ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ОЛЕСЯ ГОНЧАРА
Факультет прикладної математики
Кафедра обчислювальної математики та математичної кібернетики

Курсова робота
на тему «Порівняльний аналіз алгоритмів кластеризації»

Перший (бакалаврський) рівень вищої освіти
Спеціальність 124 Системний аналіз
Освітня програма "Системний аналіз"

Виконавець
студент групи ПС–18–1
Каманцев Артем Сергійович

Керівник
асистент кафедри ОМ та МК
_____ О.С. Магас

Кількість балів _____
Оцінка за національною шкалою _____

Члени комісії:
_____ В.А. Турчина
_____ Л.Л. Гарт
_____ А. Є. Шевельова

2021

ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ОЛЕСЯ ГОНЧАРА
Факультет прикладної математики
Кафедра обчислювальної математики та математичної кібернетики

Курсова робота
на тему «Порівняльний аналіз алгоритмів кластеризації
DBSCAN, CMDD та SpectACl»

Перший (бакалаврський) рівень вищої освіти
Спеціальність 124 Системний аналіз
Освітня програма “Системний аналіз”

Виконавець
студент групи ПС–18–1
Каманцев Артем Сергійович

Керівник
асистент кафедри ОМ та МК
_____ О.С. Магас

Кількість балів _____
Оцінка за національною шкалою _____

Члени комісії:
_____ В.А. Турчина
_____ Л.Л. Гарт
_____ А. Є. Шевельова

2021

РЕФЕРАТ

Курсова робота: 14с., 4 рис., 3 табл., 20 джерел, 1 додаток.

Об’єкт дослідження: алгоритми кластеризації DBSCAN, CMDD та SpectACl.

Мета роботи: визначити ефективність застосування алгоритмів DBSCAN, CMDD та SpectACl в залежності від даних та цілей кластеризації.

Одержані висновки та їх новизна: одержані правила вибору найбільш придатного алгоритму серед DBSCAN, CMDD та SpectACl в залежності від даних та цілей кластеризації.

Результати дослідження можуть бути застосовані при визначенні доцільності кластеризації наявних даних за допомогою алгоритмів DBSCAN, CMDD та SpectACl.

Перелік ключових слів: КЛАСТЕРИЗАЦІЯ, КЛАСТЕРИЗАЦІЯ ОСНОВАНА НА ЩІЛЬНОСТІ, НАВЧАННЯ БЕЗ ВЧИТЕЛЯ, DBSCAN, CMDD, SpectACl, ПОРІВНЯЛЬНИЙ АНАЛІЗ.

ЗМІСТ

ВСТУП

Кластеризація – поділ даних на групи схожих об’єктів. Кожна група, що називається кластером, складається із об’єктів, які схожі між собою і не схожі з об’єктами інших груп. З точки зору машинного навчання, кластеризація відноситься до класу задач навчання без вчителя [1].

Алгоритми кластеризації відіграють важливу роль при аналізі даних і, згідно з [2], дозволяють:

1. Застосовувати до кожного кластеру окремий метод аналізу.
2. Стиснути вибірку, залишивши по одному найтиповішому представнику з кожного кластеру.
3. Виявити нетипові об’єкти, які не можна приєднати до жодного з кластерів.

Існує багато алгоритмів кластеризації а також способів їх класифікації. Розглянемо один з можливих варіантів такої класифікації:

1. Ієрархічні методи
    a. Об’єднувальні
    b. Роздільні
2. Методи розбиття
    a. Методи переміщення розбиття
    b. Розбиття на основі щільності
3. Методи на основі сітки
4. Методи кластеризації категоріальних даних
5. Інші методи

Ієрархічні методи будують ієрархію кластерів, яку називають дендрограмою. Виділяють об’єднувальні та роздільні ієрархічні методи. Роздільні методи починають роботу з одного кластеру, що складається з усіх даних, і розділяє його на менші кластери оптимальним чином. Об’єднувальні методи спочатку розміщують кожний об’єкт даних в окремий кластер, потім

кластери об’єднуються оптимальним чином. Процес продовжується до виконання критерію зупинки (найчастіше це бажана кількість кластерів). Представники цього класу алгоритмів: SLINK, COBWEB, CURE, CHAMELEON.

Методи на основі сітки розбивають простір об’єктів, над якими виконується кластеризація, на клітини (клітиною називають прямий Декартовий добуток окремих піддіапазонів атрибутів), а потім об’єднує клітини щоб побудувати кластери. Представники цього класу алгоритмів: BANG, STING, WaveCluster.

Категоріальні дані найчастіше представляються за допомогою арифметичних векторів, які мають такі особливості:

1. Мають велику розмірність
2. Більшість компонент таких векторів є порожніми
3. Для будь-якої пари об’єктів з множини, над якою проводиться кластеризація, відповідні арифметичні вектори, що їх представляють, мають дуже мало спільних значень

Звичайні методи кластеризації, засновані на мірі схожості, не працюють добре з такими даними. Але оскільки категоріальні дані є дуже важливими у задачах профілювання клієнтів, планування асортименту, веб аналітиці та інших, то були розроблені алгоритми: ROCK, SNN, CACTUS та ін.

На відміну від ієрархічних алгоритмів, алгоритми розбиття вивчають кластери безпосередньо. Серед алгоритмів розбиття виділяють методи переміщення розбиття та методи розбиття на основі щільності.

Серед методів переміщення розбиття можна виділити два основних підходи. Перший підхід визначає кожний кластер за допомогою певної моделі, параметри якої необхідно знайти. Наприклад, імовірнісна модель припускає, що дані походять з кількох популяцій, розподіл яких ми хочемо знайти. На такому підході засновані алгоритми: Expectation-Maximization (EM) method, SNOB, AUTOCLASS, MCLUST. Другий підхід полягає у визначенні цільової функції, яка залежить від розбиття. Наприклад, такою функцією може бути відстань до

деякого представника кожного кластеру. Такий підхід реалізують алгоритми: K-Medoids, K-Mean та інші [1].

Методи розбиття на основі щільності засновані на ідеї, що кластер у просторі даних це неперервна область з високою щільністю об’єктів, яка відділена від інших таких кластерів неперервними областями з низькою щільністю об’єктів. Об’єкти у областях, що розділяють кластери і мають низьку щільність, зазвичай вважають шумом [3]. Представники цього класу алгоритмів: DBSCAN [4], GDBSCAN, OPTIC, DBCLASD, DENCLUE [1].

Кожен із згаданих алгоритмів має свої переваги і недоліки, що обумовлюють область застосування кожного з них. Оскільки не існує найкращого алгоритму кластеризації у загальному випадку, то теми розробки нових чи вдосконалення вже існуючих алгоритмів кластеризації залишаються полем активних досліджень в останні роки, а тому і тема їх порівняльного аналізу залишається актуальною.

У даній роботі увагу приділено алгоритмам кластеризації на основі щільності, оскільки саме з класом цих алгоритмів пов’язана найбільша кількість публікацій в останні роки (згідно з результатами пошуку сервісу [5]). Скоріш за все, така увага до алгоритмів кластеризації на основі щільності обумовлена превагами, що властиві усім алгоритмам даного класу:

1. Можливість виявляти кластери довільної форми, у тому числі і кластери, що оточені іншими кластерами.
2. Алгоритмічна складність, яка у більшості алгоритмів є меншою квадратичної ($O(n^2)$).
3. Досить ефективна робота із даними великої розмірності.
4. Більшість алгоритмів можуть бути виконані паралельно.

На даний момент вже існує низка порівняльних аналізів алгоритмів кластеризації на основі щільності. Наприклад у роботі [6] порівнюються такі класичні алгоритми як: DBCLASD, DENCLUE, DBSCAN. У [7] методи на основі щільності порівнюються із такими класами методів кластеризації як: ієрархічні

методи, методи розбиття та методи на основі сітки. За алгоритмічною складністю порівнюються алгоритми DBSCAN, FDBSCAN, LDBSCAN, VDBSCAN, ST-DBSCAN у роботі [8].

Найбільш відомим представником класу алгоритмів кластеризації на основі щільності є DBSCAN, проте його основним недоліком є неможливість знаходити кластери, що мають різну щільність даних. Для вирішення цієї проблеми у роботі [9] було запропоновано новий алгоритм – CMDD. Цей алгоритм складається з таких кроків:

1. Оцінюється локальна щільність для кожного об'єкту даних як сума відстаней до k найближчих сусідів (де k – параметр алгоритму)
2. Об'єкти даних сортуються за локальною щільністю у порядку зростання
3. Алгоритм перебирає об'єкти у порядку зростання їх локальної щільності. Якщо поточний об'єкт ще не належить до жодного кластеру, то він утворює новий кластер, до якого додаються об'єкти, що мають локальну щільність близьку до локальної щільності першого об'єкту у кластері

У роботі [10] запропоновано алгоритм SpectACl, який комбінує переваги алгоритмів спектральної кластеризації і DBSCAN, полегшуючи недоліки кожного з методів. Метод знаходить кластери, що мають велику середню щільність, де відповідна щільність для кожного кластера автоматично визначається через спектр матриці суміжності. Алгоритм складається з таких кроків:

1. Обчислити матрицю суміжності W, за правилом: $W_{j,l} = 1$ тоді і тільки тоді, коли об'єкт l належить ε-околу об'єкту j, де ε – параметр алгоритму.
2. Знайти d власних векторів матриці W, що відповідають d найбільшим власним значенням матриці, та записати їх у стовпці допоміжної матриці V, яка має розмір $n \times d$, де n – кількість об'єктів даних.

3. Обчислити допоміжну матрицю U за правилом: $U_{j,k} = |V_{j,k}| \cdot |\lambda_k|^{\frac{1}{2}}$. Матриця U має розмір $n \times d$.
4. Застосувати до матриці U алгоритм k-mean та знайти r кластерів, де r – параметр алгоритму.

Алгоритм DBSCAN будемо використовувати як відправну точку при порівняльному аналізі. Тому об’єкт, предмет, мета, завдання та метод дослідження мають наступні формулювання.

Об’єкти дослідження – алгоритми кластеризації DBSCAN, CMDD, SpectACl.

Предмет дослідження – ефективність алгоритмів DBSCAN, CMDD та SpectACl.

Мета дослідження – визначити ефективність застосування алгоритмів DBSCAN, CMDD та SpectACl в залежності від даних та цілей кластеризації.

Завдання дослідження - обрати параметри та метрики для порівняння алгоритмів, обрати спосіб візуалізації багатовимірних даних, провести комп’ютерний експеримент та проаналізувати отримані результати.

Метод дослідження – аналіз даних.

ПОСТАНОВКА ЗАДАЧІ

За наявними даними необхідно:

1. Провести кластеризацію за допомогою алгоритмів DBSCAN, CMDD та SpectACl, тобто знайти таке розбиття об’єктів даних на групи, щоб в кожній групі знаходились об’єкти які схожі між собою і не схожі з об’єктами інших груп.
2. За допомогою вірних ідентифікаторів, що вказують, до яких кластерів насправді належать об’єкти даних, визначити якість кластеризації.
3. Визначити параметри розглянутих алгоритмів кластеризації, що впливають на ефективність їх практичного застосування, та порівняти алгоритми за цими параметрами.

Дані:

1. Синтетичні дані. 1500 арифметичних векторів довжини 3, що представлені матрицею $D_1^{1500\times3}$, де перші два стовпці – координати x і y точки у прямокутній декартовій системі координат відповідно, а третя – ідентифікатор кластеру, до якого належить точка. Кожний кластер містить рівну кількість точок.

   Точки отримані за допомогою методу make_blobs пакету datasets бібліотеки Scikit-learn [11] із такими параметрами:

$$n\_samples = 1500,$$
$$cluster\_std = [1.0, 2, 0.5],$$
$$centers = [[-8.95, -5.46], [-4.59, 0.09], [1.94, 0.51]]$$

   Дані представляють собою три кулі на площині, що мають різне стандартне відхилення. Вектори координат точок є випадковими векторами що мають нормальний розподіл, тому при кожній новій генерації синтетичних даних отримуємо дещо різні вектори.

2. Реальні дані. 682 арифметичних векторів довжини 10, що представлені матрицею $D_2^{682\times10}$, де $1-9$ стовпці - результати аналізів клітин молочної залози, а 10 стовпець складається з цифр 2 та 4, що означають результати дослідження.

   Матриця D_2 отримана на основі даних «breast-cancer-wisconsin.data» з джерела [12]. Стовпці матриці D_2 відповідають стовпцям оригінальних даних $2-11$, рядки матриці D_2 отримані з рядків оригінальних даних шляхом видалення записів, що містять пропущені дані (16 рядків). До кластеру з ідентифікатором 2 належать 443 об’єкти, до кластеру з ідентифікатором 4 – 239 об’єктів.

1 Формальна постановка задачі

Нехай X – множина об’єктів, Y – множина номерів (імен, міток) кластерів. Задано функцію відстані між об’єктами ρ. Наявна скінчена вибірка об’єктів $X^m = \{x_1, x_2, \dots, x_m\} \subset X$. Необхідно розбити вибірку на неперетинні підмножини, що називаються кластерами, так, щоб кожний кластер складався з об’єктів, близьких за метрикою ρ, а об’єкти різних кластерів суттєво відрізнялись за цією метрикою. При цьому кожному об’єкту $x_i \in X^m$ приписується номер кластеру $y_i \in Y$.

У якості функції у роботі використовується евклідова метрика.

Алгоритм кластеризації – функція $f: X \to Y$, яка будь-якому об’єкту $x \in X$ ставить у відповідність номер кластеру $y \in Y$. Множина Y у деяких випадках відома заздалегідь, однак частіше ставиться задача визначити оптимальну кількість кластерів, з точки зору того чи іншого критерію якості.

Кластеризація (навчання без вчителя) відрізняється від класифікації (навчання з учителем) тим, що мітки вихідних об'єктів y_i спочатку не задані, і навіть може бути невідомою сама множина Y [2].

2 Критерії та метрики для порівняння алгоритмів кластеризації

Для оцінки якості кластеризації існує дві найрозповсюдженіші метрики: скорегований індекс Ранду (Adjusted Rand Index - ARI) і скорегована взаємна інформація (Adjusted Mutual Information - AMI). У роботі [13] автори рекомендують використовувати ARI у випадку рівних за обсягом (збалансованих) кластерів, а AMI у випадку незбалансованих кластерів. Оскільки коректна оцінка кластеризації реальних даних важливіша, ніж синтетичних і оскільки в кластери у реальних даних є не збалансованими, то у якості метрики будемо використовувати AMI, а тому розглянемо її детальніше.

AMI є мірою взаємної залежності між двома випадковими змінними, що має скорегований ефект згоди при випадковій кластеризації. AMI приймає значення один, коли розподіли за кластерами ідентичні і нуль, коли розподіл за кластерами еквівалентний випадковому. Детальні теоретичні обґрунтування AMI міри можна знайти у статті [14]. Обчислення ARI у даній роботі виконано за допомогою функції adjusted_rand_score пакету metrics бібліотеки Scikit-learn [11].

Окрім ефективності самого алгоритму кластеризації, на можливість його успішного застосування на практиці впливають ще такі їх властивості та параметри:

1. Можливість знаходити кластери довільної щільності.
2. Можливість виявляти шум у даних.
3. Кількість параметрів алгоритму, їх інтуїтивна інтерпретація та легкість налаштування.
4. Алгоритмічна складність.

Тому у подальшому, буде наведено не тільки оцінку ефективності алгоритмів кластеризації за метрикою AMI, але і буде проведено їх порівняння за переліченими властивостями та параметрами.

3 Візуалізація багатовимірних даних

Візуалізувати дані, що представлені арифметичними векторами довжини три або менше не є складною задачею, проте арифметичні вектори набору реальних даних мають довжину дев’ять (не враховуючи останні компоненти векторів, оскільки вони позначають істинний кластер, до якого відноситься той чи інший вектор і не приймають участь у кластеризації) і такі дані неможливо відобразити на площині без попередньої обробки.

Існує багато різних підходів для зменшення розмірності даних. Наприклад у бібліотеці Scikit-learn [11] для вирішення цієї задачі існують алгоритми: аналіз головних компонент, випадкова проекція та об’єднувальна ієрархічна кластеризація. Серед цих алгоритмів було обрано аналіз головних компонент, оскільки він дав найкращі практичні результати.

Графіки було побудовано з використанням бібліотеки Plotly [15].

4 Програмна реалізація

У роботі використано реалізацію алгоритму DBSCAN з бібліотеки Scikit-learn [11] (пакет cluster).

Реалізація алгоритму SpectACl використана авторська, код є публічно доступним [16].

Код алгоритму CMDD не є доступним публічно, тому було створено власну реалізацію цього алгоритму (див. Додаток А Програмна реалізація алгоритму CMDD). У джерелі [17] було знайдено набір даних, схожий на один з тих, що використовувалися у [9]. Кластеризацію цих даних було проведено алгоритмом CMDD з параметрами, що використовувались для цих даних у [9], а саме: $k = 13, minpts = 4$ (Рисунок 4.1). Оцінка кластеризації AMI досягла одиниці (тобто максимуму), отже власну реалізацію алгоритму CMDD можна вважати коректною.

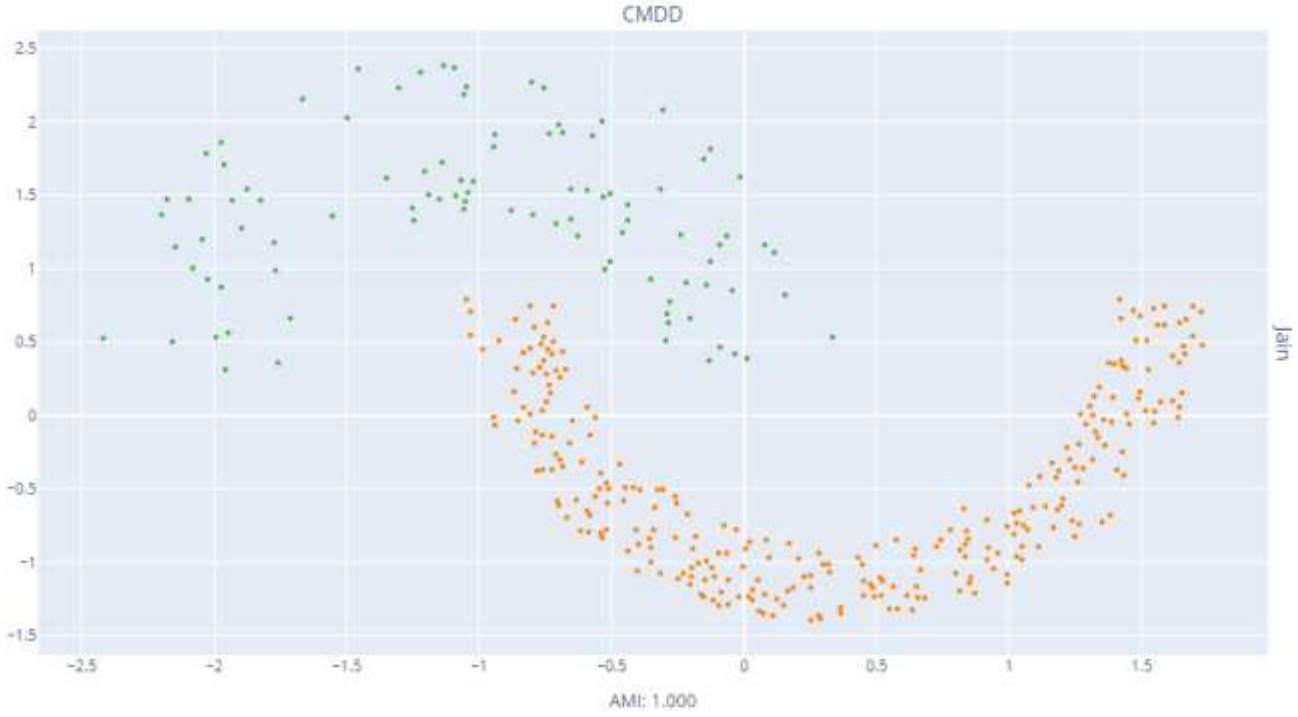


Рисунок 4.1 - Результат кластеризації набору даних «Jain» з джерела [17] за допомогою власної реалізації алгоритму CMDD. $k = 13,\ minpts = 4$.

5 Комп’ютерний експеримент

У роботі використано бібліотеку Scikit-learn версії 0.24.2. Параметри методів, що не вказані у роботі приймають значення за замовченням згідно з версією бібліотеки.

Існує дві варіації алгоритму SpectACl: з нормалізацією матриці суміжності ε-графу найближчих сусідів та без неї. Варіант з нормалізацією має гірші результати як на синтетичному так і на реальному наборі даних. Такі результати співпадають з експериментальними результатами, що були отримані авторами даного алгоритму, тому версія із нормалізацією у подальшому розглядатися не буде.

Параметри алгоритмів, за яких кожен з них досягає максимальної AMI метрики було визначено шляхом перебору для кожного набору даних. Експерименти проводяться за цими знайденими параметрами.

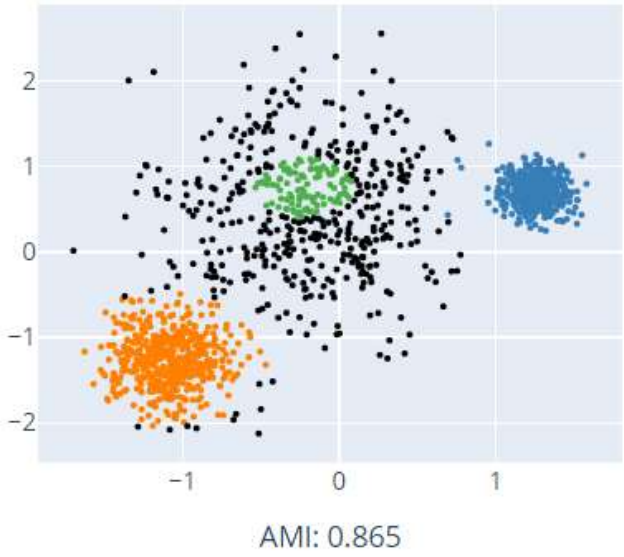
Для алгоритму аналізу головних компонент параметр $random\_state$ було зафіксовано рівним 42.

Перед кластеризацією усі дані завжди стандартизуються за допомогою класу StandardScaler бібліотеки Scikit-learn [11].

5.1 Синтетичні дані

Оскільки синтетичний набір даних генерується кожного разу дещо інший, то розглянемо два визначні результати експерименту: Рисунок 5.1, Рисунок 5.2.

а. Синтетичні дані



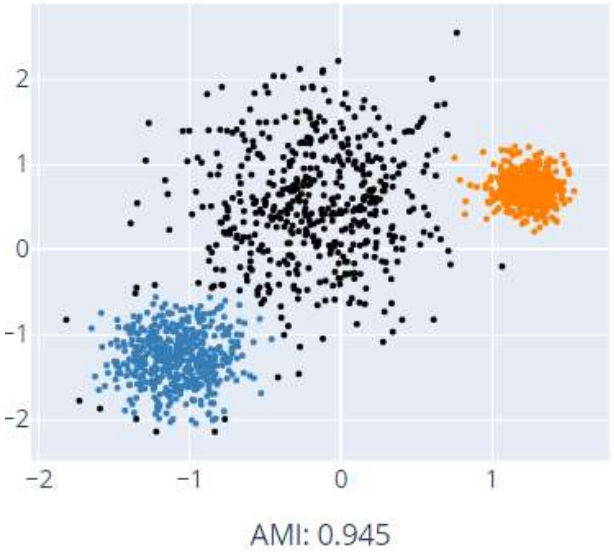
б. DBSCAN: Minpts=85, Eps=0.3

в. CMDD: Minpts=7, K=20

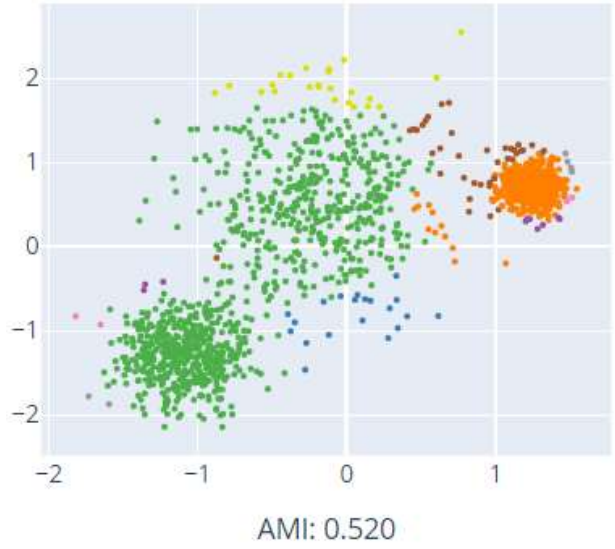
г. SpectACl: N_clusters=3, Eps=0.5

Рисунок 5.1 - Результат кластеризації синтетичного набору даних за допомогою алгоритмів DBSCAN, CMDD, SpectACl. Перший визначний результат експерименту

а. Синтетичні дані



б. DBSCAN: Minpts=85, Eps=0.3



в. CMDD: Minpts=7, K=20

г. SpectACl: N_clusters=3, Eps=0.5

Рисунок 5.2 – Результат кластеризації синтетичного набору даних за допомогою алгоритмів DBSCAN, CMDD, SpectACl. Другий визначний результат експерименту.

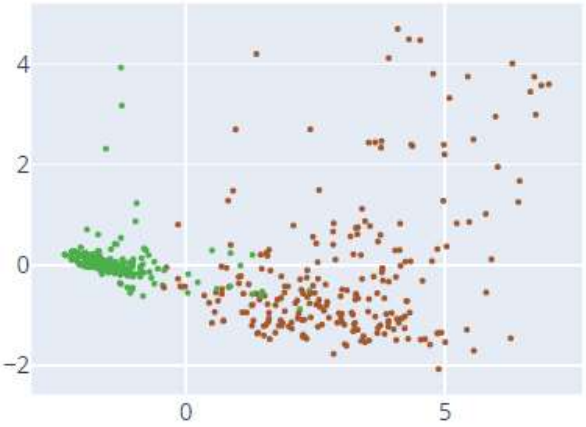
На ста генераціях синтетичного набору даних було обчислено середнє значення та дисперсію AMI метрики для кожного алгоритму. Результати обчислень наведені у Таблиці 5.1.

Таблиця 5.1- AMI метрика при кластеризації синтетичних даних

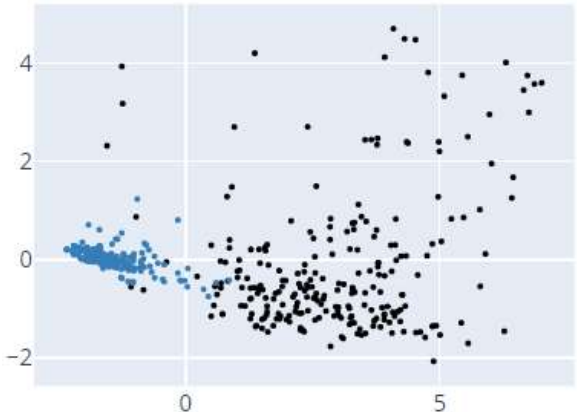
AMI	DBSCAN	CMDD	SpectACl
Середнє значення	0.923	0.905	0.968
Дисперсія	$1.7 \cdot 10^{-3}$	$3.6 \cdot 10^{-3}$	$0.06 \cdot 10^{-3}$

5.2 Реальні дані

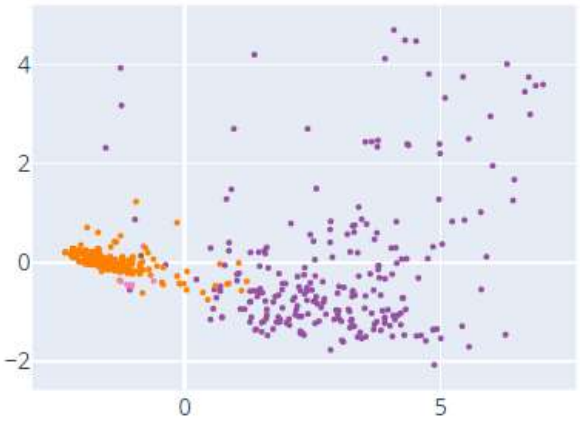
Графічні результати кластеризації реальних даних зображені на Рисунку 5.3.



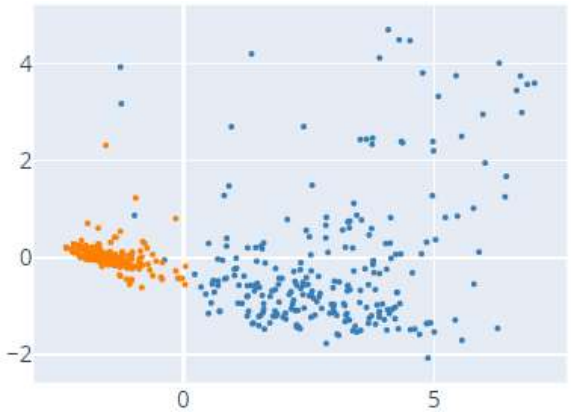
а. Реальні дані



б. DBSCAN: Minpts=11, Eps=1.55



в. CMDD: Minpts=12, K=71



г. SpectACl: N_clusters=2, Eps=3.01

Рисунок 5.3 - Результати кластеризації реального набору даних за допомогою алгоритмів DBSCAN, CMDD, SpectACl

Значення AMI метрики при кластеризації синтетичних даних наведено у Таблиці 5.2

Таблиця 5.2 – AMI метрика при кластеризації реальних даних

	DBSCAN	CMDD	SpectACl
AMI	0.852	0.813	0.869

6 Порівняння властивостей та параметрів

Усі розглянуті алгоритми мають параметри, що допомагають їх налаштувати для застосування на конкретному наборі даних. Важливою властивістю цих параметрів є можливість їх інтуїтивної інтерпретації, бо істині класи об’єктів даних є невідомими при практичному застосуванні алгоритмів кластеризації. Тому стає значно складніше оцінити якість кластеризації і підібрати найкращі параметри шляхом перебору стає складніше.

Для розгляду параметрів алгоритмів DBSCAN та CMDD необхідно ввести наступне означення:

Ядровий об’єкт– такий об’єкт даних, приналежність до деякого кластеру якого вже є визначеною і відносно якого в подальшому алгоритм буде намагатись розширити кластер.

Параметри алгоритму DBSCAN:

1. Eps – об’єкти, що мають відстань до одного з ядрових об’єктів кластеру меншу за значення цього параметру, включаються у цей кластер.
2. $Minpts$ – об’єкт, що належить деякому кластеру і має у своєму Eps-околі не менше ніж $Minpts$ об’єктів, є ядровим об’єктом.

Для розгляду параметрів алгоритму CMDD введемо ще одне означення:

$D_i(K)$ – сума відстаней до K найближчих сусідів об’єкту, який було додано до кластера i першим.

Параметри алгоритму CMDD:

1. K – якщо сума відстаней до p найближчих сусідів деякого ядрового об’єкту c кластеру i є меншою за $D_i(K)$, то кожний з p найближчих сусідів об’єкту c включається до кластера i.
2. $Minpts$ – об’єкт, що належить деякому кластеру i та має суму відстаней до $Minpts$ своїх найближчих сусідів меншу від $D_i(K)$, є ядровим об’єктом.

Параметри алгоритму SpectACl:

1. N_clusters – кількість кластерів, що необхідно знайти.
2. Eps – визначає Eps-граф найближчих сусідів, матриця суміжності якого використовується аналогічно методу спектральної кластеризації.
3. D – визначає вимірність простору, у якому буде відбуватись фінальна кластеризація за допомогою методу k-mean.

Питання про те, наскільки зрозумілим є зміст розглянутих параметрів є досить суб'єктивним, проте на мою думку, за зменшенням очевидності практичного сенсу параметрів алгоритми можна впорядкувати наступним чином:

1. DBSCAN
2. CMDD
3. SpectACl

Проте варто зазначити, що алгоритм SpectACl є досить стійким відносно своїх параметрів (зміна параметрів не суттєво впливає на якість результатів) [10].

Окрім параметрів є й інші властивості алгоритмів кластеризації, що впливають на можливість ефективно застосовувати їх на практиці. Відомості щодо цих властивостей неведені у Таблиці 6.1.

Таблиця 6.1 – Властивості алгоритмів кластеризації

	DBSCAN	CMDD	SpectACl
Знаходить кластери довільної щільності	Ні	Так	Так
Виявляє шум	Так	Ні (з модифікацією Так)	Ні
Алгоритмічна складність	$n \cdot \log(n)$	$n \cdot \log(n) + n \cdot m \cdot K$	$D \cdot n^2 + n \cdot D \cdot C \cdot I$

Де:

1. n – кількість об’єктів даних
2. m – вимірність даних
3. K – відповідний параметр алгоритму CMDD
4. D – відповідний параметр алгоритму SpectACl
5. C – кількість класів, на які не обхідно кластеризувати об’єкти
6. I – максимальна кількість ітерацій алгоритму k-mean при фінальній кластеризації алгоритму SpectACl

7 Порівняльний аналіз алгоритмів DBSCAN, CMDD, SpectACl

Синтетичні дані були досить важкими для усіх розглядуваних алгоритмів з таких причин:

1. Кластери мають різне стандартне відхилення, що є складною ситуацією для DBSCAN (див. Рисунок 5.1.б).
2. Кластери не завжди мають достатньо чітке зниження щільності на краях, що є складним для CMDD (див. Рисунок 5.2.в).
3. Дані представлені випадковими векторами, а отже в наявності є шум, що ускладнює роботу SpectACl.

Незважаючи на складність даних, усі алгоритми показали приблизно однакове високе середнє значення AMI метрики. Під час експерименту SpectACl стабільно демонстрував найкращі середні значення, тоді як DBSCAN та CMDD змінювали лідерство.

Значення дисперсії можна розглядати як показник стійкості алгоритмів відносно своїх параметрів. Вважатимемо алгоритм стійким відносно своїх параметрів, якщо він не суттєво змінює результат при незначній зміні параметрів або даних. З експерименту бачимо, що найбільшу стійкість відносно своїх параметрів має SpectACl, значно меншу DBSCAN і ще трохи меншу CMDD.

На реальних даних алгоритми показали результат аналогічний до результату на синтетичних даних: найточнішим виявився SpectACl, менш точним був DBSCAN і ще менш точним - CMDD.

Варто зазначити, що незважаючи на те, що і синтетичні і реальні дані містили кластери зі змінною щільністю DBSCAN продемонстрував досить добрі результати. Це пояснюється тим, що і для реальних і для синтетичних даних поріг щільності для DBCAN був обраний трохи меншим за щільність найменш щільного кластеру. Тому найменш щільний кластер найчастіше повністю був помічений алгоритмом як шум, і таким чином утворював власний коректний кластер. Два кластери синтетичних даних, що не є найменш щільними, розділені просторово тому також не становили складності для DBSCAN з того моменту,

як його поріг щільності було встановлено достатнім для виявлення обох кластерів.

Хоч зміст параметрів алгоритму SpectACl не є очевидним, проте алгоритм є дуже стійким відносно своїх параметрів. Тому неочевидний зміст параметрів не можна назвати недоліком алгоритму. Алгоритми DBSCAN та CMDD мають доволі зрозумілу інтуїтивну інтерпретацію свої параметрів, що є їх перевагою.

При наявності у даних кластерів, що не є найменш щільними, проте дотикаються один до одного і мають різну щільність, DBSCAN не дає задовільних результатів, тому для таких даних перевагу слід надавати алгоритмам CMDD та SpectACl.

Якщо є необхідність виявляти шум у даних, то SpectACl не підходить для виконання цієї задачі. CMDD може виявляти шум, тільки якщо його модифікувати так, щоб шумом вважались невеликі кластери. DBSCAN може добре виявляти шум.

За будь-якої розмірності даних найкращу алгоритмічну складність має DBSCAN. Якщо дані мають великий обсяг але невелику розмірність, то CMDD є ефективнішим від SpectACl, але при невеликих за обсягом проте значних за розмірністю даних, SpectACl є ефективнішим за CMDD. Це пояснюється тим, що CMDD використовує евклідову відстань, тому його алгоритмічна складність зростає зі збільшенням розмірності даних.

Варто відзначити, що найбільш складним етапом для алгоритмів DBSCAN та SpectACl є пошук об’єктів, що належать -околу деякого об’єкту, а для CMDD – пошук найближчих сусідів деякого об’єкту. Тому їх ефективність значною мірою залежить від ефективності виконання цих операцій. Для ефективного виконання названих операцій часто використовують такі структури даних: $R^* - tree$ [18] , k-d tree [19], ball tree [20] та їх модифікації.

ВИСНОВКИ

Отже, проведений аналіз дає можливість зробити наступні висновки:

1. Алгоритми CMDD та SpectACl є гарною альтернативою DBSCAN якщо відомо, що дані мають кластери різної щільності і кластери, які мають не найменшу щільність, дотикаються один до одного.
2. CMDD має інтуїтивно зрозумілі параметри і їх визначення за допомогою експертної оцінки може іноді дозволити досягти високої точності. З незначною модифікацією алгоритм може бути застосований для кластеризації даних, у яких необхідно виявити шум. Алгоритмічна складність є непоганою для великих за обсягом вибірок, проте вона не є стійкою до вимірності даних і алгоритм є менш точним від SpectACl
3. SpectACl може продемонструвати найбільш точні результати, має алгоритмічну складність досить стійку до вимірності, проте алгоритмічна складність є великою для великих за обсягом вибірок і алгоритм не здатний виявляти шум у даних.

На основі цих висновків можна сформувати наступні рекомендації при виборі між алгоритмами кластеризації DBSCAN, CMDD, SpectACl:

1. DBSCAN доречно використовувати, якщо максимальна точність не є необхідною і в даних немає кластерів зі змінною щільністю або якщо кластери, що не є найменш щільними не дотикаються один до одного.
2. CMDD слід обирати, якщо:
    а. Необхідно виявити шум у даних.
    б. Вважливою є можливість інтуїтивної інтерпретації параметрів алгоритму.
    в. Розмірність даних не дуже велика, а точність не є критичною.
2. SpectACl буде найкращим вибором якщо:
    а. Обсяг даних невеликий.
    б. Необхідно досягти максимальної точності.
    в. Вимірність даних велика.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Berkhin P. A Survey of Clustering Data Mining Techniques // Grouping Multidimensional Data. Springer, Berlin, Heidelberg. 2006. P. 25-71. DOI: 10.1007/3-540-28349-8_2.
2. Кластеризация. Дата оновлення: 24.12.2011. URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F (дата звернення: 02.06.2021].
3. Sander J. Density-Based Clustering // Encyclopedia of Machine Learning. Springer, Boston, MA. 2011. DOI: 10.1007/978-0-387-30164-8_211.
4. A density-based algorithm for discovering clusters in large spatial databases with noise / Ester M., Kriegel H. P., Sander J., Xiaowei Xu // Proceedings of the second international conference on knowledge discovery and data mining, Portland, 2-6 Aug 1996. United States, 31.12.1996, PB: 405 p. URL: https://www.osti.gov/biblio/421283.
5. Google Scholar // Google. URL: https://scholar.google.com.ua/ (дата звернення 02.06.2021).
6. Pooja Batra Nagpal, Priyanka Ahlawat Mann. Comparative Study of Density based Clustering Algorithms // International Journal of Computer Applications. August 2011. № 27. P. 44-47.
7. Mihika S., Sindhu N. A Survey of Data Mining Clustering Algorithms // International Journal of Computer Application. NY, USA, October 2015. №128. P. 1-5. DOI: 10.5120/ijca2015906404.
8. Hardik P. C., Prof. Shakti V. P. A Survey – Time Complexity of Density based clustering Algorithms // *International Journal of Engineering Development and Research.* 2015. № 3. P. 26-30.
9. Fahim A. Clustering Algorithm for Multi-density Datasets // *Romanian Journal of Information Science and Technology*. December 2019. №22. P. 244-258.
10. The SpectACl of Nonconvex Clustering: A Spectral Approach to Density-Based Clustering / Proceedings of the AAAI Conference on Artificial Intelligence. July 2019. № 33. P. 3788-3795. DOI: 10.1609/aaai.v33i01.33013788.

11. Scikit-learn: machine leaning in python. URL: https://scikit-learn.org/stable/ (дата звернення: 02.06.2021).

12. Breast Cancer Wisconsin (Diagnostic) Data Set // UCI repository. Дата оновлення: листопад 1995. URL: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic) (дата звернення: 02.06.2021).

13. Adjusting for Chance Clustering Comparison Measures / Romano S., Nguyen X. V., Bailey J., Verspoor K. // *Journal of Machine Learning Research.* 2016. №17. P. 134:1-134:32.

14. Vinh N., Epps J., Bailey J. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance // Journal of Machine Learning Research. October 2010. №11. P. 2837-2854.

15. Plotly: The front end for ML and data science models // Plotly. URL: https://plotly.com (дата звернення: 02.06.2021).

16. The SpectACl of Nonconvex Clustering: A Spectral Approach to Density-Based Clustering // Startseite der Universität Dortmund. URL: https://sfb876.tu-dortmund.de/spectacl/ (дата звернення: 02.06.2021).

17. Clustering datasets // University of Eastern Finland. URL: http://cs.joensuu.fi/sipu/datasets/ (дата зверення: 02.06.2021).

18. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles / Beckmann N., Kriegel H.-P., Schneider R., Seeger B. // SIGMOD Rec. New York, USA, May 1990. №19. P. 322-331. DOI: 10.1145/93605.98741.

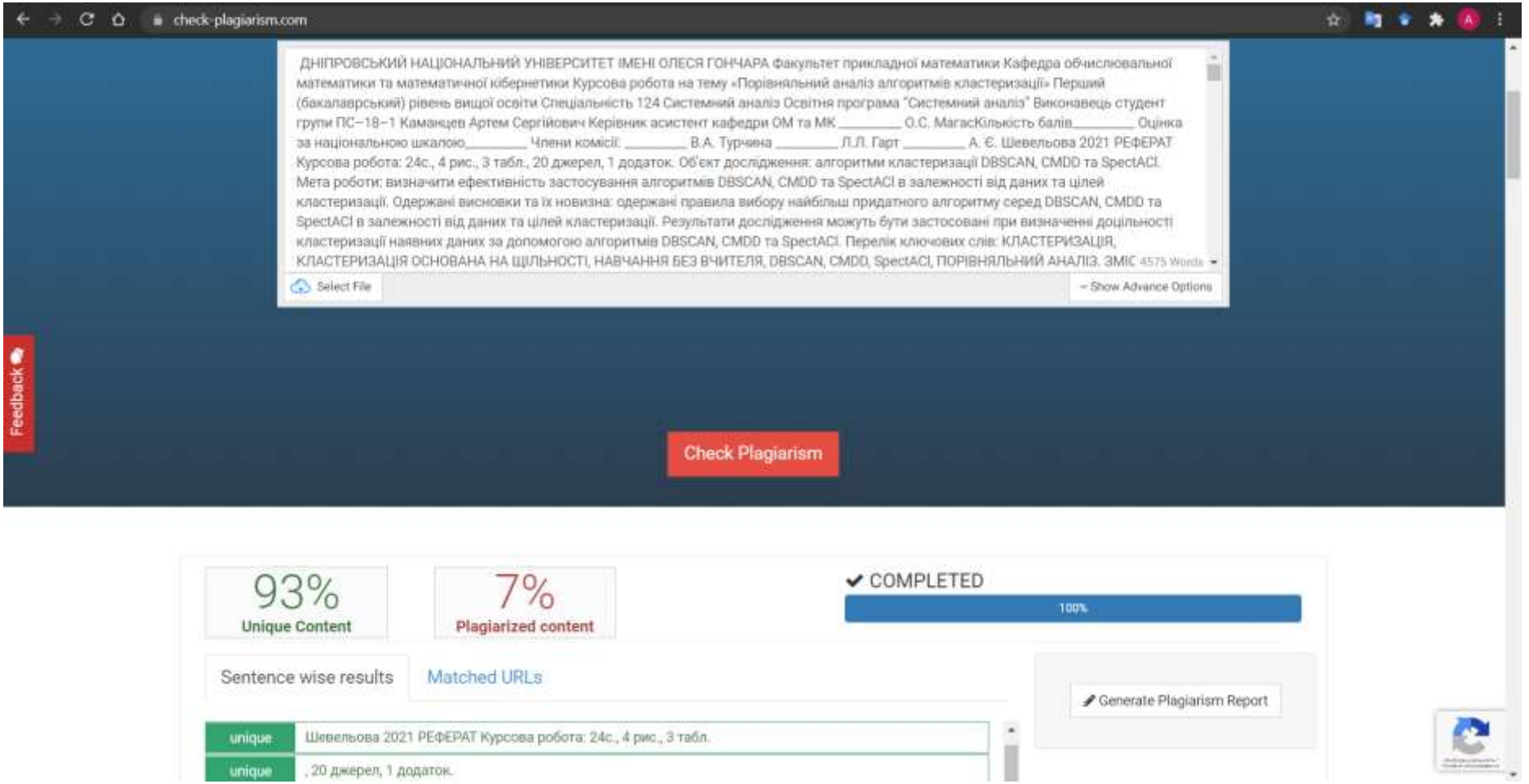
19. Bentley J. L. Multidimensional Binary Search Trees Used for Associative Searching // Commun. ACM. New York, USA, September 1975. №18. P. 509-517. DOI: 10.1145/361002.361007.

20. Omohundro S. Five Balltree Construction Algorithms // ICSI Technical Report TR-89-063. Berkeley, California, USA, 1989. URL: http://www.icsi.berkeley.edu/ftp/global/pub/techreports/1989/tr-89-063.pdf.

Додаток А Програмна реалізація алгоритму CMDD

```
import numpy as np
from rtree import index

UNCLASSIFIED_ID = None


class CMDD:
    def __init__(self, k, minpts):
        if minpts > k:
            raise Exception('Invalid arguments. k cannot be greater than
minpts!')
        self.k = k
        self.minpts = minpts
        self.dataset = None

    def fit(self, X):
        objects_count = X.shape[0]
        dataset = DataSet(X, self.k)

        # calculate densities
        id_density = []
        for i in range(objects_count):
            id_density.append((
                i,
                dataset.distance(i, self.k)
            ))

        cluster_id = 1
        id_density_sorted = list(sorted(id_density, key=lambda t: t[1]))
        for point_identifier, point_k_den in id_density_sorted:
            if dataset.get_class_id(point_identifier) == UNCLASSIFIED_ID:
                self.expand_cluster(dataset, point_identifier, cluster_id,
point_k_den)
                cluster_id += 1

        self.dataset = dataset

    def fit_predict(self, X):
        self.fit(X)

        return self.dataset.result

    def expand_cluster(self, dataset, point_identifier, cluster_id,
point_k_den):
        seeds = [point_identifier]

        dataset.change_cluster_ids(seeds, cluster_id)
        while len(seeds) > 0:
            current_identifier = seeds.pop(0)
            result = dataset.region_query(current_identifier)
            if dataset.distance(current_identifier, self.minpts) <= point_k_den:
                for i, k_neighbour_id in enumerate(result):
                    if dataset.distance(current_identifier, i + 1) >
point_k_den:
                        break
                    if dataset.get_class_id(k_neighbour_id) == UNCLASSIFIED_ID:
                        seeds.append(k_neighbour_id)
                        dataset.change_cluster_ids([k_neighbour_id], cluster_id)
```

```
class DataSet:
    def __init__(self, dataset, k):
        self.result = [UNCLASSIFIED_ID] * dataset.shape[0]
        self.__init_distances(dataset, k)

    def __init_distances(self, dataset, k):
        rtree_property = index.Property()
        rtree_property.dimension = dataset.shape[1]
        rtree_index = index.Index(properties=rtree_property)
        for i, point in enumerate(dataset):
            rtree_index.add(i, (*point, *point))

        data = [[]] * dataset.shape[0]
        for i, point in enumerate(dataset):

            k_nearest = list(rtree_index.nearest((*point, *point), k + 1))
            if k_nearest[0] == i:  # removes current point
                k_nearest = k_nearest[1:]
            k_nearest = k_nearest[:k]  # removes excess if present
            data_item = [[0, 0] for i in range(k)]
            data_item[0] = [k_nearest[0], np.linalg.norm(point -
dataset[k_nearest[0]])]
            for j, point_identifier in list(enumerate(k_nearest))[1:]:
                data_item[j] = [
                    point_identifier,
                    data_item[j - 1][1] + np.linalg.norm(point -
dataset[point_identifier])
                ]
            data[i] = data_item

        self.data = data

    def region_query(self, point_identifier):
        return [t[0] for t in self.data[point_identifier]]

    def distance(self, from_point_id, k):
        return self.data[from_point_id][k - 1][1]

    def get_class_id(self, point_id):
        return self.result[point_id]

    def change_cluster_ids(self, point_identifiers, cluster_id):
        for point_identifier in point_identifiers:
            self.result[point_identifier] = cluster_id
```


check-plagiarism.com
ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ОЛЕСЯ ГОНЧАРА Факультет прикладної математики Кафедра обчислювальної математики та математичної кібернетики Курсова робота на тему «Порівняльний аналіз алгоритмів кластеризації» Перший (бакалаврський) рівень вищої освіти Спеціальність 124 Системний аналіз Освітня програма "Системний аналіз" Виконавець студент групи ПС–18–1 Каманцев Артем Сергійович Керівник асистент кафедри ОМ та МК _____ О.С. МагасКількість балів_____ Оцінка за національною шкалою_____ Члени комісії: _____ В.А. Турчина _____ Л.Л. Гарт _____ А. Є. Шевельова 2021 РЕФЕРАТ Курсова робота: 24с., 4 рис., 3 табл., 20 джерел, 1 додаток. Об'єкт дослідження: алгоритми кластеризації DBSCAN, CMDD та SpectACl. Мета роботи: визначити ефективність застосування алгоритмів DBSCAN, CMDD та SpectACl в залежності від даних та цілей кластеризації. Одержані висновки та їх новизна: одержані правила вибору найбільш придатного алгоритму серед DBSCAN, CMDD та SpectACl в залежності від даних та цілей кластеризації. Результати дослідження можуть бути застосовані при визначенні доцільності кластеризації наявних даних за допомогою алгоритмів DBSCAN, CMDD та SpectACl. Перелік ключових слів: КЛАСТЕРИЗАЦІЯ, КЛАСТЕРИЗАЦІЯ ОСНОВАНА НА ЩІЛЬНОСТІ, НАВЧАННЯ БЕЗ ВЧИТЕЛЯ, DBSCAN, CMDD, SpectACl, ПОРІВНЯЛЬНИЙ АНАЛІЗ. ЗМІС
4575 Words
Select File
– Show Advance Options
Feedback
Check Plagiarism
93%
Unique Content
7%
Plagiarized content
✔ COMPLETED
100%
Sentence wise results
Matched URLs
Generate Plagiarism Report
unique
Шевельова 2021 РЕФЕРАТ Курсова робота: 24с., 4 рис., 3 табл.
unique
, 20 джерел, 1 додаток.