# Владислав Поліщук, Ольга Мацуга

**(Дніпро, Україна)**

# ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ КЛАСТЕРИЗАЦІЇ ДАНИХ НА ОСНОВІ МОДИФІКАЦІЙ МЕТОДУ *K-*СЕРЕДНІХ

Задача кластеризації (розбиття об’єктів деякої множини на групи за принципом схожості) постає в багатьох галузях наукової та практичної діяльності. Одним з найбільш відомих та використовуваних методів її розв’язання є *k*-середніх [1].З огляду на широку популярність він реалізований в багатьох програмних продуктах, що підтримують кластеризацію даних. Проте метод має певні недоліки. Наприклад, він вимагає, щоб кількість кластерів була відома заздалегідь, а на практиці це не завжди можливо. Також усі алгоритми, що реалізують цей метод, є ітераційні і вкрай чутливі до вибору початкових центрів кластерів. Для подолання цих недоліків запропоновано значну кількість модифікацій методу, які не знайшли належного впровадження у сучасному програмному забезпеченні. З огляду на це було розроблено програмне забезпечення кластеризації даних«*k-meansclustering*» на основі алгоритмів методу *k*-середніх та їх модифікацій. Його опис представлено в даній роботі.

Метод*k*-середніх з’явився у 50-х рокахминулого століття. Для реалізації його ідеї було запропоновано декілька алгоритмів, як то Ллойда, МакКіна та Хартігана–Вонга. Пізніше з’явилися їх модифікації, націлені на подолання недоліків самого методу та йогокласичних алгоритмів. Так, для автоматичного визначення кількості кластерів було розроблено наступні алгоритми:

1. *X*-means[2]. Модифікація,в якій для кожного кластеру перевіряється доцільність його розбиття на два на основі *BIC* критерію.
2. *G*-means [3].В цьому алгоритмі кластери розглядаються як сукупності точок, що мають нормальний розподіл, і рішення щодо того, чи є сукупність точок кластером з нормальним розподілом,приймається на основі тесту Андерсона-Дарлінга.

Для знаходження початкових центрів кластерів можна відмітити такі алгоритми:

1. Алгоритм *k*-means++ [4].Популярна модифікація алгоритму *k*-середніх (в оригінальній роботі у варіанті Ллойда), яка передбачає знаходження центрів випадково, але з імовірністю пропорційною до квадрату відстані між ними.
2. Алгоритм відсіювання [5].Алгоритм *J* разів кластеризує випадкові, невеликі за обсягом підмножини об’єктів, після чого виконує кластеризацію центрів одержаних кластерів, за результати якої обирає остаточні центри.
3. Алгоритм вибору взірця [6]. Є подібний до *k*-means++.
4. Алгоритм жадібного видалення [6].Він передбачає вибір усіх об’єктів як початкових центрів і покрокове видалення «найгірших» об’єктів, доки не буде одержано *k*-центрів. На великих за обсягом даних стає вкрай повільним, тому розробниками запропонований комбінований алгоритм.
5. Комбінований алгоритм [6].Для скорочення кількості центрів використовується алгоритм вибору взірця, далі – жадібне видалення.

Усі вищезгадані алгоритмиувійшли до ядра програмного забезпечення«*k-meansclustering*», яке надає користувачу наступні можливості:

1. Завантажувати файл з багатовимірними даними з будь-якого носія.
2. Генерувати набори даних на основі таких вхідних параметрів: кількості кластерів, кількості елементів в кластерах, математичних сподіваннь та діагональних дисперсійно-коваріаційних матриць кластерів.
3. Кластеризувати дані одним із п’яти алгоритмів: Ллойда, МакКіна, Хартігана–Вонга, *X-*means або*G-*means. Перші три безпосередньо реалізують метод *k*-середніх, для них кількість кластерів задає користувач. Два останні алгоритми забезпечують автоматичне визначення кількості кластерів, у роботі їх реалізовано на базі алгоритму Хартігана – Вонга. Для них користувач повинен визначити діапазон можливих значень кількості кластерів. У користувача також є можливість обрати, з якою метрикою відстані застосовувати алгоритми (евклідовою чи Махаланобіса) та як обчислювати центри кластерів (як медіану або як середнє арифметичне).
4. Знаходити початкові центри кластерів один із шести алгоритмів: випадковим, *k*-means++, відсіювання, вибором взірця, жадібним видаленням або комбінованим.
5. Візуалізувати багатовимірні дані в площині двох обраних користувачем ознак.
6. Після проведення кластеризації бачити на формі кількість ітераційалгоритму кластеризації, час кластеризації, значення функціонала якості, фінальні центрі кластерів та кількість об’єктів у кожному кластері.

Програма складається з чотирьох файлів: k-means.exe – додаток, що реалізує усі вищезазначені можливості, а також три допоміжні бібліотеки від Accord, що використовуються для обчислення значень квантилів та емпіричної функції нормального розподілу.

Програму реалізовано на мові програмування C# високого рівня під платформу .NET Framework v.4.5.

У процесі розробки програмного забезпечення було використано усі принципи об’єктно-орієнтованого підходу: інкапсуляцію, наслідування та поліморфізм. Розроблено наступні класи:

1. Distance Measurer (Distance Measurer.cs файл). Містить статичні методи Euclidian та Mahalanobis, що використовуються для знаходження відстані між двома *m*-вимірними точками: евклідової та Махаланобіса відповідно. Приклад використання: Distance.Euclid(x,y) – повертає дійсне число.
2. DataLoader (DataLoader.cs файл). Відповідає за взаємодію програми з файловою системою користувача, зчитування даних із файлу та формування структури даних List<Point> з об’єктами вибірки. Містить один статичний асинхронний метод LoadData, що приймає параметр List<Point> points, який заповнює даними, іповертає ім’я файлу.
3. Point (Point.cs файл). Клас описує об’єкт як точку *m*-вимірного простору. Зберігає для об’єкту значення ознак та посиланнядо двох найближчих кластерів (необхідно для реалізації алгоритму Хартігана – Вонга).
4. Cluster (Cluster.cs файл). Клас описує кластер. В ньому зберігаютьсяусі об’єкти кластеру, їх кількість, центр кластеру, попередній центр кластеру, вимірність простору даних. Містить один метод RefreshCenter, в якому реалізовано обчислення нового центру кластера після додавання/видалення об’єктів. Клас реалізує інтерфейс IComparable для підтримки сортування.
5. Center Resolver (Center Resolver.cs файл) –клас, що реалізує усі перелічені вище алгоритми вибору початкових центрів.
6. Clustering Resolver (Clustering Resolver.cs файл) –клас, що реалізує усі перелічені вище алгоритми кластеризації.
7. Solver (Solver.cs файл). Основний клас програми, що використовує усі інші для проведення кластерного аналізу.

Велику кількість циклічних обчислень у програмному забезпечення було виконано з використанням вбудованих можливостей мови C#, а саме технології LINQ to Objects. LINQ (англ. Language-Integrated Query) – проста та зручна мова запитів до даних. В якості джерела даних може виступати будь-який об’єкт, що реалізує інтерфейс IEnumerable,наприклад, масиви, стандартні та узагальнені колекції, а також набори даних DataSet, документ XML. Для використання LINQ необхідно підключити простір імен System.Linq. У програмі активно використовувалися вирази LINQ у вигляді розширюючих методів (ToList, Avarage, Sum, Where, Sort та інших).

Для запобігання зниження продуктивності додатку і підвищення швидкості його відклику було використано прийоми асинхронного програмування. За допомогою класу Task та ключових слів async, await та lock булореалізовано асинхронне виконання певних етапів обробки даних. При цьому нові версії .NET Framework та C# інкапсулюють основну роботузі створення і менеджменту потоків та роботу з пулом потоків. Завдяки цьому більше немає необхідності вручну створювати окремий потік для вирішеннязадачі за допомогою класу Thread. CLR (англ. – Common Language Runtime) динамічно аналізує код і сама вирішує, чи дійсно необхідно виконати поставлену задачу асинхронно, чи продуктивність від цього впаде (або не зміниться) і слід виконати операцію синхронно.

З метою уникнення екстреного припинення виконання програми у разі непередбачених помилок було використано прийоми захисного програмування. Так, у програмі використовуються конструкції try/catch/finally. Якщо в блоці коду, що розміщений у конструкції try, виникає будь-яка виключна ситуація, управління передається блоку catch (обов’язково) або finally (якщо є необхідність виконати набір інструкцій незалежно від того, успішно виконався основний набір команд або відбувся збій), де помилки конвертуються у текстове повідомлення користувачу. Також у додатку використовується ключове слово throw, що слугує для ручного створення виключної ситуації (використовується при валідації: якщо дані були пошкоджені або невірні, збуджується виключення, яке потім оброблюються на іншому рівні додатку).

Створена програма має дружній до користувача графічний інтерфейс (рис. 1) і одразу після запуску готова до роботи із параметрами за замовчуванням, треба лише обрати файл із вибіркою. Після завантаження даних вони візуалізуються у просторі двох вказаних користувачем ознак, на панелі «Info» з’являється ім’я відкритого файлу і відображається вимірність даних. Для проведення кластеризації даних користувач повинен обрати алгоритм кластеризації, алгоритм вибору початкових центрів, метрику відстані між об’єктами, вказати бажану кількість кластерів*k* (для алгоритмів Ллойда, МакКіна та Хартигана–Вонга) абодіапазон значень*k* (для *Х*-means та *G*-means).

За допомогою розробленого програмного забезпечення було проведено ряд обчислювальних експериментів, що мали на меті виявити найкращі алгоритми вибору початкових центрів кластерів та визначення кількості кластерів. Експерименти було здійснено на 12-ти вибірках з різною кількістю об’єктів (від 100 до 10 000), кількістю кластерів (від 2 до 10) та вимірністю даних (від 2 вимірів до 5).

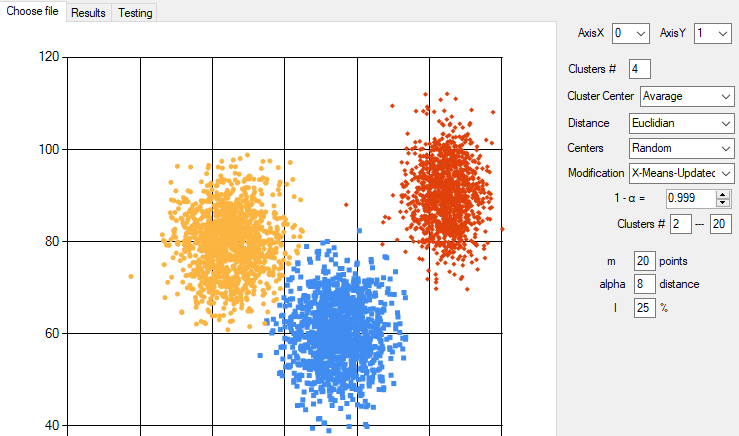


Рисунок 1 – Інтерфейс програми

За результати проведених експериментів можна зробити наступні висновки. Усі реалізовані алгоритми вибору початкових центрів кластерів на невеликих за обсягом, малої розмірності та лінійно роздільних даних показали приблизно однакові, близькі до істинних результати. Тому на таких даних, щоб мінімізувати час проведення кластеризації, доцільно використовувати найпростіші алгоритми вибору центрів, такі як *k*-means++ або вибір взірця. Від випадкового вибору початкових центрів слід відмовитись у будь-якому разі, оскільки його використання, зазвичай, призводить до найгіршої за якістю кластеризації. Більш того, у швидкості він майже не відрізняється від   
*k*-means++, в якому відшукання центрів хоча і займає більше часу, проте сам процес кластеризації збігається швидше через краще початкове наближення. На великих вибірках, щоб отримати розбиття близьке до дійсного, краще застосовувати алгоритм відсіювання, якийпоказав відмінні результати якості за цілком прийнятний час майже на всіх наборах даних. Алгоритм жадібного видалення також показав дуже добрі результати за якістю кластеризації, проте через свою повільну роботи у випадку великих вибірок є неприйнятним.

Серед алгоритмів з автоматичним вибором кількості кластерів (*G-*means та *X-*means) перевагу згідно результатів експериментів має алгоритм*G-*means, У більшості випадків він дозволив одержати оцінку параметра *k*, що збігається зі справжньою кількістю кластерів. *X-*means у свою чергу був схильний давати завищену оцінку шуканого параметра *k.* Крім того, кластеризація алгоритмом *G-*means виявилася швидшою, хоча обидва алгоритми показали себе повільними на великих вибірках. Проте швидкість їх роботи можна збільшити, застосувавши запропоновані авторами оптимізації.

# Література:

1. Steinley D. *K*-means clustering: A half-century synthesis / D. Steinley // British Journal of Mathematical and Statistical Psychology. – 2006. – Vol. 59. –  
    P. 1–34.
2. Pelleg D. *X*-means: Extending *K*-means with Efficient Estimation of the Number of Clusters / D.P elleg, A. Moore // Proceedings of the Seventeenth International Conference on Machine Learning. – 2000. – P. 727–734.
3. Hamerly G. Learning the *k* in *k*-means / G. Hamerly, C. Elkan // Proceedings of the 16th International Conference on Neural Information Processing Systems. – 2003. – P. 281–288.
4. Arthur D. k-means++: The Advantages of Careful Seeding / D. Arthur,   
   S. Vassilvitskii // Proceedings of the 18th annual ACM-SIAM symposium on Discrete algorithms. New Orleans, Louisiana, USA, January 07–09, 2007. – 2007. – P. 1027–1035.
5. Bradley P. S. Refining Initial Points for K-Means Clustering / P. S. Bradley, U. M. Fayyad // Proceedings of the 15th International Conference on Machine Learning (ICML’98). – 1998. – P. 91–99.
6. Ostrovsky R. Effectiveness of Lloyd-Type Methods for the *k*-Means Problem / R. Ostrovsky, Y. Rabani, L. Schulman, C. Swamy // Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS’06). – 2006. – P. 165–176.