17]. Згодом Хачіян побудував на основі методу еліпсоїдів поліноміальний алгоритм для розв’язання задач лінійного програмування, чим і довів існування такого методу для цієї задачі [18].

Метод еліпсоїдів має низку гарних якостей, що вирізняють його серед інших алгоритмів:

а) метод еліпсоїдів гарантовано знаходить розв’язок задачі для заданої точності за скінченну кількість ітерацій, до того ж ця кількість залежить лише від розмірності простору і не залежить від складності і виду функції. Доведення цього факту наведене в [16];

б) відноситься до методів недиференційовної оптимізації, що дозволяє використовувати його для мінімізації негладких функцій;

в) має нескладну ідею та зрозумілу геометричну інтерпретацію;

З недоліків варто відмітити, що метод еліпсоїдів не підходить для задач великої розмірності. В [19] показано, що

$$K \le 2n^2 \log\left(\frac{RG}{\varepsilon}\right) \ , \tag{2.5}$$

де K — кількість ітерацій;

n — кількість змінних;

R — початковий радіус;

G — максимальна евклідова норма субградієнта з субдиференціалу початкового еліпсоїда;

ε — задана точність.

З (2.1) видно, що при збільшенні кількості змінних K буде значно зростати, через що час виконання задачі буде занадто довгим.

Ідея методу еліпсоїдів для мінімізації опуклої функції $f : \mathbb{R}^n \to \mathbb{R}$ полягає в генерації "спадної" послідовності еліпсоїдів у просторі $\mathbb{R}^n$, яким гарантовано належить точка мінімуму x^*. За допомогою обчислення субградієнта в центрі еліпсоїда можемо визначити півпростір, де точно не знаходитиметься мінімум функції f, з'ясовуючи тим самим, де відбудеться наступна ітерація.

Нехай k-тому кроці маємо еліпсоїд $\varepsilon^{(k)}$, що гарантовано містить мінімум функції f. Обчисливши субградієнт $g^{(k)}$ у точці $x^{(k)}$, центрі $\varepsilon^{(k)}$, знаходимо пів еліпсоїд, що міститиме шуканий мінімум. Ці точки можна описати як

$$\varepsilon^{(k)} \cap \left\{ z \vee \ g^{(k)T}\left(z - x^{(k)}\right) \le 0 \right\} \ .$$

Наступний еліпсоїд $\varepsilon^{(k+1)}$ знаходиться як еліпсоїд найменшого об'єму, що містить в собі тільки що знайдений пів еліпсоїд з мінімумом функції f (див. рис. 2.3). Цей процес повторюється, поки різниця між $f(x^{(k)})$ та її мінімальним значенням f^* не стане меншою за задану точність ε. Звичайно, f^* заздалегідь

невідоме, але різницю з цим значенням можна оцінити. Вигляд цієї оцінки наведений в [19].

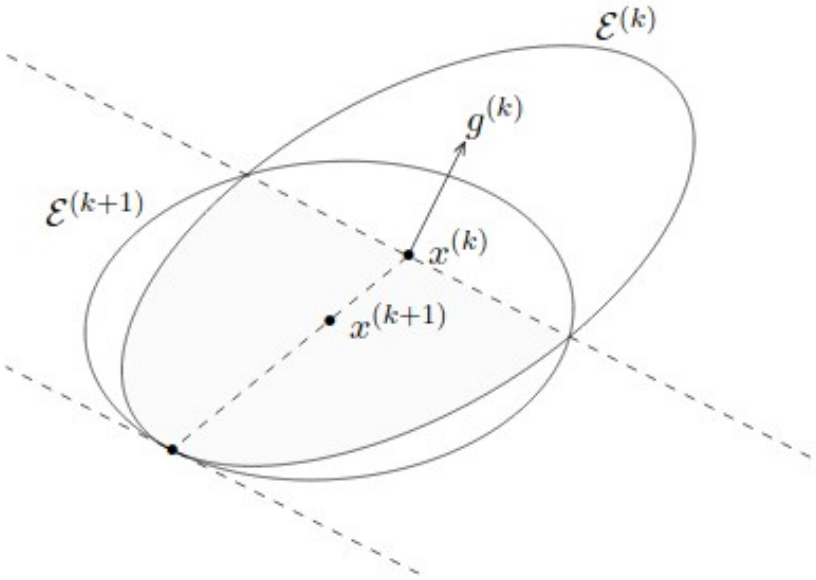


Рисунок 2.3 — Геометрична інтерпретація методу еліпсоїдів

З кожним кроком геометричний об’єм еліпсоїда зменшуватиметься, завдяки чому і вдається довести збіжність методу [19].

У цій роботі пропонується імплементація алгоритму Шора, одного з реалізацій методу еліпсоїдів. Алгоритм Шора без обмежень на допустимі значення x_p* має такий вид.

Нехай дана опукла функція $f:\mathbb{R}^n \to \mathbb{R}$, x_p* позначатиме знайдене оптмальне значення змінної, а f_p* — знайдене мінімальне значення функції f.

Ініціалізація. Нехай дано стартову точку $x_0 \in \mathbb{R}^n$ та початковий радіус r_0. Введемо $n \times n$-матрицю перетворення простору B і покладемо B_0 рівною одиничній $n \times n$-матриці E_n, f_p* покладемо нескінченність, x_p* поки буде рівним x_0. Перейдемо до першої ітерації зі значеннями x_0, r_0, B_0, f_p*, x_p*.

Нехай на k-ій ітерації були знайдені значення x_k, r_k, B_k. Перехід до $(k+1)$-ої ітерації відбувається таким чином:

Крок 1. Рахуємо значення функції $f(x_k)$ та значення субградієнта $g(x_k)$. Якщо $f(x_k)$ менше за f_p^*, то x_p^* присвоюється значення x_k, f_p^* присвоюється $f(x_k)$. Якщо виконується

$$\left\| B_k^T g(x_k) \right\| r_k \le \varepsilon \quad ,$$

то зупиняємось і повертаємо знайдене значення x_p^*. Інакше переходимо до наступного кроку.

Крок 2. Покладемо

$$\xi_k = \frac{B_k^T g(x_k)}{\left\| B_k^T g(x_k) \right\|} \quad .$$

Крок 3. Обчислимо наступну точку

$$x_{k+1} = x_k - h_k B_k \xi_k \quad ,$$

де $h_k = \frac{1}{n+1} r_k$.

Крок 4. Обчислимо нові матрицю перетворення простору та радіус

$$B_{k+1} = B_k + \left(\sqrt{\frac{n-1}{n+1}} - 1 \right) (B_k \xi_k) \xi_k^T \quad ,$$

$$r_{k+1} = r_k \frac{n}{\sqrt{n^2 - 1}}$$

Крок 5. Переходимо до $(k+1)$-ої ітерації зі значеннями x_{k+1}, r_{k+1}, B_{k+1}.

Наведений алгоритм з незначними модифікаціями можна застосовувати і у випадку наявності обмежень для x_p^*, що показано у [15].

2.4 Застосування методу еліпсоїдів для знаходження оптимального положення центроїдів

Перед тим, як застосувати субградієнтний метод оптимізації, а саме метод еліпсоїдів, необхідно, насамперед, знайти спосіб обчислити субградієнт функції, яку ми збираємось мінімізувати.

Отже, цільова функція має вигляд

$$c(x)=\sum_{i=1}^{n}\sum_{j=1}^{k}u_{ij}\left\|a_i-x_j\right\|_2^2 \quad . \tag{2.6}$$

В будь-якій точці цільова функція визначається як сума певного набору квадратів евклідової норми. Для того, аби знайти формулу для субградієнта функції (2.6) в деякій точці x_0, необхідно виконати такі кроки:

1) виразити субградієнт квадрата евклідової норми через субградієнт евклідової норми;

2) визначити субградієнт евклідової норми;

3) підставити вираз a_i-x_j замість аргументу функції та застосувати необхідні перетворення.

Крок 1. Оскільки функція норми є опуклою функцією, що набуває лише невід'ємних значень, а піднесення в квадрат — опукла, неспадна для невід'ємного аргументу функція, то для знаходження субградієнта на цьому етапі можемо застосувати властивість в) субградієнта, вказану у розділі 1.3. Тоді маємо, що для $g\in\partial\left\|x_0\right\|^2, g'\in\partial\left\|x_0\right\|$ справедливий вираз

$$g=2\left\|x_0\right\|g' \quad .$$

Крок 2. Субградієнт евклідової норми в точці x_0 визначається за формулою , $\frac{x_0}{\left\|x_0\right\|}$ що легко доводиться за означенням.

Крок 3. Враховуючи, що евклідова норма шукається від a_i-x_j , отримуємо загальну формулу для визначення субградієнта у точці x_0:

$$g=-2\sum_{i=1}^{n}\left(a_i-x_i'\right) \quad ,$$

де x_i' — найближчий центроїд до i-тої точки вибірки.

Тепер, володіючи формулою субградієнта, нарешті запускаємо метод еліпсоїдів. Починаємо з малого — з відомої вибірки ірисів [20]. Результати роботи алгоритму з випадково обраними початковими центроїдами наведені у вигляді точкових діаграм (рис. 2.4), побудованих після застосування PCA (див. розділ 1.2.1) до вихідних даних.

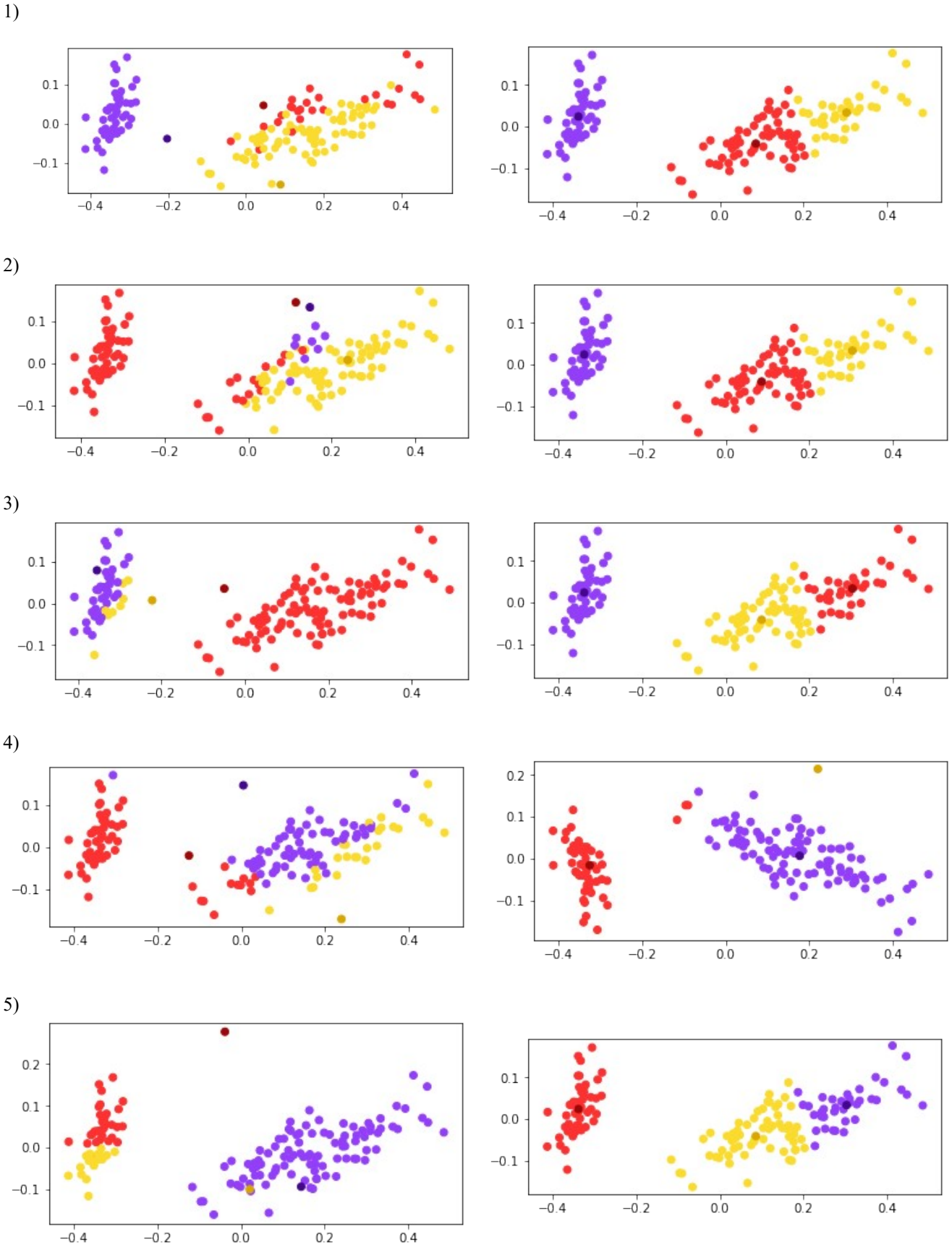


Рисунок 2.4 — Результати кластеризації даних про іриси за допомогою методу еліпсоїдів (зліва наведені початкові розподіли точок, справа - фінальні)

З діаграм можемо бачити, що методу еліпсоїдів в більшості випадків вдається знайти глобальний мінімум і найкращу кластеризацію, яка можлива для цієї вибірки з використанням критерію k-means. Однак як і при звичайній реалізації k-means, прослідковується залежність якості результату від вибору початково наближення. Так, в четвертому експерименті центроїди були обрані невдало, що призвело до виродженого розподілу екземплярів по кластерах — один з них виявився пустим. Цікаво відзначити, що якість результату кластеризації різна для різних реалізацій. Наприклад, для початкових центроїдів з п’ятого експерименту бібліотечна реалізація k-means пакета scikit-learn знаходить лише два кластери (рис. 2.5), в той час як для точок з четвертого знаходить всі кластери.

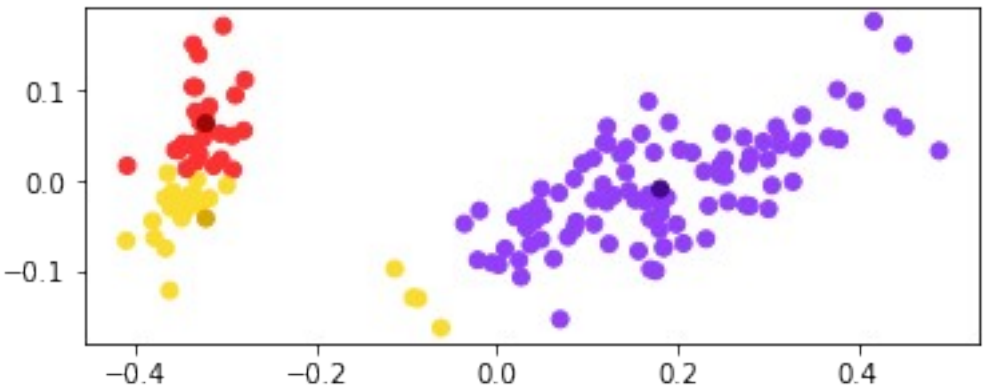
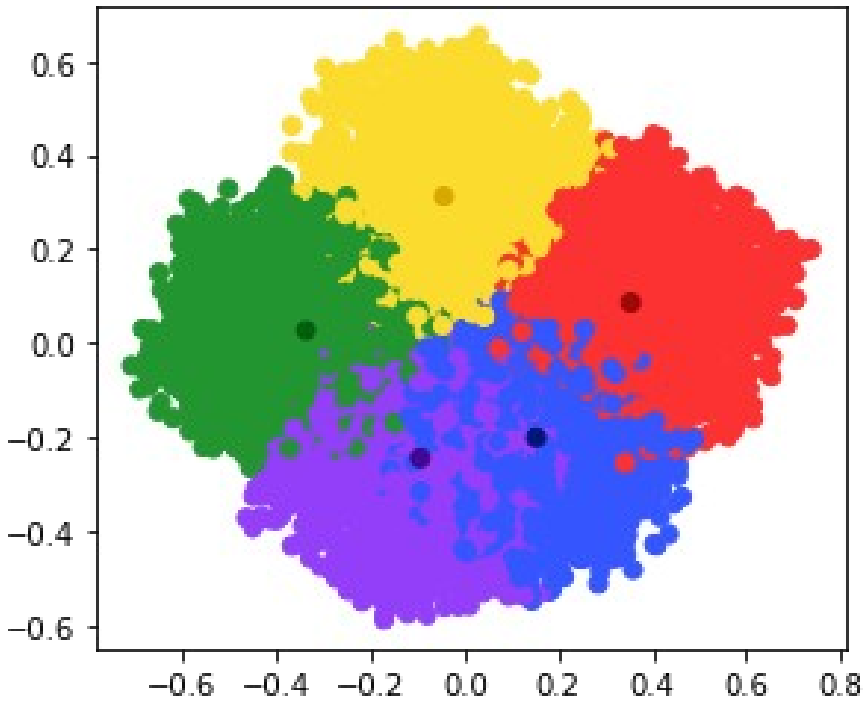


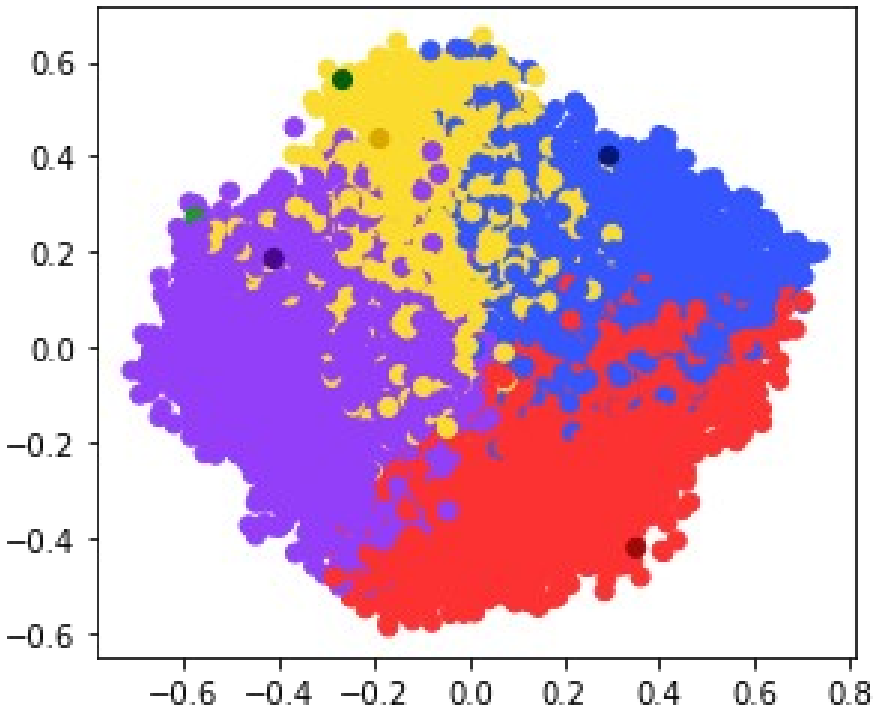
Рисунок 2.5 — Результат роботи Kmeans(...).fit(...) з пакету sklearn.cluster з такими же початковими центроїдами як і в експерименті 4) рис. 2.4

Збільшимо обсяг вхідних даних зі 150 до 10 000 об’єктів. Для цього експерименту згенеруємо десять тисяч чотирьохвимірних векторів навколо п’яти випадково обраних центрів і кластеризуємо їх за допомогою алгоритму на основі методу еліпсоїдів.

Рис. 2.6 *а*) зображає точкову діаграму даних з центроїдами, навколо яких генерувалась вибірка. На рис. 2.6 б) демонструє початкове положення центроїдів, що були передані на вхід алгоритмам. На рис. 2.6 *в*) наведений результат кластеризації за допомогою методу еліпсоїдів.

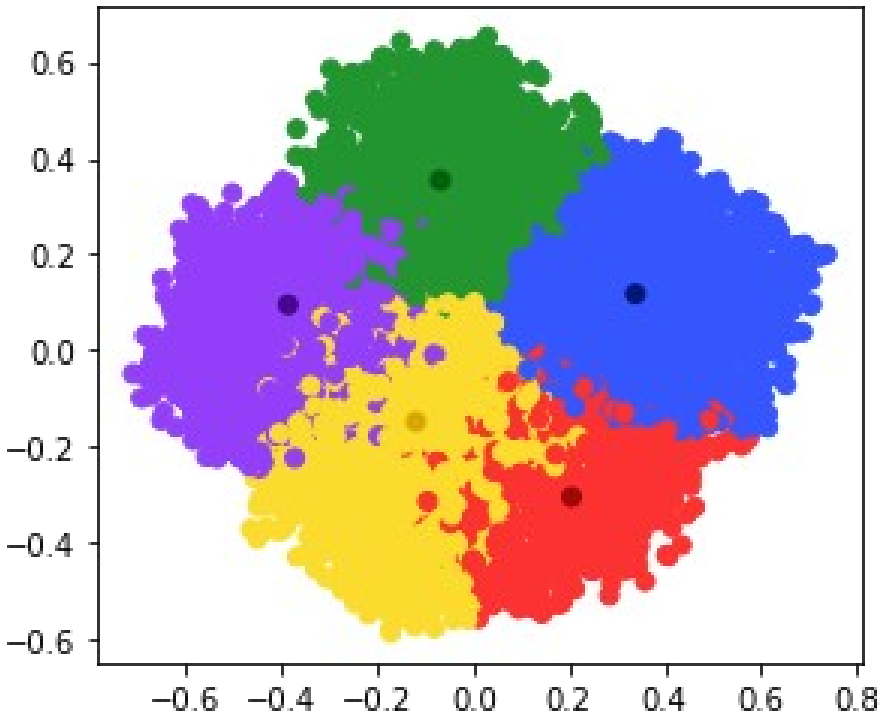


а)



б)

Рисунок 2.6 — Експеримент з більшою вибіркою: а — центроїди, використані під час генерації; б — випадково обрані початкові центроїди для кластеризації; в — результат кластеризації методом еліпсоїдів;



в)

Рисунок 2.6 (продовження)

З графіків можна бачити, що алгоритм на основі методу еліпсоїдів відновив згенерований поділ, що свідчить на користь того, що відсутність строгої опуклості цільової функції не заважає знаходити досить оптимальний розподіл.

Однак, є проблема: алгоритм на основі метода еліпсоїдів витрачає на вирішення задачі цього розміру цілих сорок хвилин. Виявляється, що для знаходження оптимального поділу довелося виконати аж 5013 ітерацій. Повертаючись до попереднього експерименту, навіть на малесенькій вибірці зі всього 150 екземплярів алгоритм на основі методу еліпсоїдів здійснює значну кількість ітерацій. У таблиці 2.2 наведено кількість ітерацій, виконаних під час п’яти запусків алгоритму з рис. 2.4.

Таблиця 2.2 – Кількість ітерацій, здійснених методом еліпсоїдів

Кількість ітерацій
541
871
688

370
614

Чому так відбувається? По-перше, порушення опуклості заважатиме методу швидко збігтися до глобального мінімуму. По-друге, повернувшись ще раз до графіка цільової функції для найпростішого варіанту вхідних даних (рис. 2.2), бачимо ще можливу причину — яружність цієї функції.

Запустивши алгоритм ще раз, розглянемо значення цільової функції на різних ітераціях (табл. 2.3).

Таблиця 2.3 — Значення цільової функції на різних кроках оптимізації за допомогою методу еліпсоїдів

№ ітерації	Значення цільової функції
1	751.414
2	648.817
3	562.949
4	491.179
5	431.279
21	160.164
22	158.506
23	157.628
24	156.704
25	155.964
26	155.813
100	152.742
101	152.828
200	152.425
201	152.449
280	152.382
281	152.348

Найзначніше функція зменшується за перші декілька ітерацій, і вже до двадцятої ітерації функція змінюється слабо, блукаючи по яру, що містить мінімум, ще більш ніж 250 ітерацій.

Отже, порушення строгої опуклості цільової функції призводить до погіршення ефективності методу еліпсоїдів, який і так є відносно повільним. Він збігається зі швидкістю геометричної прогресії зі знаменником, близьким до одиниці [16]. Також через порушення опуклості на роботу методу негативно впливає яружність функції, незважаючи на те, що метод еліпсоїдів — це метод з перетворенням простору, що адаптований до роботи з функціями такого характеру.

Шляхів подолання цієї проблеми можна запропонувати декілька: змінити цільову функцію таким чином, щоб вона стала опукла та/або менш яружна, чи замінити метод еліпсоїдів на інший алгоритм, що може краще справлятися з зазначеними недоліками цільової функції. Для початку, змінімо цільову функцію.

2.5 Зміна цільової функції на критерій p-medians

Для зменшення ступеню яружності функції можна спробувати використати мангеттенську норму. Це приводить нас до критерію іншого відомого методу оптимізаційної кластеризації під назвою p-medians. Цільова функція p-medians має вигляд

$$c(x)=\sum_{i=1}^{n}\sum_{j=1}^{k} u_{ij}\left\|a_i - x_j\right\|_1 \quad . \tag{2.7}$$

За рахунок відсутності піднесення до квадрату як різниці координат, так і самої норми різниці векторів a_i та x_j, така цільова функція має більше змінюватися навколо мінімуму. Зазвичай він розташований серед точок з вибірки, через що різниця координат з деякими з них може бути незначною, і після піднесення до другої степені їх вплив на значення цільової функції стає ще меншим, через що мінімум опиняється посеред дна яру.

Для візуалізації цієї тези побудуємо графік функції (2.7) для даних з табл. 2.1. З рис. 2.7 можна бачити, що функція дійсно стала менш яружною і тепер спадає до мінімуму значно швидше.

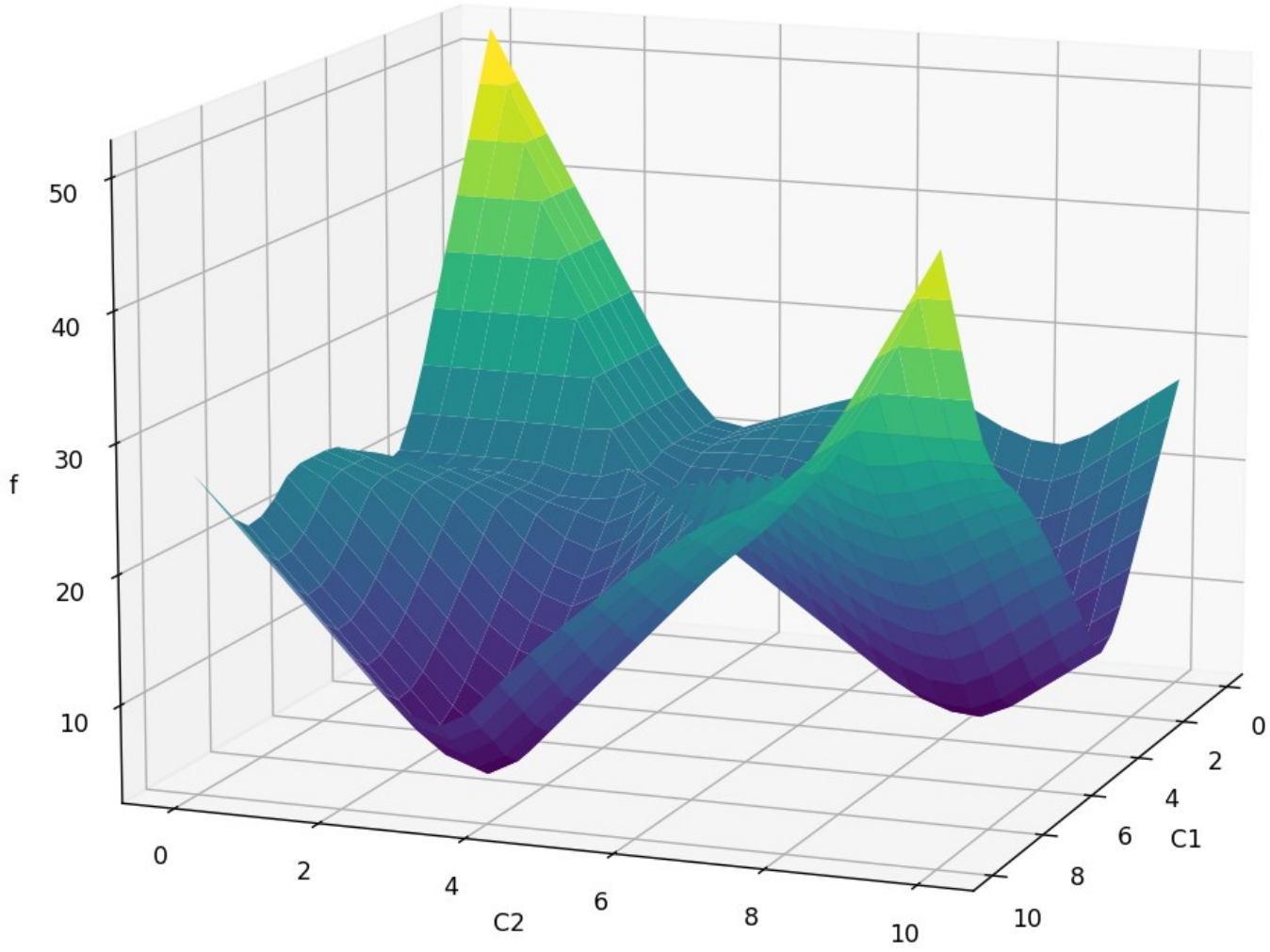


Рисунок 2.7 — Графік функції (2.7) для даних з табл. 2.1

Однак в порівнянні з попереднім варіантом цільової функції, функція (2.7) частіше змінює опуклість. Перегинів стало більше, що може негативно вплинути на роботу алгоритму.

Порівнюючи результати роботи алгоритмів з цими двома функціями, бачимо, що дійсно порушення опуклості грає більшу роль в ефективності роботи алгоритму, ніж яружність. В таблиці 2.4 наведено кількість здійснених ітерацій з

кожною з функцій на одних і тих же даних (вибірка про іриси, випадкові початкові центроїди). Мінімізація критерію p-medians в середньому потребує втричі більше ітерацій і, як наслідок, втричі більше часу.

Таблиця 2.4 — Кількість ітерацій методу еліпсоїдів при мінімізації двох відомих критеріїв

Критерій k-means	Критерій p-medians
675	1727
691	1729
730	1696
606	1731
249	779

Однак, варто зазначити, що незважаючи на повільну роботу, мінімізація функції (2.7) врешті-решт призводить до знаходження глобального оптимуму у більшості випадків. На рис. 2.8 зображено один з результатів кластеризації з використанням критерію p-medians. Він збігається з більшістю інших кластеризацій.

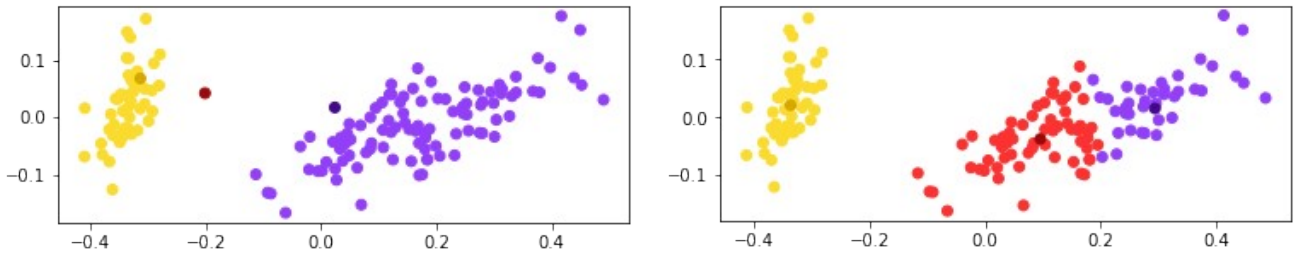


Рисунок 2.8 — Результат кластеризації з використанням критерію p-medians

Така зміна функції не призвела до очікуваних покращень. Однак у нас все ще є два невипробувані підходи для покращення швидкодії кластеризації на основі методу негладкої оптимізації.

2.6 Застосування методу штрафів для критерію k-means

Альтернативою зменшенню ступеня яружності цільової функції може стати забезпечення її опуклості. Повертаючись до цільової функції (2.6), що є критерієм k-means, можемо скорегувати цю функцію за допомогою методу штрафів [21] для забезпечення її опуклості. Нова цільова функція матиме вигляд

$$c(u,x)=\sum_{i=1}^{n}\sum_{j=1}^{k}|u_{ij}|\|a_i-x_j\|_2^2+\sum_{i=1}^{n}\left(R_i\left|\sum_{j=1}^{k}|u_{ij}|-1\right|\right), \tag{2.8}$$

де u_{ij} — це тепер змінні, що позначають міру належності i-тої точки вибірки j-тому кластеру;

R_i – штрафні коефіцієнти, константи.

Другий доданок функції (2.8) накладає обмеження на u_{ij}, що сума мір її приналежностей до всіх кластерів має дорівнювати одиниці. Також цей доданок забезпечує опуклість функції. При порушенні закладених обмежень значення функції стрімко зростатиме. Варто відзначити, шо за рахунок модуля на змінних u_{ij} задача по суті є багатоекстремальною, однак будь-який екстремум буде глобальним, адже знак u_{ij} не впливає на значення функції.

Запустивши метод еліпсоїдів з такою цільової функцією, одразу стикаємось з проблемою швидкодії. Порівняно з критеріями k-means та p-medians, ця цільова функція для вибірки ірисів мінімізується за кілька хвилин, а не секунд. Для аналізу причин такої поведінки розгляньмо роботу алгоритму на крихітній вибірці з 8 точок у двовимірному просторі (рис. 2.9).

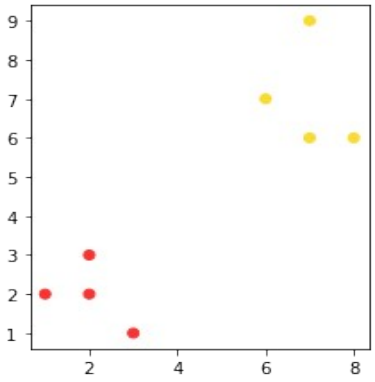


Рисунок 2.9 — Точкова діаграма вибірки, що розглядається

Навіть на такій маленькій вибірці метод еліпсоїдів здійснює тисячі ітерацій. Таке стрімке зростання кількості ітерацій пов'язане зі значним збільшенням кількості змінних у векторі аргументу цільової функції. Замість kd змінних тепер там налічується аж $nk + kd$ змінних, через що розмірність задачі значно зростає. Звертаючись до нерівності (2.5), бачимо, що кількість ітерацій методу еліпсоїдів пропорційна квадрату розмірності задачі, тому не дивно, що кластеризація вибірки з ірисами виявляється непосильною задачею — кількість змінних для неї виявляється рівною аж 462.

Кількість ітерацій можна дещо варіювати, змінюючи порядок штрафного коефіцієнта. На рис. 2.10 видно, що чим більшим є цей коефіцієнт, тим більше ітерацій виконує метод еліпсоїдів.

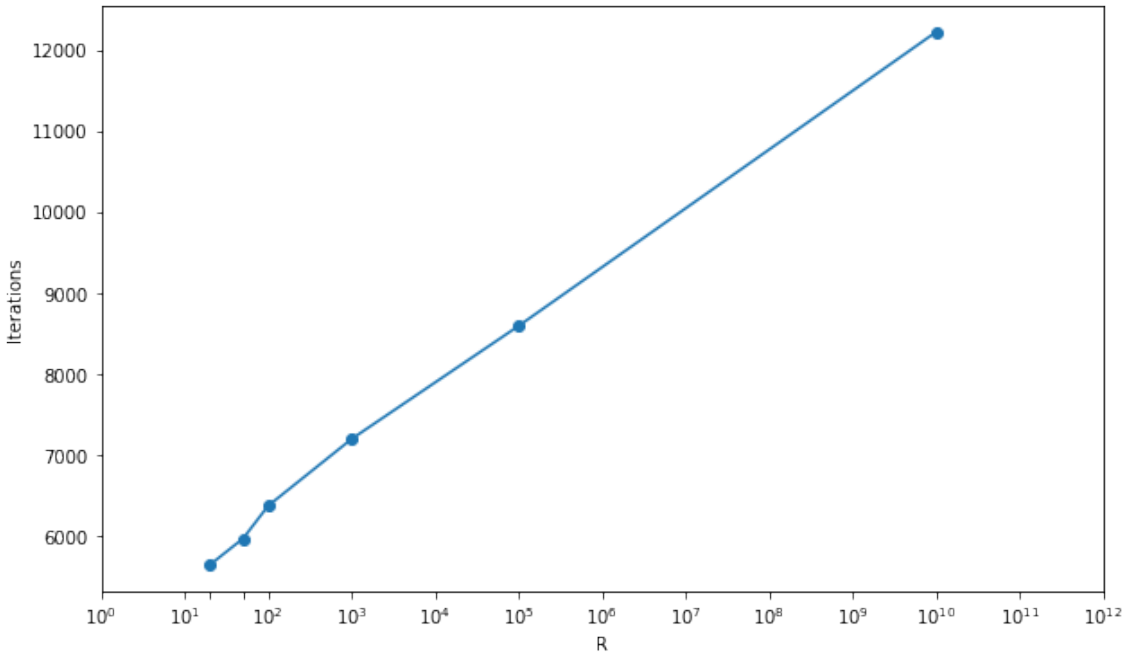


Рисунок 2.10 — Графік залежності кількості ітерацій від штрафного коефіцієнта

Причиною є лінійна залежність кількості ітерацій від логарифма максимальної евклідової норми субградієнта з початкового еліпсоїда. Субрадієнт функції (2.8) визначається як сума субградієнтів по всіх змінних. Субградієнт g за змінною u_{ij}, зокрема, визначається за формулою

$$g = g_1 + g_2 \ ,$$

де $$g_1 = \begin{cases} \|a_i - x_j\|_2^2, u_{il} \geq 0 \\ -\|a_i - x_j\|_2^2, u_{il} < 0 \end{cases} ;$$

$$g_2 = \begin{cases} R_i, u_{ij} \left(\sum_{l=1}^{k} |u_{il}| - 1 \right) \geq 0 \\ -R_i, u_{ij} \left(\sum_{l=1}^{k} |u_{il}| - 1 \right) < 0 \end{cases} .$$

Тобто, порядок норми субградієнта функції (2.8) визначатиметься порядком штрафного коефіцієнта. Для найбільш ефективної роботи з цією функцією варто обирати такий штрафний коефіцієнт, щоб його порядок незначно відрізнявся від порядку значень цільової функції, коли закладені в неї обмеження виконуються. Великі значення штрафів не допоможуть функції збігтись швидше.

Поліпшення опуклості теж не допомогло. Тому спробуємо застосувати інший підхід — зміну алгоритму оптимізації. Сімейство субградієнтних методів під назвою r-алгоритми краще адаптоване до роботи з яружними функціями, і саме один з таких методів я спробую застосувати у наступному розділі.

3 ЗАСТОСУВАННЯ R-АЛГОРИТМУ

3.1 r-алгоритм

Сімейство r-алгоритмів — це субградієнтні методи з розтягом простору в напрямку різниці двох послідовних субградієнтів, що можуть значно прискорити мінімізацію негладких опуклих функцій завдяки застосуванню таких двох принципів:

а) використання процедури найшвидшого спуску в напрямку антисубградієнта опуклої функції в перетвореному просторі змінних;

б) використання операції розтягу простору в напрямку різниці двох послідовних субградієнтів, що допомагає зменшити ступінь витягнутості поверхонь рівня яружних функцій у перетвореному просторі змінних.

Завдяки цим принципам r-алгоритми показали себе ефективними для мінімізації сильно яружних функцій, хоча і проблема обґрунтування збіжності цих алгоритмів є досі відкритою.

Існують декілька варіацій r-алгоритмів, так званих форм, що відрізняються обсягом використовуваної оперативної пам’яті, трудомісткістю ітерації та стійкістю методу, зокрема B-форма, економна B-форма та H-форма. В цій роботі буде застосовано Python-адаптацію B-форми r-алгоритму, що була реалізована на основі коду програми ralgb5 на мові Octave [22], що є найбільш стійкою варіацією серед інших алгоритмів [23].

3.2 Застосування r-алгоритму для розв’язання задачі кластеризації

Протестувавши метод на основі r-алгоритму на вибірці ірисів, можемо бачити, що отримані результати кластеризації збігаються з попередніми (рис. 3.1).

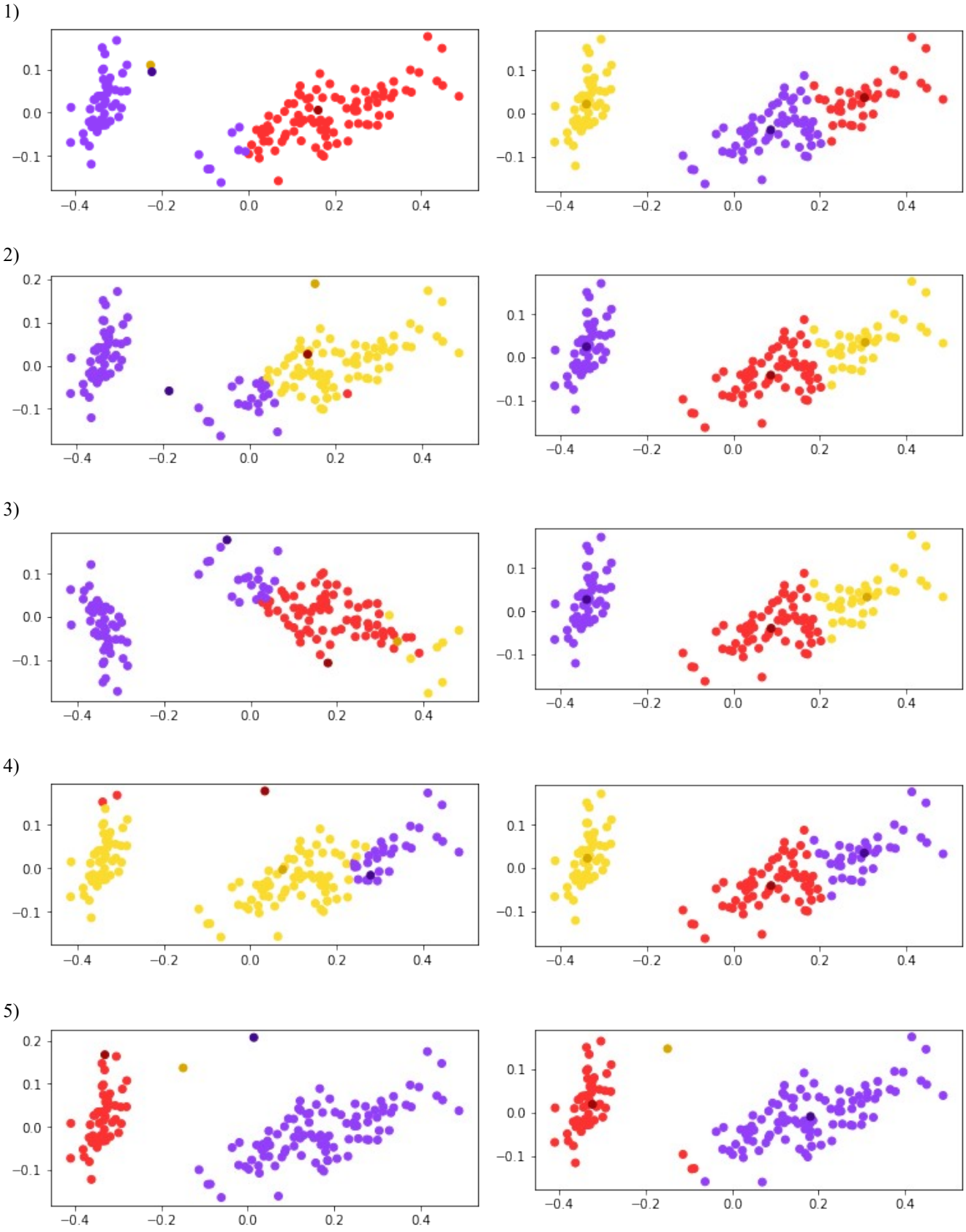


Рисунок 3.1 — Результати кластеризації даних про іриси за допомогою r-алгоритму (зліва наведені початкові розподіли точок, справа — фінальні)

Робота r-алгоритму з критерієм k-means є задовільною. З п’ятого експерименту на рис. 3.1 видно, що, як і для інших реалізацій, при невдалому виборі початкових центроїдів будується вироджене групування, тобто в цьому питанні r-алгоритм не дає переваги. Однак він збігається швидше, ніж алгоритм на основі метода еліпсоїдів: порівняно з ним середня кількість ітерацій скоротилась в 6 разів (табл. 3.1).

Таблиця 3.1 - Кількість ітерацій, здійснених r-алгоритмом

Кількість ітерацій
87
104
90
97
61

Перевага у швидкості r-алгоритму зберігається і при збільшенні розміру вибірки. Кластеризація вибірки обсягом 10 000 об’єктів, яку алгоритм на основі метода еліпсоїдів виконував 40 хвилин, займає лише за півтори хвилини.

А от якщо спробувати застосувати r-алгоритм для мінімізації критерію p-medians чи функції зі штрафами, то виявляється, що він не справляється з задачею. Це відбувається через те, що при мінімізації цих функції виникають лінійно залежні послідовні субградієнти, що унеможливлює коректну роботу r-алгоритму. З дослідженнями роботи r-алгоритму з кусково-лінійними функціями, чим по суті і є, наприклад, критерій p-medians, можна ознайомитися в [24].

3.3 Порівняння розглянутих методів між собою та з пакетом scikit-learn

Серед розглянутих алгоритмів вирішення задачі кластеризації найшвидшим виявився метод на основі r-алгоритму, що мінімізує критерій k-means. Алгоритми, що застосовували метод еліпосоїдів, значно програють йому в часі через неефективну роботу з яружною квазіопуклою функцією, причому чим більше

порушується опуклість, тим більше стає кількість ітерацій, що призводить до збільшення часу виконання. На рис. 3.2 зображений порівняльний графік часу роботи зазначених алгоритмів при кластеризації вибірки про іриси.

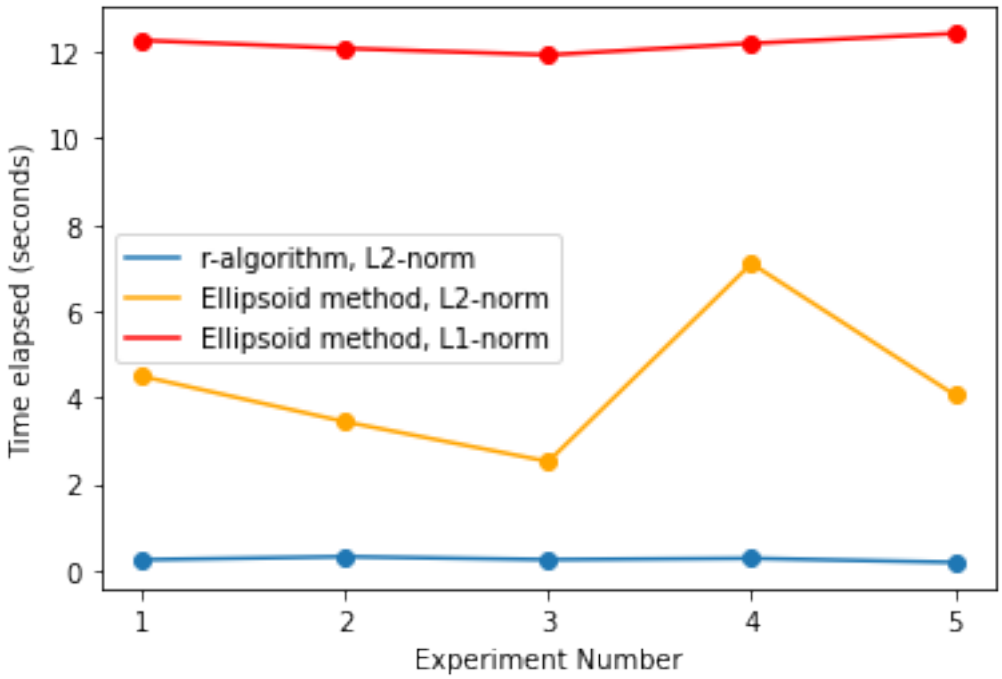


Рисунок 3.2 — Порівняльний графік швидкодії розглянутих методів

Варто зазначити, що принцип вибору початкового наближення положень центроїдів не впливають на час виконання кластеризації (рис. 3.3). Як при випадковому виборі, так і при застосування алгоритму k-means++ [25] кластеризація виконується за один і той же час. Приріст часу відсутній, бо більша частина ітерацій все одно відбувається у ярі коло мінімуму, і точніше початкове наближення скорочує лише кілька зайвих ітерацій на початку. Однак k-means++ корисний для вибору наближення, що з меншою ймовірністю приводить до локального мінімуму.

При порівнянні найшвидшого варіанту алгоритму на базі методу негладкої оптимізації з реалізацією k-means з відомого пакету scikit-learn виявляється, що класична бібліотечна реалізація алгоритму k-means працює швидше (рис. 3.4). Зі збільшенням розміру вибірки, розрив у часі також збільшується (рис. 3.5).

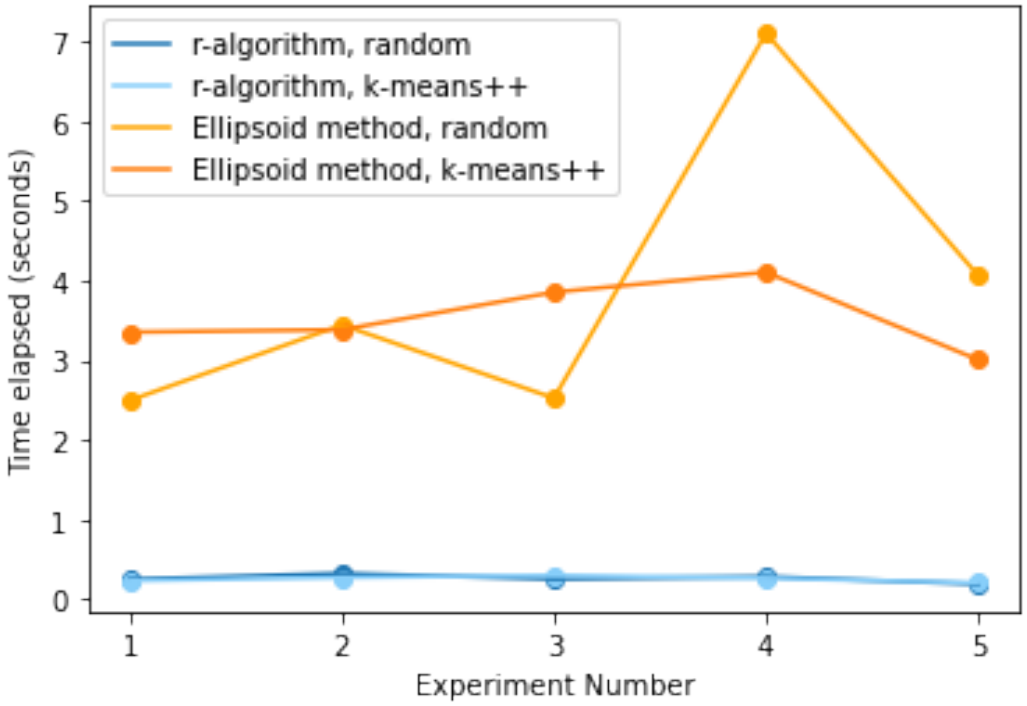


Рисунок 3.3 — Порівняльний графік різних підходів вибору початкових центроїдів (для всіх випадків використовувався критерій k-means)

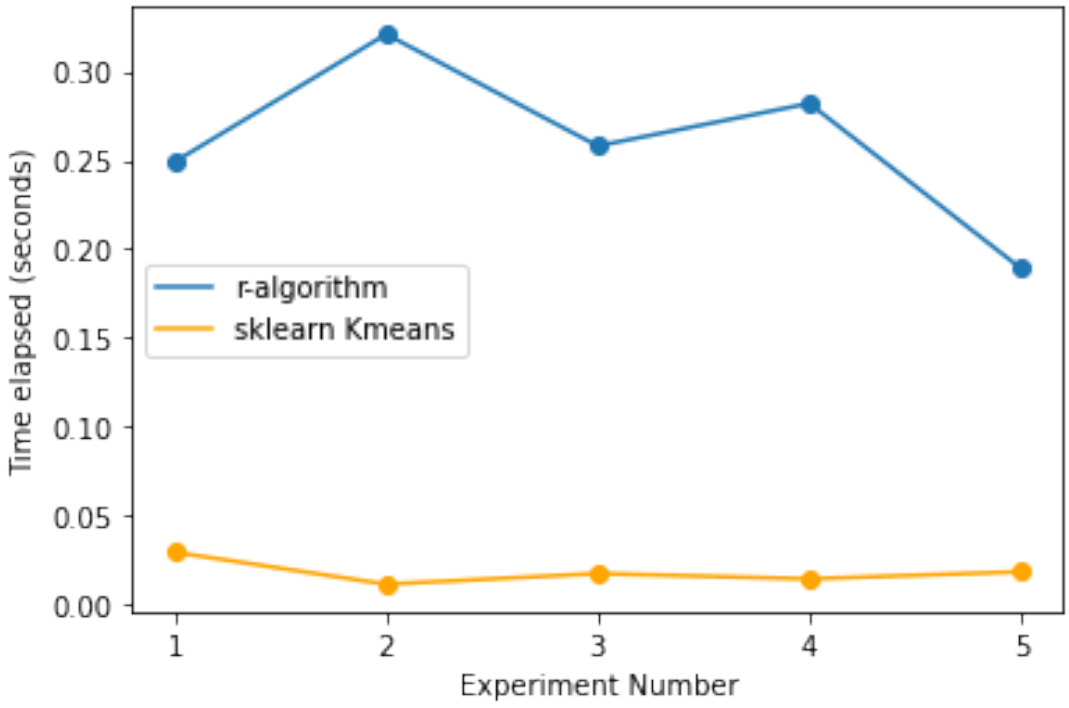


Рисунок 3.4 — Порівняння r-алгоритму та бібліотечної реалізації sklearn

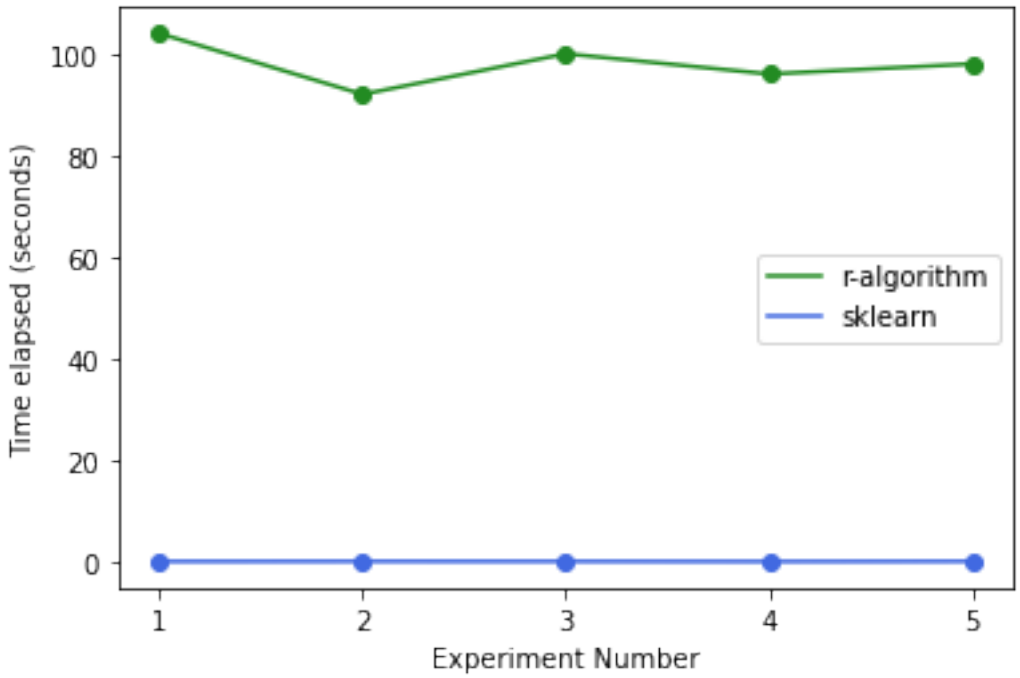


Рисунок 3.5 — Порівняння часу роботи r-алгоритму та sklearn з вибіркою обсягом 10 000 об’єктів

Для ще більших вибірок (100 000 екземплярів), час виконання перевищуватиме десять хвилин.

Певний вклад у часовий розрив між розглянутим алгоритмом і бібліотечною реалізацією вносить факт, що частина модулів scikit-learn використовують Cython, C та C++ [26], що значно прискорює роботу пакета. Однак, як можна бачити з графіка на рис. 3.5, час роботи бібліотечного k-means на великих вибірках значно не зростає, на відміну від методу на основі r-алгоритму, що наводить на думку, що мова програмної реалізації не є найголовнішим фактором.

В цілому, метод на основі r-алгоритму, що мінімізує критерій k-means, виявився найшвидшим серед розглянутих методів, і на невеликих вибірках працює не набагато гірше за класичний k-means. Також він непогано показав себе на вибірках більшого розміру: час, необхідний для проведення кластеризації, хоч і зріс, однак не настільки сильно, як для методу еліпсоїдів.

ВИСНОВКИ

У цій кваліфікаційній роботі було досліджено використання методів негладкої оптимізації для розв’язання задачі кластеризації. Зокрема, було з’ясовано, що вони придатні для розв’язання такої задачі, оскільки розглянуті цільові функції, що є критеріями якості кластеризації двох відомих алгоритмів k-means та p-medians, виявилися квазіопуклими і можуть бути мінімізовані такими методами.

Через наявні порушення опуклості і яружність цільових функцій застосовані в роботі методи, а саме метод еліпсоїдів та r-алгоритм, не завжди працювали максимально ефективно і швидко. Метод еліпсоїдів виявився дуже чутливим до опуклості і через її порушення дуже повільно мінімізував яружний критерій k-means, незважаючи на те, що він є субградієнтним методом з перетворенням простору, призначеним для роботи з функціями такого характеру. Більш стійким до таких функцій є r-алгоритм, і кластеризація за його допомогою при використанні критерію k-means відбувалася значно швидше на всіх розглянутих вибірках.

Спроба зменшити ступінь яружності цільової функції шляхом заміни евклідової норми на мангеттенську не була успішною, оскільки яружність зменшилась ціною збільшення перегинів, через що метод еліпсоїдів став збігатися ще повільніше, а r-алгоритм виявився зовсім непридатним через появу лінійно залежних послідовних субградієнтів.

Застосування методу штрафів до критерію k-means задля забезпечення опуклості теж не принесло бажаних результатів. Робота r-алгоритму теж ускладнилася наявністю лінійно залежних послідовних субградієнтів, а метод еліпсоїдів став працювати ще повільніше через значне збільшення розмірності вхідного вектора аргументів.

Представлена кваліфікаційна робота представляє як науковий, так і практичний інтерес. Було цікаво подивитися на типову задачу машинного

навчання з трохи іншого боку та спробувати застосувати інший підхід до її вирішення. З практичної точки зору результати цієї роботи можуть стати корисними при вирішенні задач з інших галузей машинного навчання, де виникають схожі функції, які треба оптимізувати. Вони далеко не завжди є диференційовними або навіть опуклими, тому важливо шукати нові або досліджувати придатність вже існуючих методів для оптимізації подібних функцій.

Зокрема, методи негладкої оптимізації можуть бути цікавими для вирішення задачі кластеризації, транспортних задач, дискретної оптимізації, нелінійної регресії тощо.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Géron A. Hands-on Machine Learning with Scikit-Learn and TensorFlow / Géron A.: США – O`Reilly Media Inc., 2017. – 23 с.
2. [Електронне джерело] Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Cluster_analysis
3. Cluster Analysis, 5th edition / [Everitt B.S., Landau S., Leese M., Stahl D.] – Велика Братанія – John Wiley & Sons, Ltd, 2011. – 15-142 с.
4. Silverman B.W. Density Estimation for Statistics and Data Analysis / Silverman B.W.: Лондон – Chapman and Hall CRC, 1986.
5. Kernel Smoothing / Wand M.P., Jones M.C.: Лондон – Chapman and Hall CRC, 1995.
6. [Електронне джерело] Режим доступу до ресурсу: https://www.csc2.ncsu.edu/faculty/nfsamato/practical-graph-mining-with-R/slides/pdf/Graph_Cluster_Analysis.pdf
7. Lloyd S. Least squares quantization in PCM / Lloyd S.: – IEEE Trans. Inf. Theory, 1982. – Vol. 28, 129-137 с.
8. Refining initial points for K-means clustering / Bradley P., Fayyad U.: – Machine Learning. Proceedings of the Fifteenth International Conference ICML’98. – 91-99 с.
9. Steinley D. Local optime in K-means clustering: What you don’t know may hurt you / Steinley D.: – Pscychological Methods – 8. – 294-304 с.
10. Density-based Clustering / [Kriegel H.-P., Kröger P., Sander J., Zimek A.]: – WIREs Data Mining and Knowledge Discovery – Vol.1. – 231-240 с.
11. A density-based algorithm for discovering for discovering clusters in large spatial database with noise / [Ester M., Kriegel H.-P., Sander J., Xu X., eds. Simoudis E., Han J., Fayyad U.]: – Proceeding of the Second International Conference on Knowledge Discovery and Data Mining, KDD’96. – 226-231 с.

12. OPTICS: Ordering Points to Identify the Clustering Structure / [Ankerst M., Breunig M.M., Kriegel H.-P., Sander J.]: – ACM SIGMOD international conference on Management of data 1999. – 49-60 с.
13. Жалдак М.І. Основи теорії і методів оптимізації: Начальний посібник \ Жалдак М.І., Триус Ю.В. — Черкаси: Брама-Україна, 2005 — 140-167 с.
14. [Електронне джерело] Режим доступу до ресурсу: https://see.stanford.edu/materials/lsocoee364b/01-subgradients_notes.pdf
15. Алгоритмы метода эллипсоидов для нахождения Lp-решения системы линейных уравнений / П.И. Стецюк, В.А. Стовба, И.С. Мартынюк // Теорія оптимальних рішень: Зб. наук. пр. — 2017. — № 2017. — 139-146 с. — Бібліогр.: 5 назв. — рос.
16. Шор Н.З. Метод отсечения с растяжением пространства для решения задач выпуклого программирования / Шор Н.З: – Кибернетика. 1977. №1. – 94-95 с.
17. Юдин Д.Б. Информационная сложность и эффективные методы решения выпуклых экстремальных задач / Юдин Д.Б., Немировский А.С.: – Экономика и математические методы. 1976. – Вып. 2. – 357-369 с.
18. Хачиян Л.Г. Полиномиальный алгоритм в линейном программировании / Хачиян Л.Г.: – Докл. АН СССР. 1979 – Том 244, №5. – 1093-1096 с.
19. [Електронне джерело] Режим доступу до ресурсу: https://web.stanford.edu/class/ee364b/lectures/ellipsoid_method_notes.pdf
20. Fisher R.A. The use of multiple measurements in taxonomic problems / Fisher R.A.: – Annals of Eugenics – Vol. 7. – 179-188 с.
21. Smith A. Handbook of Evolutionary Computation / Smith A., Coit D.W.: – Oxford University Press and Institute of Physics Publishing, 1996. – Section C 5.2
22. Стецюк П.І. Комп'ютерна програма "Octave-програма ralgb5a: r(α)-алгоритм з адаптивним кроком". Свідоцтво про реєстрацію авторського

права на твір №85010. Україна. Міністерство освіти і науки. Державний департамент інтелектуальної власності. Дата реєстрації 29.01.2019.

23. Субградієнтні алгоритми та задачі на комбінаторних конфігураціях / Стецюк П.І.. Донець Г.П., Ненахов Е.І.: – Київ: Унів. Вид-во ПУЛЬСАРИ, 2019. – 5-25 с.
24. Стецюк П.І. К вопросу сходимости r-алгоритмов / Стецюк П.І.: – Кибернетика и системный анализ, 1995 – №6. – 173-177 с.
25. k-means++: the advantages of careful seeding / Arthur D., Vassilvitskii S.: – Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics Philadelphia, PA, USA. – 1027-1035 с.
26. [Електронне джерело] Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Scikit-learn#Implementation