Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ТА ДОСЛІДЖЕННЯ МЕТОДУ ДОСТОВІРНОЇ КЛАСТЕРИЗАЦІЇ ДАНИХ НА ОСНОВІ МОДИФІКОВАНОГО АЛГОРИТМА ГУСТАФСОНА-КЕССЕЛЯ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-18-2

Россіна Т.С.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Шафроненко А.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Россіній Тамарі Сергіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та дослідження методу достовірної кластеризації даних на основі модифікованого алгоритма Густафсона-Кесселя

затверджена наказом університету від 16 травня 2022 року № 541Ст

2. Термін подання студентом роботи до екзаменаційної комісії 28 травня 2022 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, середовище розробки програмних систем Microsoft Visual Studio, програмне забезпечення Microsoft Excel.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд класичних та нечітких методів кластеризації.
2. Аналіз актуальності методів кластеризації.
3. Аналіз алгоритму Густафсона-Кесселя.
4. Огляд мови розробки програмного забезпечення.
5. Дослідження ефективності застосування різних методів кластеризації.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Дендрограма, що зображує кластери, 3D модель алгоритму Густафсона-Кесселя, лістинг коду програмної реалізації.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Бєлова Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	18.04.2022	
2	Аналіз завдання, підбір літератури	18.04.22-21.04.22	
3	Аналіз літератури з досліджуваної проблеми	22.04.22-25.04.22	
4	Аналіз технічних засобів	26.04.22-30.04.22	
5	Розробка методу	01.05.22-14.05.22	
6	Програмна реалізація	15.05.22-23.05.22	
7	Оформлення пояснювальної записки	24.05.22-26.05.22	
8	Перевірка на плагіат	27.05.22	
9	Рецензування	28.05.22	
10	Підготовка презентації та доповіді	29.05.22-02.06.22	
11	Занесення роботи в електронний архів	30.05.22	
12	Попередній захист кваліфікаційної роботи	06.06.22	

Дата видачі завдання 18 квітня 2022 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Шафроненко А.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 41 с., 4 табл., 14 рис., 34 джерела.

КЛАСИЧНІ МЕТОДИ КЛАСТЕРИЗАЦІЇ, НЕЧІТКІ НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, ЩО САМООРГАНІЗУЄТЬСЯ, ДОСТОВІРНА НЕЧІТКА КЛАСТЕРИЗАЦІЯ, МОДИФІКАЦІЯ АЛГОРИТМА ГУСТАФСОНА-КЕССЕЛЯ.

Об’єктом роботи є дослідження методу достовірної кластеризації даних.

Метою роботи є розробка модифікованої процедури нечіткої кластеризації, що заснована на алгоритмі Густафсона-Кесселя, яка дозволяє формувати кластери з вільної фіксації осей.

Проведено дослідження методів кластеризації з деякими порівняннями, включаючи в основному класичні методи кластеризації, такі як алгоритми k-середніх, ієрархічні методи кластеризації, нечіткі методи, що формують класи які відрізняються від сферичної.

У результаті роботи здійснена програмна реалізація методу достовірної кластеризації даних на основі модифікованого алгоритму Густафсона-Кесселя.

CLASSICAL CLUSTERING METHODS, FUZZY NEURAL NETWORKS, MACHINE LEARNING, NEURAL NETWORK, SELF-ORGANIZING, PLAUSIBLE FUZZY CLUSTERING, MODIFICATION OF GUSTAFSON-KESSEL ALGORITHM.

The object of the work is to study the method of reliable data clustering.

The aim of the work is to develop a modified fuzzy clustering procedure based on the Gustafson-Kessel algorithm, which allows to form clusters with free fixation of axes.

A study of clustering methods with some comparisons, including mainly classical clustering methods, such as k-means algorithms, hierarchical clustering methods, fuzzy methods that form classes that differ from spherical.

As a result of work the software implementation of a method of reliable clustering of data on the basis of the modified algorithm is carried out Gustafson-Kessel.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

EM – Expectation-maximization Algorithm (алгоритм очікування – максимізації)

PAM – Partitioning Around Medoids

FCM – Fuzzy *C*-means (нечіткі *C*-середні)

GMM – Gaussian Mixture Modals (Гаусові модальні суміші)

FMLE – Fuzzy Maximum Likelihood Estimation (нечітка оцінка максимальної правдоподібності)

SFCM – Suppressed Fuzzy *C*-means (пригнічені нечіткі *C*-середні)

ВСТУП

На даний час все більше і більше даних створюється в основному з сайтів, соціальних мереж, цифрових камер тощо. Тому для впорядкування та класифікації всіх цих величезних обсягів даних використовуються методи класифікації та кластеризації, які об’єднують дані в гомогенні групи, які називаються кластерами. Кластеризація даних є важливим кроком у багатьох областях, включаючи видобуток даних, розпізнавання образів, медицина, комп’ютерний зір, вебстатистика.

Алгоритм нечіткої кластеризації – це алгоритм кластеризації, заснований на розбитті. Його ідея полягає в тому, щоб максимізувати подібність між об’єктами, розділеними на той самий кластер, і мінімізувати подібність між різними кластерами. Нечітка кластеризація виконує нечітке розподілення даних і використовує ступінь приналежності, щоб вказати ступінь належності вибірки до певного класу. Багато алгоритмів нечіткої кластеризації утворюють кластери гіперсферичної форми, відповідно є ймовірність, що при обробці інформації кількість сформованих кластерів буде відрізнятися від фактичної кількості. Саме алгоритм Густафсона-Кесселя формує не сферичні класи, а кластери, що мають гіпереліпсоїдальні форми з довільною орієнтацією осей у просторі ознак.

1 ОГЛЯД МЕТОДІВ КЛАСТЕРИЗАЦІЇ

1.1 Класичні методи кластеризації

Для класичних методів кластеризації існує ефект «метелика», який характеризується тим, що кілька елементів вихідного набору даних відносяться до більшої кількості кластерів. Тут використання нечітких алгоритмів дозволяє побудувати ефективніші методи, ніж чіткі.

1.1.1 Алгоритм *k*-means та варіації

Алгоритм k-means винайдений в 1950-х роках і є на сьогоднішній день найбільш використовуваним алгоритмом кластеризації через його простоту впровадження та ефективність. Алгоритм *k*-means використовує евклідову відстань (формула традиційної відстані між двома точками), і його мета – мінімізувати відстань всередині того ж кластера і максимізувати відстань між кластерами шляхом мінімізації цільової функції *J* [1].

$$J = \sum_{i=1}^{n} \sum_{k=1}^{n} z_i^k \ \|x_{ik} - v_i\|^2, \tag{1.1}$$

де n – кількість точок даних;

x_i – точки даних;

z – є членською функцією;

v_i – кластерні центри.

$$v_i = \frac{\sum_{k=1}^{n} x_{ik} x_{kj}}{\sum_{k=1}^{n} x_{ik}}, \tag{1.2}$$

$$Z_i^k = \begin{cases} 1, \ if \ x_i \ \in cluster \ k, \\ 0, \ if \ not. \end{cases} \quad (1.3)$$

Кроки алгоритму *k*-means:

Крок 1. Вибрати вручну параметр k (кількість кластерів).

Крок 2. Центри кластерів v_i вибираються випадковим чином.

Крок 3. Точки даних x_{ij} призначаються до найближчого кластера.

Крок 4. Повторно обчислити центри кластерів v_i за допомогою рівняння.

Крок 5. Повторюються кроки Крок 3 та Крок 4 доки J не стане інваріантним (дисперсія $< \varepsilon$).

Найважливішими проблемами для кластеризації *k*-means є те, що результат кластеризації сильно залежить від ініціалізації центри кластерів та їх кількості. Алгоритм дійсний лише для числових даних. Крім того, *k*-means не знаходяться до глобального мінімуму, але знаходяться до локального мінімуму. Щоб вирішити ці проблеми, можливо зробити багато *k*-means ітерацій, а потім вибрати найменш об'єктивні значення функції. Щоб усунути деякі обмеження в алгоритмі *k*-means, було запропоновано безліч покращень (варіантів), таких як алгоритм *k*-means++ та *k*-medians [2].

K-means++ – це розумний алгоритм ініціалізації центроїда, а решта алгоритму такий самий, як і у *k*-means. Для ініціалізації центроїда необхідно виконати наступні кроки:

Крок 1. Вибрати першу точку центру випадковим чином.

Крок 2. Обчислити відстань усіх точок у наборі даних від вибраного центроїда.

Крок 3. Зробити точку як новий центроїд, який має максимальну ймовірність.

Крок 4. Повторюються кроки Крок 2 та Крок 3 доки не знайдеться *k*-центроїди.

Проблема з кластеризацією *k*-means і *k*-means++ полягає в тому, що кінцеві центроїди не піддаються інтерпретації, центроїди є не фактичною точкою, а середнім значенням точок, присутніх у цьому кластері.

Ідея кластеризації *k*-medoids полягає в тому, щоб зробити кінцеві центроїди фактичними точками даних. Цей результат робить центроїди інтерпретованими.

Використовуючи евклідову відстань, алгоритм *k*-means знаходить лише сферичні кластери. Також можна використовувати відстань Махаланобіса для пошуку еліпсоїдного кластеру, але з більш високою складністю. *K*-medoids може працювати з будь-якою відстанню та з категоріальними даними, з відхиленнями, але має набагато більшу складність і потребує більше ітерацій. Алгоритм кластеризації *k*-medoids називається Partitioning Around Medoids (PAM) [3]. Основним недоліком алгоритму *k*-medoids є те, що він не підходить для кластеризації несферичних (довільної форми) груп об’єктів. Це пояснюється тим, що він покладається на мінімізацію відстаней між немедіанними об’єктами та центром кластера, бо він використовує компактність як критерій кластеризації замість зв’язності. Отже, можна отримати різні результати для різних запусків на одному наборі даних, оскільки перші *k*-medoids вибираються випадковим чином [4].

1.1.2 Expectation-maximization алгоритм

Алгоритм EM використовується для отримання оцінки максимальної правдоподібності параметрів, коли частина даних відсутня. Однак у загальному випадку алгоритм EM може застосовуватися і за наявності латентних, тобто даних, які ніколи не передбачалося спостерігати в першу чергу. Тут припускається, що латентні дані відсутні. Латентна змінна — це випадкова величина, яку неможливо спостерігати ні на тренуванні, ні етапі тестування [5]. Ці змінні не можна виміряти в кількісній шкалі. Відсутні

значення набору даних можуть бути заповнені за допомогою всіх прийомів і способів, але все одно вони будуть викликати невизначеність, яка перешкоджатиме побудові будь-якої ймовірнісної моделі.

Прихована змінна є прямою причиною для всіх параметрів. Тепер з кінцевою моделлю працювати набагато простіше і вона має таку ж ефективність без зниження гнучкості моделі. У латентних змінних є один недолік: навчити ці моделі важче. Алгоритм ЕМ має багато застосувань у всій статистиці, він часто використовується, наприклад, у машинному навчанні та пошуку даних, а також у Байєсівській статистиці [6]. Алгоритм заснований на знаходженні параметрів максимального параметра максимальної правдоподібності ймовірнісних модальностей. Кластеризація моделі забезпечується кінцевою сумішшю розподілів *f*.

$$f(x/\theta) = \sum_{i=1}^{K} p_i f_i(x/\alpha_i), \tag{1.4}$$

де p_i – це частка *I* класу ($p_i \geq 0\ i\ \sum_i p_i = 1$).

$$\alpha_i = (\sum_i \mu_i), \tag{1.5}$$

де μ_i – центр;

$\sum i$ – дисперсія матриці.

Тоді логарифмічна імовірність глобального параметра максимізується для оцінки попередніх параметрів.

$$\ln f(X/\theta) = \sum_{j=1}^{n} \ln f\left(x_j/\theta\right), \tag{1.6}$$

де $X = (x_1, \dots, x_n)$;

θ – дисперсія параметра;

ε – фіксований поріг.

Нормований добуток ймовірностей, таких як Gaussian Mixture Models (GMM), використовується для вираження розподілу. Модель Гаусової суміші – це тип алгоритму кластеризації, який передбачає, що точка даних генерується із суміші гаусових розподілів з невідомими параметрами. Метою алгоритму є оцінка параметрів гаусових розподілів, а також пропорції точок даних, які надходять з кожного розподілу. На відміну від цього, *k*-means є алгоритмом кластеризації, який не робить жодних припущень щодо основного розподілу точок даних. Замість цього він просто розбиває точки даних на *k* кластерів, де кожен кластер визначається своїм центроїдом. Хоча моделі суміші Гауса є більш гнучкими, їх може бути складніше навчити, ніж *k*-means. *K*-means, як правило, швидше зближуються, і тому можуть бути кращими у випадках, коли час виконання є важливим фактором, загалом, будуть швидшими та точнішими, коли набір даних великий, а кластери добре розділені. Гаусові моделі суміші будуть точнішими, коли набір даних невеликий або кластери не добре розділені [7]. Також, моделі Гаусової суміші є більш гнучкими з точки зору форми кластерів, тоді як *k*-means обмежується сферичними кластерами.

Моделі Гаусової суміші можна використовувати в різних сценаріях, у тому числі коли дані генеруються комбінацією гаусових розподілів, коли існує невизначеність щодо правильної кількості кластерів і коли кластери мають різну форму. У кожному з цих випадків використання моделі суміші Гауса може допомогти підвищити точність результатів. Наприклад, коли дані генеруються за допомогою комбінації гаусових розподілів, використання моделі суміші Гауса може допомогти краще визначити основні закономірності в даних. Крім того, коли існує невизначеність щодо правильної кількості кластерів, використання моделі суміші Гауса може допомогти зменшити частоту помилок.

Алгоритм EM ітеративно використовує очікування та максимізацію, доки не зійдеться. Крок очікування використовується для оцінки ймовірності, потім крок максимізації ймовірності з використанням оцінених параметрів

іншого кроку. Потім результат максимізації використовується іншим кроком очікування тощо. Алгоритм сходиться, коли дисперсія параметра θ нижче фіксованого порогу ε. Алгоритм EM підходить в основному для даних з різним розміром і корельованими даними, але чутливий до шуму. Інші варіанти алгоритму EM використовують адаптивну відстань з використанням гістограми зображення та просторової інформації для обробки зашумлених даних [8].

1.1.3 Ієрархічні методи кластеризації

Ієрархічна кластеризація – це впорядкування даних, у яких будується ієрархія вкладених кластерів. Методологія слідує за стратегією рекурсивного поділу, яка може здійснюватися або зверху вниз, або знизу вгору. Виділяють два класи методів ієрархічної кластеризації агломеративний та дивізивний. Концепція проілюстрована на рисунку 1.1. Також є сім позначених шаблонів що описано на дендограмі, показаний на рисунку 1.2.

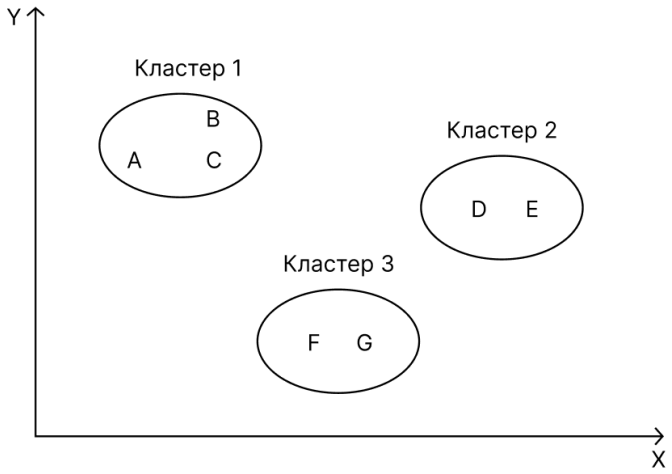

Рисунок 1.1 – Представлення точок для трьох кластерів

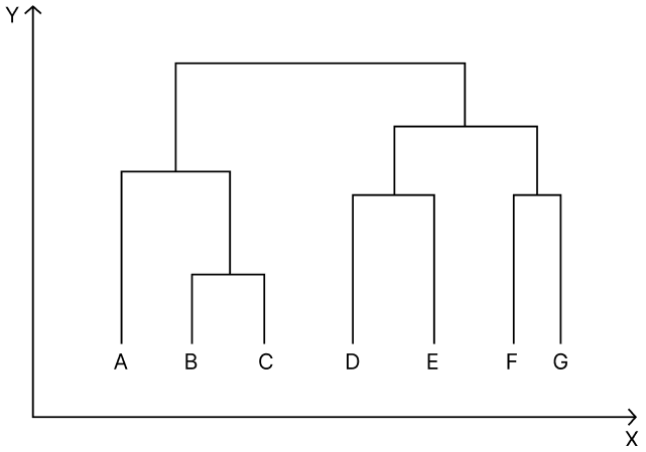

Рисунок 1.2 – Дендрограма, що зображує кластери

Під дендрограмою зазвичай розуміється дерево, побудоване за матрицею мір близькості. Дендрограма дозволяє зобразити взаємні зв’язки між об’єктами із заданої множини. Для створення дендрограми потрібна матриця подібності (або відмінності), яка визначає рівень подібності пари кластерів [9]. Найчастіше використовуються агломеративні методи. Для побудови матриці подібності (відмінності) необхідно задати міру відстані між двома кластерами. Існує безліч методів вимірювання відстані між групами: одиночна ланка (мінімальна відстань), середня ланка (середня відстань), повна ланка (максимальна відстань), метод Уорда (цільова функція) та центроїдний метод.

Агломеративна ієрархічна кластеризація – це методологія агломерації знизу нагору, де кластери мають підкластери і так далі. Вона починається з того, що кожен об’єкт утворює свій власний кластер і ітеративно поєднує кластери у всі великі та великі кластери, поки всі кластери у свою чергу не об’єднаються, і таким чином виходить бажана кластерна структура. Єдиний кластер стає корінням ієрархії. На етапі злиття знаходяться два кластери, які знаходяться найближче один до одного, і об’єднуються в один кластер [10].

Дивізивна ієрархічна кластеризація – це методологія кластеризації зверху вниз. Ця методологія починається з одного кластера, що містить усі об’єкти, а потім, у свою чергу, розбиває наступні кластери доти, доки не

залишаться тільки кластери окремих об’єктів, а також не буде отриманий бажаний кластер. Ця методологія майже аналогічна до агломеративного підходу, за винятком того, що вона починається з єдиного кластера, що складається з усіх об’єктів. Проте ця методологія використовується рідше.

Дивізивна кластеризація є складнішою в порівнянні з агломеративною, оскільки у випадку роздільної кластеризації потрібен метод плоскої кластеризації як «підпрограма», щоб розділити кожен кластер, доки не буде кожен власний кластер одиночного кластера. Дивізивна кластеризація ефективніша, якщо не створюється повна ієрархія аж до окремих листів даних. Алгоритм дивізивний також більш точний. Агломеративна кластеризація приймає рішення, враховуючи локальні шаблони або сусідні точки, спочатку не беручи до уваги глобальний розподіл даних. Ці ранні рішення неможливо скасувати. тоді як розподільна кластеризація враховує глобальний розподіл даних при прийнятті рішень щодо розділення верхнього рівня.

Ієрархічна кластеризація – це дуже корисний алгоритм кластеризації без нагляду. Однак є певні труднощі. Використовуючи цей алгоритм, необхідно враховувати і визначити міру несхожості, тип зв’язку та точку відсікання дендрограми. Хоча єдиної правильної відповіді на ці фактори немає, кожне рішення відкриватиме цікаві аспекти даних, які можна проаналізувати та врахувати.

1.2 Нечіткі методи кластеризації

Математична теорія нечітких множин та нечітка логіка є узагальненнями класичної теорії множин та класичної формальної логіки. Дані поняття були вперше запропоновані американським ученим Лотфі Заде у 1965 р. Основною причиною появи нової теорії стала наявність нечітких та наближених міркувань при описі людиною процесів, систем, об’єктів. Нечітка логіка

намагається діяти як люди та використовувати прості логічні правила для вирішення реальних, складних та нелінійних проблем.

Методи нечіткої кластеризації засновані на нечіткому приналежності, тоді як у класичних методах жорсткої кластеризації дані розподіляються за різними кластерами, отже кожен елемент даних належить до одного кластеру [11]. У методах нечіткої кластеризації (також званої м'якої кластеризації), елементи даних можуть бути членами більш ніж одного кластера, так що об'єкти можуть належати до багатьох кластерів одночасно. Існує три категорії методів нечіткої кластеризації: засновані на нечіткому відношенні, засновані на правилі *k*-найближчого сусіда та на основі об'єктивної функції. Остання категорія найбільш використовується в нечіткій кластеризації, нечіткі методи використовуються для покращення результатів кластеризації результатів, коли межі перекриваються.

1.2.1 Алгоритм нечітких *c*-means

Алгоритм нечітких *c*-means заснований на класичному алгоритмі c-means, розробленому Данном в 1973 році, а потім удосконалений Бездеком в 1981 році. Він досі дуже широко використовується для кластеризації даних. Функція членства – це не просто 0 або 1, а значення між 0 та 1, тому вектор членства (для *k*-середніх) замінюється матрицею членства (для *c*-середніх). Членство представлено матрицею *c* на *n*, де *c* – кількість нечітких підмножин, а *n* – кількість об'єктів. Кожен рядок є приналежністю всіх n об'єктів до певного нечіткого підмножини, а кожен стовпець представляє приналежність об'єкта до всіх c нечітких підмножин. Алгоритм *c*-means дуже подібний до алгоритму *k*-means, але з новою об'єктивною функцією *J'*.

$$J'(U,V) = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ij}^{m} \ \|x_j - v_i\|^2, \qquad (1.7)$$

де u – нечітка матриця розділу;

v – набір прототипів;

n – кількість прототипів;

c – кількість кластерів;

x_j – точка вимірюваних даних;

v_i – кластерні центри.

$$v_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m}, 1 \leq i \leq c, \quad (1.8)$$

де u_{ij} – це значення членства x_j з відношенням до кластера i:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} (\|x_j - v_i\| / \|x_j - v_k\|)^{2/(m-1)}}, m \geq 1, \quad (1.9)$$

де m – визначає ступінь нечіткості кластера.

Параметр фазіфікатора m фіксується відповідно до програми. Він використовується для зниження чутливості класових центрів до шуму. Більше значення m призводить до більш нечітких кластерів, а мале значення $m \cong 1$ призводить до чітких кластерів [12]. Бездек довів, що фазифікатор m має бути встановлений у [1,5; 2,5]. Загалом, m встановлено на 2. Етапи алгоритмів подібні до алгоритму чіткого *k*-means:

Крок 1. Вибрати центри с випадковим чином (вектор *V*).

Крок 2. Обчислення матриці розподілу u.

Крок 3. Оновлення центрів v_i.

Крок 4. Обчислення цільової функції *J*.

Крок 5. Повторюються кроки Крок 3 та Крок 4 до зближення.

Найважливішими проблемами для алгоритму є обчислювальний час, він набагато повільніше, ніж чіткі c-means через ітераційний характер [13], вибір метричної відстані (евклідова відстань не завжди найкраща), вибір початкових

кластерів, вибір кількості кластери, вибір параметра фазіфікатора m та чутливість шуму.

1.2.2 Варіанти нечітких *c*-means

Багато авторів запропонували варіанти нечітких *c*-means для вирішення деяких з цих проблем, наприклад, використання інших функцій віддаленості, використання інших функцій членства або використання ієрархічного агломеративного алгоритму або хонен-мереж для визначення кількості кластерів [14-16].

Густафсон і Кессель запропонували використовувати адаптивну норму відстані, яка індукує матрицю *n* на *n*, щоб оптимізувати функцію відстані. В цьому алгоритмі замість евклідової відстані використовується відстань Махаланобіса. Оскільки відстань Махалонобіса формує кластери у формі еліпса, у той час як евклідова відстань використовується для виявлення кластероподібних кластерів.

Гат і Гева запропонували FMLE (Fuzzy Maximum Likelihood Estimation) з відстанню норми відстані. Цей метод дає кращі результати, особливо для гіпереліпсоїдних класів [17].

Ахмед запропонував у 2002 році модифікований алгоритм нечітких *c*-means на основі модифікованої цільової функції на основі інформації сусіда. Цей алгоритм є більш надійним, ніж кластеризація с-середніх, і дає хорошу класифікацію, в основному для шумних зображень.

Інші автори запропонували SFCM (Supressed FCM). Алгоритм полягає у збільшенні та приглушенні найбільших ступенів приналежності. Цей алгоритм сходиться швидше, ніж FCM, але його продуктивність сильно залежить від випадкового параметра α. Модифікований SFCM визначає параметр α за допомогою підходу до навчання, керованого прототипом.

Ніхіл у 2005 році запропонував імовірний FCM, який є нечіткою версією можливостних *c*-means. PFCM має як переваги нечіткого членства, так і типовість PCM. PFCM використовує просторову інформацію в цільовій функції, щоб отримати кращі результати кластеризації. На відміну від алгоритму FCM, який змушує викиди належати до одного або кількох кластерів, що спотворює результат кластеризації, PFCM є більш надійним для викидів. Цільова функція PFCM має два випадково фіксованих параметри *a* (ступінь нечіткості) і *b* (ступінь типовості).

У 2011 році Саад запропонував Enhanced PFCM, у якому параметри *a* і *b* не мають випадкових значень, а обчислюються відповідно до точок даних, щоб мати кращі результати кластеризації. У 2014 році Чжао запропонував інший модифікований SFCM.

1.2.3 Нечіткі множини другого типу

Заде запропонував нечітку логіку другого типу в 1975 року. Це узагальнення нечіткої множини типу 1, яке полягає у використанні невизначеної (або нечіткої) функції належності. На відміну від нечіткої логіки, яка використовує двовимірне членство, нечітка логіка другого типу є тривимірним членством, тому вона може обробляти більше невизначеностей і краще виявляти викиди. Для нечіткої логіки другого типу функція власності називається FOU [18].

FOU являє собою розмиття функції належності типу 1 і повністю описується двома його функціями, що обмежують, нижчою функцією належності (LMF) і функцією верхньої належності (UMF), обидві з яких є типом 1 нечіткий набір. Інтелектуальні системи, що базуються на нечіткій логіці, є фундаментальними інструментами моделювання нелінійних складних систем. Нечіткі множини та нечітка логіка є основою для нечітких систем, метою яких було моделювання того, як мозок маніпулює неточною

інформацією. Нечіткі множини типу 2 краще використовуються для моделювання невизначеності та неточності [19]. Нові поняття були введені Менделем і Ліангом що дозволяє охарактеризувати нечітке безліч типу 2 з найвищою функцією власності та нижчою функцією власності; кожна з цих двох функцій може бути представлена функцією належності нечіткої множини типу 1. Інтервал між цими двома функціями є слідом невизначеності (FOU), який використовується для характеристики нечіткої множини типу 2. Невизначеність є недосконаліть знання про природний процес або природний стан. Статистична невизначеність - це випадковість чи помилка, що виникає з різних джерел, коли використовуємо її у статистичній методології [20]. Існують різні джерела невизначеності у процесі оцінки та обчислення.

П’ять типів невизначеності, що виникають із природного стану неточного знання:

– похибка вимірювання, помилка величин, що спостерігаються;

– невизначеність процесу, динамічна випадковість;

– невизначеність моделі, неправильна специфікація структури моделі;

– оцінка невизначеності, яка може виникнути з будь-якої з попередніх невизначеностей або їх комбінації, і називається вона неточністю;

– невизначеність реалізації, наслідок мінливості, що у результаті політики сортування, нездатність досягти конкретної стратегічної мети.

1.3 Постановка задачі

Об’єктом роботи є дослідження методу достовірної кластеризації даних.

Метою роботи є розробка модифікованої процедури нечіткої кластеризації, що заснована на алгоритмі Густафсона-Кесселя, яка дозволяє формувати кластери із вільної фіксації осей.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз класичних методів кластеризації;

– провести аналіз нечітких методів кластеризації, що формують класи які відрізняються від сферичної;

– розглянути процедуру типу Густафсона-Кесселя, що має робастні властивості;

– реалізувати комп’ютерну модель для перевірки дії алгоритму;

– зробити порівняльний аналіз результатів роботи модифікованого алгоритму Густафсона-Кесселя у порівнянні з іншими алгоритмами кластеризації.

2 МОДИФІКОВАНИЙ АЛГОРИТМ ГУСТАФСОНА-КЕССЕЛЯ

2.1 Кластери гіперелліпсоїдної форми

Більшість популярних алгоритмів нечіткої кластеризації призначені для обробки інформації в пакетному режимі, коли весь масив даних, що підлягають обробці, задається заздалегідь. Зрозуміло, що цей підхід не є ефективним для завдань Big Data Mining і Data Stream Mining, коли дані подаються на обробку послідовно, можливо, в онлайн-режимі. У таких завданнях на перший план виступають рекурентні процедури нечіткої кластеризації, в яких кожне оброблене спостереження надалі не використовується, тобто фактично забувається. Модифікованої процедури нечіткої кластеризації з урахуванням алгоритму Густафсона-Кесселя, тобто фактично з відстанню Махаланобіса, що дозволяє формувати кластери гіперелліпсоїдної форми з довільною орієнтацією осей [21].

Відстань Махаланобіса – відстань у евклідовому просторі, що узагальнює поняття евклідової відстані. Визначається формулою:

$$d(X,Y,Z) = \sqrt{(X-Y)^T S^{-1}(X-Y)}\,, \tag{2.1}$$

де X – вектор;

Y – вектор;

T – позначає операцію транспонування;

S – матриця.

Відстань Махаланобіса використовується в багатовимірному статистичному аналізі, зокрема при перевірці гіпотез, класифікації спостережень і в кластерному аналізі. У цих застосуваннях S є коваріаційною матрицею деякого багатовимірного розподілу, що дозволяє визначити відстань між випадковими векторами із цього розподілу із врахуваннями

кореляцій між компонентами [22]. У випадку коли S – одинична матриця, відстань Махаланобіса збігається з евклідовою відстанню.

Алгоритм обчислення відстані між двома точками та між точкою та класом:

Крок 1. Обчислити математичні очікування значень ознак точок класу.

Крок 2. Обчислити середньоквадратичні відхилення значень ознак точок класу.

Крок 3. Обчислити коваріації між усіма парами ознак точок класу та скласти підступну матрицю.

Крок 4. Якщо матриця оборотна, то вирахувати відстань по Махаланобісу.

У процесі нечіткої кластеризації з використанням розглянутих алгоритмів формовані класи мають форму гіперсфер, що не завжди відповідає реальним умовам, коли ці кластери можуть мати довільну форму. Більш адекватною і зручною формою є кластери гіпереліпсоідальної форми з довільною орієнтацією осей у просторі ознак [23]. Такі кластери можна формувати за допомогою алгоритму Густафсона-Кесселя та його модифікацій, де відстань має вигляд:

$$\mathrm{D}^2_{\mathrm{A_q}}\big(\mathrm{x}(\tau),\mathrm{c_q}\big) = \big\|\mathrm{x}(\tau)-\mathrm{c_q}\big\|^2_{\mathrm{A_q}} = \big(\mathrm{x}(\tau)-\mathrm{w_q}\big)^{\mathrm{T}}\mathrm{A_q}(\mathrm{x}(\tau)-\mathrm{c_q}), \qquad (2.2)$$

$$\begin{cases} A_q = (det\, S_q)^{\frac{1}{n}} S_q^{-1}, \\ S_q = \sum_{\tau=1}^{N} \mu_q^{\beta}(\tau)\big(x(\tau)-c_q\big)\big(x(\tau)-c_q\big)^T. \end{cases} \qquad (2.3)$$

Мінімізація призводить до результату [24]:

$$\begin{cases} \mu_q(k) = \dfrac{\left(D^2_{A_q}(x(\tau),c_q)\right)^{\frac{1}{1-\beta}}}{\sum_{l=1}^{m}\left(D^2_{A_l}(x(\tau),c_l)\right)^{\frac{1}{1-\beta}}}, \\ c_q = \dfrac{\sum_{\tau=1}^{N}\mu_q^{\beta}(\tau)x(\tau)}{\sum_{\tau=1}^{N}\mu_q^{\beta}(\tau)}. \end{cases} \quad (2.4)$$

Приймається значення фаззифікатора як $\beta = 2$ [25-28], маємо рішення у вигляді:

$$\begin{cases} \mu_q(\tau) = \dfrac{1}{1+\dfrac{D^2_{A_q}(x(\tau),c_q)}{\sigma_q^2(\tau)}}, \\ \sigma_q^2(\tau) = \left(\sum_{\substack{l=1 \\ l\neq q}}^{m} D_{A_l}^{-2}(x(\tau),c_l)\right)^{-1}, \\ c_q = \dfrac{\sum_{k=1}^{N}\mu_q^2(\tau)x(\tau)}{\sum_{\tau=1}^{N}\mu_q^2(\tau)}. \end{cases} \quad (2.5)$$

Таким чином, співвідношення по суті є процедурою імовірнісної нечіткої кластеризації, але класи, що утворюються, мають вигляд гіпереліпсоїдів з довільною орієнтацією осей.

2.2 Модифікація алгоритму

Щоб ввести рекурентну модифікацію алгоритму Густафсона-Кесселя, можемо використовувати формулу обігу матриці Шермана-Моррісона [29]. та лему про визначника матриці, що призводить до процедури:

$$\begin{cases} \mu_q(\tau+1) = \dfrac{\left(D^2_{A_q(\tau)}\left(x(\tau+1), c_q(\tau)\right)\right)^{\frac{1}{1-\beta}}}{\sum_{l=1}^{m}\left(D^2_{A_q(\tau)}\left(x(\tau+1), c_l(\tau)\right)\right)^{\frac{1}{1-\beta}}}, \\ S_q(\tau+1) = S_q(\tau) + \mu_q^{\beta}(\tau+1)\big(x(\tau+1) - c_q(\tau)\big)\big(x(\tau+1) - c_q(\tau)\big)^T, \\ S_q^{-1}(\tau+1) = S_q^{-1}(\tau) - \dfrac{\mu_q^{\beta}(\tau+1) S_q^{-1}(\tau)\big(x(\tau+1) - c_q(\tau)\big)\big(x(\tau+1) - c_q(\tau)\big)^T S_q^{-1}(\tau)}{1+\mu_q^{\beta}(\tau+1)\big(x(\tau+1) - c_q(\tau)\big)^T S_q^{-1}(\tau)\big(x(\tau+1) - c_q(\tau)\big)}, \\ det\, S_q\,(\tau+1) = \big(det\, S_q\,(\tau)\big)\Big(1 + \mu_q^{\beta}(\tau+1)\big(x(\tau+1) - c_q(\tau)\big)^T\big(x(\tau+1) - c_q(\tau)\big)\Big), \\ A_q(\tau+1) = (det\, S_q\,(\tau+1))^{\frac{1}{n}} S_q^{-1}(\tau+1), \\ c_q(\tau+1) = c(\tau) + r(\tau+1)\mu_q^{\beta}(\tau+1) A_q(\tau+1)\big(x(\tau+1) - c_q(\tau)\big). \end{cases} \tag{2.6}$$

Метод Густафсона-Кесселя легко модифікувати у разі можливої нечіткої кластеризації [30]. У цьому випадку цільова функція набуває вигляду:

$$E\big(\mu_q(\tau), c_q, \eta_q\big) = \sum_{\tau=1}^{N}\sum_{q=1}^{m}\mu_q^{\beta}(\tau) D^2_{A_q}\big(x(\tau), c_q\big) + \\ + \sum_{q=1}^{m}\eta_q\sum_{\tau=1}^{N}\big(1 - \mu_q(\tau)\big)^{\beta}, \tag{2.7}$$

та у пакетній формі:

$$\begin{cases} \mu_q(\tau) = \left(1 + \dfrac{D^2_{A_q}\big(x(\tau), c_q\big)^{\frac{1}{\beta-1}}}{\eta_q}\right)^{-1}, \\ c_q = \dfrac{\sum_{k=1}^{N}\mu_q^{\beta}(\tau) x(\tau)}{\sum_{k=1}^{N}\mu_q^{\beta}(\tau)}, \\ S_q = \sum_{\tau=1}^{N}\mu_q^{\beta}(\tau)\big(x(\tau) - c_q\big)\big(x(\tau) - c_q\big)^T, \\ A_q = (det\, S_q)^{\frac{1}{n}} S_q^{-1}, \\ \eta_q(\tau) = \dfrac{\sum_{\tau=1}^{N}\mu_q^{\beta}(\tau)\big(x(\tau) - c_q\big)^T A_q\big(x(\tau) - c_q\big)}{\sum_{\tau=1}^{N}\mu_q^{\beta}(\tau)}. \end{cases} \tag{2.8}$$

Алгоритм можна записати в рекурентній формі:

$$\begin{cases} \mu_q(\tau) = \left(1 + \left(\frac{D^2_{A_q}(\tau)(x(\tau+1),c_q(\tau))}{\eta_q(\tau)}\right)^{\frac{1}{\beta-1}}\right)^{-1}, \\ S_q(\tau+1) = S_q(\tau) + \mu_q^{\beta}(\tau+1)\big(x(\tau+1) - c_q(\tau)\big)\big(x(\tau+1) - c_q(\tau)\big)^T, \\ S_q^{-1}(\tau+1) = S_q^{-1}(\tau) - \frac{\mu_q^{\beta}(\tau+1)S_q^{-1}(\tau)\big(x(\tau+1)-c_q(\tau)\big)\big(x(\tau+1)-c_q(\tau)\big)^T S_q^{-1}(\tau)}{1+\mu_q^{\beta}(\tau+1)\big(x(\tau+1)-c_q(\tau)\big)^T S_q^{-1}(\tau)\big(x(\tau+1)-c_q(\tau)\big)}, \\ det\, S_q\,(\tau+1) = \big(det\, S_q\,(\tau)\big)\Big(1 + \mu_q^{\beta}(\tau+1)\big(x(\tau+1) - c_q(\tau)\big)^T\big(x(\tau+1) - c_q(\tau)\big)\Big), \\ A_q(\tau+1) = (det\, S_q\,(\tau+1))^{\frac{1}{n}} S_q^{-1}(\tau+1), \\ c_q(\tau+1) = c_q(\tau) + r(\tau+1)\mu_q^{\beta}(\tau+1)A_q(\tau+1)\big(x(\tau+1) - c_q(\tau)\big), \\ \eta_q(\tau+1) = \frac{\sum_{p=1}^{\tau+1}\mu_q^{\beta}(p)\big(x(p)-c_q(\tau+1)\big)^T A_q(x+1)\big(x(p)-c_q(\tau+1)\big)}{\sum_{p=1}^{\tau+1}\mu_q^{\beta}(p)}. \end{cases} \tag{2.9}$$

2.3 Достовірний варіант алгоритму Густафсона-Кесселя

Що стосується достовірного варіанта алгоритму Густафсона-Кесселя, то замість цільової функції слід використовувати його модифікацію у вигляді:

$$E\big(Cred_q(\tau), c_q\big) = \sum_{\tau=1}^{N}\sum_{q=1}^{m} Cred_q^{\beta}(\tau) D^2_{A_q}\big(x(\tau), c_q\big). \tag{2.10}$$

З обмеженнями можно записати так:

$$\begin{cases} S_q = \sum_{k=1}^{N} \mu_q^{\beta}(\tau)\big(x(\tau) - c_q\big)\big(x(\tau) - c_q\big)^T, \\ A_q = (det\, S_q)^{\frac{1}{n}} S_q^{-1}, \\ \mu_q(\tau) = \frac{1}{1+D^2_{A_q}\left(x(\tau),c_q\right)}, \\ \mu_q^*(\tau) = \frac{\mu_q(\tau)}{sup\, \mu_l(\tau)}, \\ Cred_q(\tau) = \frac{1}{2}\big(\mu_q^*(\tau) + 1 - sup\, \mu_l^*(\tau)\big), \\ c_q = \frac{\sum_{\tau=1}^{N} Cred_q^{\beta}(\tau)x(\tau)}{\sum_{\tau=1}^{N} Cred_q^{\beta}(\tau)}. \end{cases} \tag{2.11}$$

Співвідношення є узагальненням алгоритму у разі метрики [31]. Можемо запровадити рекурентну правдоподібну модифікацію алгоритму Густафсона-Кесселя:

$$\begin{cases} \mu_q(\tau+1) = \left(1 + D^2_{A_q}(\tau)\big(x(\tau+1), c_q(\tau)\big)\right)^{-1}, \\ \mu^*_q(\tau+1) = \frac{\mu_q(\tau+1)}{sup\,\mu_l(\tau+1)}, \\ Cred_q(\tau+1) = \frac{1}{2}\big(\mu^*_q(\tau+1) - 1 - sup\,\mu^*_l(\tau+1)\big), \\ S_q(\tau+1) = S_q(\tau) + \mu^\beta_q(\tau+1)\big(x(\tau+1) - c_q(\tau)\big)\big(x(\tau+1) - c_q(\tau)\big)^T, \\ S^{-1}_q(\tau+1) = S^{-1}_q(\tau) - \frac{\mu^\beta_q(\tau+1)S^{-1}_q(\tau)\big(x(\tau+1)-c_q(\tau)\big)\big(x(\tau+1)-c_q(\tau)\big)^T S^{-1}_q(\tau)}{1+\mu^\beta_q(\tau+1)\big(x(\tau+1)-c_q(\tau)\big)^T S^{-1}_q(\tau)\big(x(\tau+1)-c_q(\tau)\big)}, \\ det\,S_q(\tau+1) = \big(det\,S_q(\tau)\big)\left(1 + \mu^\beta_q(\tau+1)\big(x(\tau+1) - c_q(\tau)\big)^T\big(x(\tau+1) - c_q(\tau)\big)\right), \\ A(\tau+1) = (det\,S_q(\tau+1))^{\frac{1}{n}} S^{-1}_q(\tau+1), \\ c_q(\tau+1) = c_q(\tau) + r(\tau+1)\mu^\beta_q(\tau+1)A_q(\tau+1)\big(x(\tau+1) - c_q(\tau)\big). \end{cases} \quad (2.12)$$

3 КОМП'ЮТЕРНА МОДЕЛЬ АЛГОРИТМУ ГУСТАФСОНА-КЕССЕЛЯ

3.1 Обґрунтування вибору мови програмної реалізації

У рамках кваліфікаційної розробка алгоритму Густафсона-Кесселя за допомогою мови *R*. Мова *R* – головний конкурент Python для тих, хто займається статистикою та аналізом даних. Його використовують у соціальних та економічних науках для пошуку причинно-наслідкових зв'язків, порівняння вибірок, створення наочних звітів та графіків. Мова розробили вчені факультету статистики університету Окленда [32]. Спочатку це був внутрішній інструмент, але потім його зробили доступним для всіх – вже вдалим він вийшов. Синтаксис мови *R* простий і включає мінімальний набір примітивних типів даних: символьні, числові, логічні та комплексні. Примітивні типи об'єднуються у складніші. Наприклад, тип вектор – це, насправді, перелік з кількох об'єктів (чисел, рядків та інших). Числові змінні можуть набувати і особливі значення: NaN (not a number – не число), Inf (infinity – нескінченність) та NA (not available – недоступно).

Найпопулярніша команда в *R* – читання файлу, тому що треба постійно відкривати та досліджувати датасети.

Лістинг 3.1 Команда читання файлу:

```
data <- read.csv("input.csv", sep = ',')
```

За допомогою мови *R* можна реалізувати багато завдань. Обробити, очистити та перетворити дані для дослідження. Наприклад, як подивитися, скільки в середньому користувачів завантажили мобільний застосунок за кожен літній та осінній місяць. Мова *R* дозволяє виключити з графіка зиму та осінь та згрупувати їх по місяцях для подальших підрахунків [33]. Провести статистичні випробування. Припустимо, як дізнатися, чи відрізняється

середня тривалість життя чоловіків та жінок. Для цього можна запустити *t*-тест – його результати покажуть, чи є статистично значущі різницю між даними. Виконати розвідувальний аналіз. Дані необхідно перевіряти на нормальність, тому що багато статистичних методів (наприклад, той самий *t*-тест) вимагають нормального розподілу у вихідниках. Нормальний розподіл передбачає, що більша частина даних групується близько середнього значення, а інших значень набагато менше. Такий розподіл часто зустрічається в житті: людей середнього зростання у світі найбільше, а високих і низьких мало. *R* є інструменти перевірки нормальності за допомогою графіків і тестів. Також намалювати інтерактивний графік та відрегулювати його параметри – значення по осях тощо [34].

3.2 Програмна реалізація

Алгоритм Густафсона-Кесселя – це один з алгоритмів нечіткої кластеризації, який має справу з мірою належності точки до кластера замість бінарної класифікації 0/1 (або належить, або не належить). Таким чином, зрештою, кожна точка не призначається кластеру, а замість цього отримує вектор мір приналежності. Отже, точка зазвичай належить усім кластерам одночасно, але її членство має різну «силу» для різних кластерів. Однак, коли кластеризація буде завершена, можна перетворити нечіткі кластери на crips, просто призначивши точці кластер з найвищим значенням функції належності.

Спочатку представимо традиційну функцію попередньої обробки: нормалізацію та кодування (рис. 3.1). Перший нормалізує дані шляхом віднімання середніх, другий масштабує дані в інтервалі (-1,1).

Процедура створює *k* центроїдів з кількістю вимірів, що дорівнює кількості елементів. Центр ініціалізуються випадковими векторами в інтервалі [-1,1]. Кожен центроїд має «супутника» – матрицю перетворення *A*, яка ініціалізується як ідентична матриця (рис. 3.3).

```
normalize ← function(M) {
        #center data
        means = apply(M,2,mean)
        Xnorm = t(apply(M,1,function(x) {x-means}))
        Xnorm
}

encode ← function(M) {
        # put on hypershpere
        mins = apply(M,2,min)
        maxs = apply(M,2,max)
        ranges = maxs-mins
        Xnorm = t(apply(M,1,function(x) { 2*(x-mins)/ranges-1}))
        Xnorm = t(apply(Xnorm, 1, function(x) { x/norm(x,type="2")}))
        Xnorm
}
```

Рисунок 3.1 – Функція нормалізації та кодування

Тепер нормалізуємо та закодуємо набір даних райдужки (рис. 3.2).

```
iris_norm = normalize(iris_X)
iris_norm = encode(iris_norm)
```

Рисунок 3.2 – Функція нормалізації та кодування

```
initCentroidsAndA ← function(k,n_features) {
        set.seed(123)
        centroids = matrix(runif(k*n_features,-1,1), ncol = n_features)
        centroids = normalize(centroids)
        centroids = encode(centroids)
        A = vector(mode="list", length=k)
        for (i in seq(1,k)) {
                A[[i]] = diag(x=1, ncol = n_features, nrow = n_features)
        }
        list(ce = centroids, A = A)
}
```

Рисунок 3.3 – Ініціалізація центроїдів і матриць перетворення

Наступна функція обчислює відстані Махаланобіса (міра відстані між векторами випадкових величин, що узагальнює поняття евклідова відстані) для кожної точки для кожного центроїда (рис. 3.4).

```
calculate_Mahalanobis ← function(X, cAndA) {
        k = nrow(cAndA$ce)
        centroids = cAndA$ce
        A = cAndA$A
        distances = c()
        for (i in seq(1,k)) {
                dsq = as.matrix(apply(X, 1, function(x)
                { ((t(x-centroids[i,]))%*%solve(A[[i]]))
                %*%(x-centroids[i,]) }))

                colnames(dsq) = paste("centroid",i)
                distances = cbind(distances,dsq)
        }
        distances
}
```

Рисунок 3.4 – Пошук відстані

Наступна функція обчислює членство – якою мірою кожна точка належить кожному кластеру (рис. 3.5).

```
cAndA = initCentroidsAndA(3,n_features)
dst = calculate_Mahalanobis(iris_norm, cAndA)
calculate_memberships ← function(distances) {
        mus = t(apply(distances, 1, function(x)
        { (1/x)/(sum(1/x)) }))
        mus
}
mships = calculate_memberships(dst)
```

Рисунок 3.5 – Обчислення міри

Наступним важливим кроком є оновлення центроїдів і матриць перетворення, щоб була можливість перейти до наступної ітерації алгоритму (рис. 3.6, рис. 3.7).

```
update_centroids ← function(X,memberships) {
        k = ncol(memberships)
        n_features = ncol(X)
        N = nrow(X)
        full = cbind(X,memberships)
        centroids = c()
        A = vector(mode="list", length=k)
        for (i in seq(1,k)) {
                den = sum((memberships[,i])^2)
                num = 0
                Fnum = matrix(data=0, nrow = n_features, ncol = n_features)
                for (j in seq(1,N)) {
                        num = num + X[j,]*((memberships[j,i])^2)
                }
                cent = num/den
                centroids = rbind(centroids,cent)
                for (j in seq(1,N)) {
                        Fnum = Fnum + (outer( ((memberships[j,i])^2)
                        *(X[j,]-centroids[i,]) , X[j,]-centroids[i,]))
                }
                Fi = Fnum/den
                A[[i]] = ((det(Fi))^(1/n_features))*solve(Fi)
        }
        list(ce = centroids, A = A)
}
```

Рисунок 3.6 – Оновлення центроїдів і матриць

```
            [,1]       [,2]        [,3]       [,4]
cent -0.3759869  0.1449268 -0.50403834 -0.5290240
cent  0.3258143 -0.1529541  0.54963406  0.5681777
cent -0.3645772 -0.5906380 -0.05397593 -0.1659737
[1] "Current centroids"
            [,1]       [,2]        [,3]        [,4]
cent -0.4383195  0.1159104 -0.59350289 -0.61976473
cent  0.2963708 -0.2222011  0.54389983  0.55521261
cent -0.4407734 -0.7467326  0.06861647 -0.07497507
[1] "Current centroids"
            [,1]       [,2]        [,3]        [,4]
cent -0.4298221  0.1146965 -0.58237093 -0.60837306
cent  0.3044524 -0.2177096  0.55296351  0.57096645
cent -0.4175192 -0.7177399  0.06108012 -0.08144236
```

Рисунок 3.7 – Поточні центроїди

Після того, як завершилась кластеризацію, треба подивитися на результати та отримати деяку візуалізацію. Для цього потрібно отримати найближчий чіткий розділ даних (рис. 3.8).

```
hard_partition ← function(memberships) {
        apply(mships,1,which.max)
}
```

Рисунок 3.8 – Пошук ближнього чіткого розділу

Спостерігаючи результати зрозуміло, що алгоритм працює нормально, оскільки він все ще знаходить справжні кластери. Набір даних мав 4 атрибути, то можна побудувати лише у 3D, тому пропускаємо атрибут з найменшою дисперсією (рис. 3.9, рис. 3.10).

```
library(rgl)
library(car)
ind = sort(apply(iris_norm,2,var),
        index.return=TRUE,
        decreasing = TRUE)$ix[1:3]
labs = cnames[ind]
labels = hard_partition(z$mships)
sset = iris_norm[,ind]
x = sset[,1]
y = sset[,2]
z = sset[,3]
par3d("windowRect"= c(0,0,400,400))
scatter3d(x = x, y = y, z = z,
          xlab = labs[1],
          ylab = labs[2],
          zlab = labs[3],
          labels = NULL,
          groups = as.factor(labels),
          surface = FALSE,
          grid = FALSE,
          ellipsoid=TRUE)
rgl.snapshot(filename = "gk.png")
```

Рисунок 3.9 – Візуалізація алгоритму

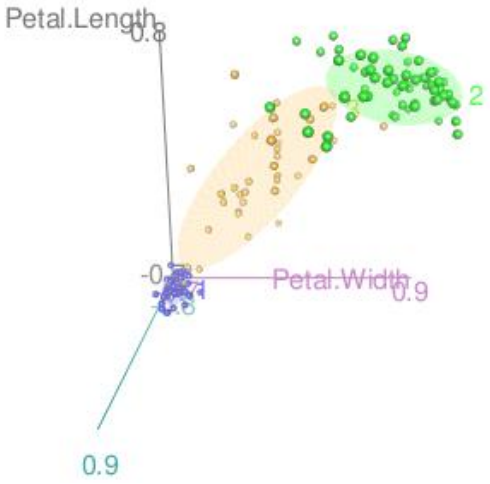

Рисунок 3.10 – 3D-модель алгоритму Густафсона-Кесселя

3.3 Результати досліджень

Для оцінки ефективності з іншими методами кластеризації та доведення переваги було обрано метод метод нечітких *c*-means (FCM) та алгоритм Густафсона-Кесселя (GK), для порівняння із запропонованим рекурентним достовірним модифікованим алгоритмом Густафсона-Кесселя (RCM_GK). Наступні експериментальні дослідження проведені на зразках трьох наборів даних: Іриси та Вино. Опис цих наборів даних показано у таблиці 3.1.

Таблиця 3.1 – Опис набору даних: набір даних, номер даних, номер атрибутів, номер кластера

Набір даних	**Кількість даних**	**Кількість атрибутів**	**Кількість кластерів**
Іриси	296	2	6
Вино	178	13	3

Середню похибку центроїдів кластерів запропонованого RCM_GK порівняли з іншими методами такими як, FCM та GK з RCM_GK отриманий результат даних показано у таблиці 3.2.

Таблиця 3.2 – Порівняння середньої похибки центроїдів кластерів, алгоритму RCM_GK з FCM та GK

Набір даних	**FCM**	**GK**	**RCM_GK**
Іриси	2,69	0,13	0,049
Вино	2,71	0,183	0,037

Щоб оцінити практичність цих методів, порівняємо час роботи кластеризації на різних наборах даних, результат даних показано у таблиці 3.3.

Таблиця 3.3 – Порівняння часу виконання у секундах алгоритму RCM_GK з FCM та GK

Набір даних	**FCM**	**GK**	**RCM_GK**
Іриси	0,41	0,20	0,14
Вино	0,23	0,25	0,16

На рисунку 3.11 представлено порівняння часу роботи цих алгоритмів. Як видно з діаграми, швидкість роботи при розв'язанні задачі рекурентним достовірним модифікованим алгоритмом Густафсона-Кесселя вища за інші алгоритми.

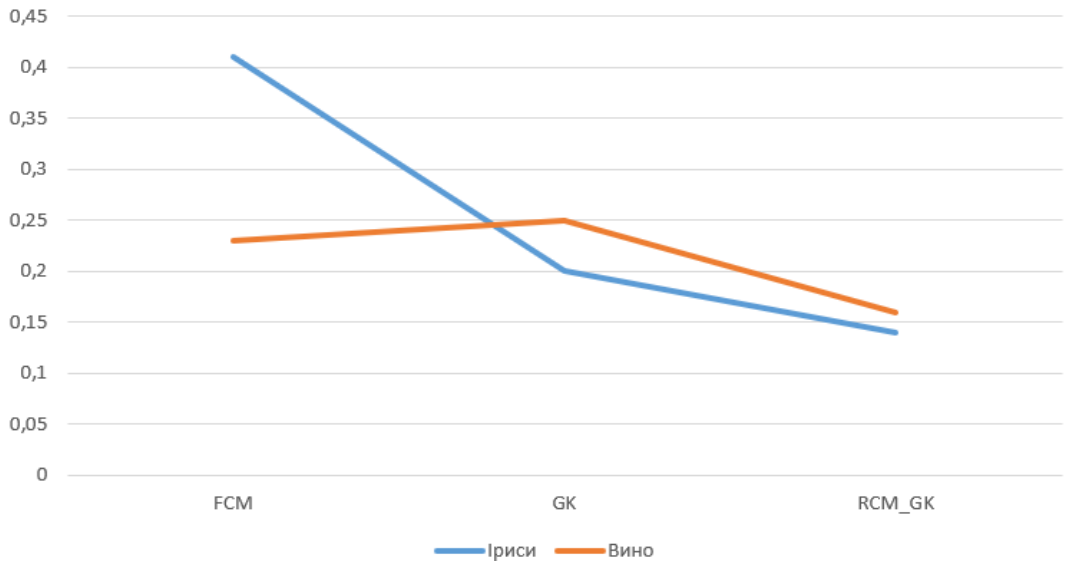

Рисунок 3.11 – Порівняння часу алгоритмів FCM, GK та RCM_GK на перевірених наборах даних

Зробимо порівняння кількості ітерацій та часу виконання, результат даних показано у таблиці 3.4 та на рисунку 3.12.

Таблиця 3.4 – Порівняння кількості ітерацій та часу виконання у секундах алгоритмів FCM, GK та RCM_GK у наборі даних Іриси

	FCM	**GK**	**RCM_GK**
Кількість ітерацій	35	100	78
Час виконання	1,58	1,22	1,17

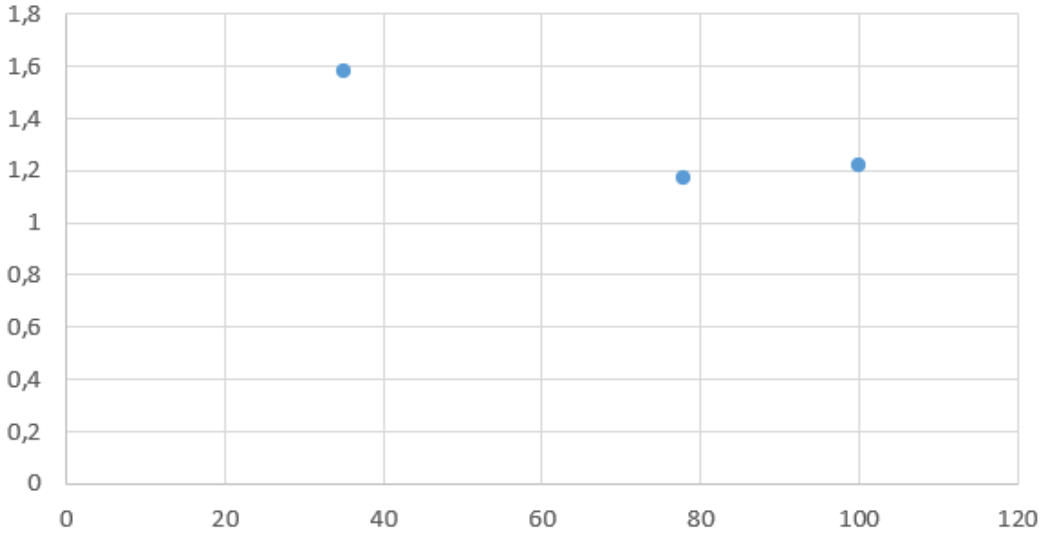

Рисунок 3.12 – Порівняння кількості ітерацій та часу алгоритмів на перевіреному наборі даних Іриси

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований метод достовірної кластеризації даних на основі модифікованого алгоритма Густафсона-Кесселя.

Запропонована модифікація алгоритму Густафсона-Кесселя, заснована на довірчому підході до нечіткої кластеризації, дозволила сформувати класи гіперліпоідальної форми, що перекриваються, з довільною орієнтацією осей у просторі ознак. Розглянуті процедури досить прості у чисельній реалізації та призначені для вирішення задач кластеризації в рамках Data Stream Mining та Big Data Mining. Проведені експерименти підтвердили ефективність запропонованої модифікації алгоритму Густафсона-Кесселя, що ґрунтується на довірчому підході до нечіткої кластеризації, що дозволяє рекомендувати його до використання на практиці для вирішення задач автоматичної кластеризації. Пропонований метод призначений для використання в гібридних системах обчислювального інтелекту та, насамперед, у завданнях навчання штучних нейронних мереж, нейронечітких систем, а також у завданнях кластеризації та класифікації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Likas, A., Vlassis, N., & Verbeek, J. J. (2003). The global k-means clustering algorithm. Pattern recognition, 36(2), 451-461.

2. Arthur, D., & Vassilvitskii, S. (2006). k-means++: The advantages of careful seeding. Stanford.

3. Park, H. S., & Jun, C. H. (2009). A simple and fast algorithm for K-medoids clustering. Expert systems with applications, 36(2), 3336-3341.

4. Reynolds, A. P., Richards, G., & Rayward-Smith, V. J. (2004, August). The application of k-medoids and pam to the clustering of rules. In International Conference on Intelligent Data Engineering and Automated Learning (pp. 173-178). Springer, Berlin, Heidelberg.

5. Бідюк, П. І., Касіцький, О. В., & Коршевнюк, Л. О. (2013). Ефективна реалізація EM-алгоритму з використанням технології GPGPU. Research Bulletin of NTUU" Kyiv Polytechnic Institute", (5).

6. Іваницька, А. Ю., Іванов, Д. Є., & Зубик, Л. В. (2019). Розробка методу аналізу складних даних на основі технологій machine learning.

7. McLachlan, G. J., & Rathnayake, S. (2014). On the number of components in a Gaussian mixture model. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 4(5), 341-355.

8. Xuan, G., Zhang, W., & Chai, P. (2001, October). EM algorithms of Gaussian mixture model and hidden Markov model. In Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205) (Vol. 1, pp. 145-148). IEEE.

9. Якимець, Р. В. (2016). Методи кластеризації та їх класифікація. Міжнародний науковий журнал, (6 (2)), 48-50.

10. Павлишенко, Б. М. (2012). Сингулярна декомпозиція матриці семантичних ознак в алгоритмі ієрархічної кластеризації текстових масивів. Математические машины и системы, 1(1), 69-76.

11. Кобилін, І. О. (2019). Нечітка кластеризація часових рядів в інтелектуальному аналізі потоків даних.

12. Ghosh, S., & Dubey, S. K. (2013). Comparative analysis of k-means and fuzzy c-means algorithms. International Journal of Advanced Computer Science and Applications, 4(4).

13. Park, D. C., & Dagher, I. (1994, June). Gradient based fuzzy c-means (GBFCM) algorithm. In Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94) (Vol. 3, pp. 1626-1631). IEEE.

14. Shafronenko, A., Bodyanskiy, Y., Pliss, I., & Patlan, K. (2019, June). Fuzzy Clusterization of Distorted by Missing Observations Data Sets Using Evolutionary Optimization. In 2019 9th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 217-220). IEEE.

15. Бодянский, Е. В., Плисс, И. П., & Шафроненко, А. Ю. ОБ ОДНОМ АЛГОРИТМЕ НЕЧЕТКОЙ КЛАСТЕРИЗАЦИИ ДАННЫХ ВЫСОКОЙ РАЗМЕРНОСТИ. INTELLECTUAL SYSTEMS FOR DECISION MAKING AND PROBLEMS OF COMPUTATIONAL INTELLIGENCE, 249.

16. Denœux, T. (2011). Maximum likelihood estimation from fuzzy data using the EM algorithm. Fuzzy sets and systems, 183(1), 72-91.

17. Кондратенко, Н. Р. (2014). Підвищення адекватності нечітких моделей за рахунок використання нечітких множин типу 2. Research Bulletin of the National Technical University of Ukraine" Kyiv Politechnic Institute", (6), 56-61.

18. Bodyanskiy, Y., & Shafronenko, A. (2013). On-line robust fuzzy clustering based on similarity measure. Системні технології, (6), 11-20.

19. Karnik, N. N., Mendel, J. M., & Liang, Q. (1999). Type-2 fuzzy logic systems. IEEE transactions on Fuzzy Systems, 7(6), 643-658.

20. Mao, J., & Jain, A. K. (1996). A self-organizing network for hyperellipsoidal clustering (HEC). Ieee transactions on neural networks, 7(1), 16-29.

21. Shafronenko, A., Bodyanskiy, Y. V., Klymova, I., & Holovin, O. (2020, May). Online credibilistic fuzzy clustering of data using membership functions of special type. In CMIS (pp. 744-753).

22. De Maesschalck, R., Jouan-Rimbaud, D., & Massart, D. L. (2000). The mahalanobis distance. Chemometrics and intelligent laboratory systems, 50(1), 1-18.

23. Gustafson, D. E., & Kessel, W. C. (1979, January). Fuzzy clustering with a fuzzy covariance matrix. In 1978 IEEE conference on decision and control including the 17th symposium on adaptive processes (pp. 761-766). IEEE.

24. Krishnapuram, R., & Kim, J. (1999). A note on the Gustafson-Kessel and adaptive fuzzy clustering algorithms. IEEE Transactions on Fuzzy systems, 7(4), 453-461.

25. Bodyanskiy, Y., Shafronenko, A., & Mashtalir, S. (2019, May). Online Robust Fuzzy Clustering of Data with Omissions Using Similarity Measure of Special Type. In International Scientific Conference “Intellectual Systems of Decision Making and Problem of Computational Intelligence” (pp. 637-646). Springer, Cham.

26. Lesot, M. J., & Kruse, R. (2008). Gustafson-Kessel-like clustering algorithm based on typicality degrees. In Uncertainty and Intelligent Information Systems (pp. 117-130).

27. Bodyanskiy, Y. V., & Shafronenko, A. Y. (2014). Tables of data with gaps restoration using multivariate fuzzy extrapolation. Системні технології, (6), 11-17.

28. Bodyanskiy, Y., Shafronenko, A., & Volkova, V. (2012). Adaptive fuzzy probabilistic clustering of incomplete data. INFORMATION MODELS & ANALYSES, 112.

29. Khatounian Filho, M., Koki, L., & Aguiar, R. (2019, August). Pattern Classification on Complex System Using Modified Gustafson-Kessel Algorithm. In 11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019) (pp. 714-720). Atlantis Press.

30. Bezdek, J. C. (1980). A convergence theorem for the fuzzy ISODATA clustering algorithms. IEEE transactions on pattern analysis and machine intelligence, (1), 1-8.

31. Shafronenko, A., Dolotov, A., Bodyanskiy, Y., & Setlak, G. (2018, August). Fuzzy clustering of distorted observations based on optimal expansion using partial distances. In 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP) (pp. 327-330). IEEE.

32. Ihaka, R., & Gentleman, R. (1996). R: a language for data analysis and graphics. Journal of computational and graphical statistics, 5(3), 299-314.

33. Chen, C., Razak, T. R., & Garibaldi, J. M. (2020, July). FuzzyR: An extended fuzzy logic toolbox for the R programming language. In 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) (pp. 1-8). IEEE.

34. Bouveyron, C., Celeux, G., Murphy, T. B., & Raftery, A. E. (2019). Model-based clustering and classification for data science: with applications in R (Vol. 50). Cambridge University Press.