УДК 004.89

МОДИФІКАЦІЯ АЛГОРИТМУ ІЄРАРХІЧНОЇ КЛАСТЕРИЗАЦІЇ ХАМЕЛЕОН НА ОСНОВІ СХОЖОСТІ ОБ'ЄКТІВ

Т.Б. Шатовська
Кандидат технічних наук, доцент*
Контактний тел.: 0958252745
E-mail: shatovska@gmail.com
О.І. Онопрієнко*
Контактний тел.: 063-855-50-74
E-mail: oksana.onoprienko@gmail.com
М.Г. Олійник*
Контактний тел.: 066-392-50-57
E-mail: onoprienko.o@mail.ru
*Кафедра програмної інженерії
Харківський національний університет радіоелектроніки
вул. Бакулина,10, м. Харків, Україна, 61166

В даній статті представлено змінений ієрархічний алгоритм кластеризації, в якому використано основну ідею алгоритму Chameleon та модифіковано алгоритми роботи з графами на різних етапах. Проведено експериментальне дослідження ефективності цього методу

Ключові слова: chameleon, ієрархічна кластеризація, модифікація, граф

В этой статье представлен измененный иерархический алгоритм кластеризации, в котором использована основная идея алгоритма Chameleon и модифицированы алгоритмы работы с графами на разных этапах. Проведено экспериментальное исследование эффективности этого метода

Ключевые слова: chameleon, иерархическая кластеризация, модификация, граф

In this paper we present a modified hierarchical clustering algorithm that used the main idea of Chameleon and modified algorithms for graphs at different stages. It was investigated experimentally the effectiveness of this method

Keywords: chameleon, hierarchical clustering, modification, graph

1. Вступ

Кластерний аналіз є широко застосовуваним інструментом для дослідження структури великої кількості об'єктів та зв'язків між ними у різних теоретичних та практичних проблемах штучного інтелекту, прийнятті рішень, розпізнаванні образів, обробки різнорідної інформації та інших областей. Кластерний аналіз найбільш яскраво відображає риси багатовимірного аналізу у класифікації, в той час як факторний аналіз – у дослідженні зв'язку. Головне призначення кластерного аналізу – розбиття множини об'єктів та ознак на однорідні, у відповідному розумінні, групи чи кластери. Це означає, що вирішується задача класифікації даних та виявлення відповідної структури у ній. Методи кластерного аналізу можна застосовувати в різних випадках, навіть тоді, коли мова йде про просте групування, в якому все зводиться до утворення груп по кількісній схожості. Велика перевага кластерного аналізу у тому, що він дозволяє проводити розбиття об'єктів не по одному параметру, а по цілому набору ознак.

2. Аналіз існуючих методів ієрархічної кластеризації

Сьогодні існує достатньо багато методів кластерного аналізу. Обираючи між ієрархічними та неієрархічними методами, необхідно враховувати наступні їх особливості.

Ієрархічні методи, на відміну від неієрархічних, відмовляються від визначення числа кластерів, а будують повне дерево вкладених кластерів. Складнощі ієрархічних методів кластеризації такі: обмеження об'єму набору даних; вибір міри близькості; негнучкість отриманих класифікацій. Перевага цієї групи методів порівняно з неієрархічними методами – їх наочність і можливість отримати детальне уявлення про структуру даних [1].

При використанні ієрархічних методів існує можливість достатньо легко ідентифікувати викиди в наборі даних і, в результаті, підвищити якість даних. Ця процедура лежить в основі двокрокового алгоритму кластеризації. Такий набір даних надалі може бути використаний для проведення неієрархічної кластеризації.

Останнім часом були розроблені алгоритми кластерного аналізу, в яких методи ієрархічної кластеризації інтегровані з іншими методами, до таких алгоритмів належать: BIRCH, CURE, CHAMELEON, ROCK.

Серед вище перерахованих методів, слід виділити метод ієрархічної кластеризації – Хамелеон, що використовує нові поняття такі як зв'язаність об'єктів усередині кожного кластера і близькість кластерів між собою. Даний алгоритм призначений для роботи на вибірках, що мають великий об'єм і високий ступінь зашумленості. Автор даного алгоритму вказує на незалежність якості кластеризації від форм досліджуваних кластерів, проте, як показали наші

дослідження, дане ствердження може бути спростоване на сферичній формі кластерів, вкладених один в одного. Більш того, основним недоліком розглянутих алгоритмів BIRCH, Clarans, CURE є та обставина, що вони вимагають завдання деяких порогів щільності точок, а це не завжди прийнятно [2]. Ці обмеження зумовлені тим, що описані алгоритми кластерного аналізу орієнтовані на надвеликі бази даних і не можуть користуватися великими обчислювальними ресурсам.

Постановка задачі дослідження

Для отримання фінальної форми кластерів на наступних кроках алгоритму буде використана ідея об'єднання підкласів, запропонована в [3], модифікована для роботи на кластерах будь-яких форм і ступеня зашумленості.

В даній роботі ставиться задача:

1) Реалізувати процедуру побудови асиметричного графу зв'язаності об'єктів за принципом k-найближчих сусідів.

2) Для зменшення розмірності вихідної вибірки, що дасть можливість проводити кластеризацію на великих об'ємах даних, реалізувати алгоритм, покращений відносно стандартного підходу до побудови гіперграфу у частині злиття гіпервершин.

3) Реалізувати алгоритм оптимального розбиття гіперграфу на підграфи на підставі запропонованої схеми розрахунку гіперребер.

4) Розробити коефіцієнт оцінки схожості підграфів для реалізації етапу злиття фінальних кластерів.

5) Розробити програмну систему, що реалізує динамічну модель ієрархічної кластеризації та провести експериментальне дослідження її ефективності на множені стандартних вибірок.

3. Алгоритм кластеризації SaTyr

Алгоритм кластеризації SaTyr складається з 4х основних етапів:

1. Побудова k-NN графу.
2. Побудова гіперграфу.
3. Поділ гіперграфу на частини.
4. Об'єднання гіперграфів для отримання фінальної кластеризації.

Розглянемо детальніше кожен з цих етапів.

Побудова k-NN графу.

В запропонованому алгоритмі використовується представлення даних у вигляді асиметричного графу k найближчих сусідів (k-NN граф). Кожна вершина k-NN графу являє собою елемент даних. Існує кілька підходів до побудови k-NN графу. У Хамелеоні використовується симетричний k-NN, тобто такий, у якого ребро між вершинами u і v існує, якщо і вершина u знаходиться у переліку k найближчих до вершини v, і вершина v знаходиться у переліку k найближчих до вершини u. В алгоритмі SaTyr використовується асиметричний k-NN: для існування ребра достатньо, щоб хоча б одна з вершин знаходилась у переліку k найближчих до іншої. Для обчислення близькості об'єктів використовується Евклідова відстань. В алгоритмі Хамелеон запропоновано в якості ваги використовувати величину, обернену до відстані. Але при такому підході вага ребра не є нормалізованою. Тому в нашому алгоритмі вага ребра між двома вершинами x та y обчислюється як

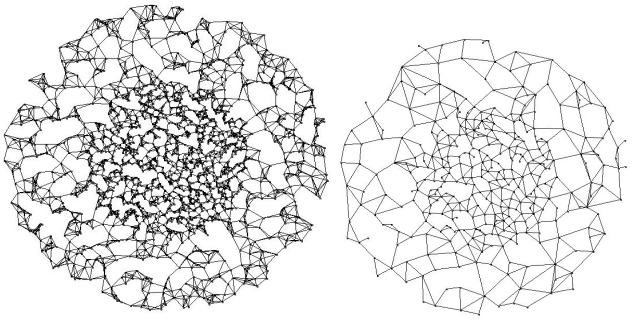
$$\text{weight}(x,y) = \sqrt{m} - \text{dist}(x,y), \quad (1)$$

де m – кількість атрибутів;

dist(x,y) – відстань між об'єктами x та y.

Чим більш схожими є два об'єкти, тим важчим буде ребро між ними. На рис. 1а зображено k-NN граф для вибірки "disk in disk" з кількістю сусідів k =5.

Використання асиметричного k-NN графу замість симетричного, який використовується у класичному алгоритмі Хамелеон, дозволяє значно знизити необхідну мінімальну кількість сусідів k та, відповідно, зменшити залежність результатів від цього параметру.



а) б)

Рис. 1. Вибірка "disk in disk": а) k-NN граф з кількістю сусідів k =5; б) гіперграф після третього етапу згортки

Побудова гіперграфу

Під час фази побудови гіперграфу (огрубіння, coarsen) граф G_0 перетворюється на послідовність менших графів G1, G2, ..., Gm так, що $|V_0|>|V_1|>|V_2|>...>|Vm|$. При побудові гіперграфу підмножина вершин графу G_i об'єднується, щоб сформувати вершину більш грубого графу G_{i+1}. Вага вершини грубого графу дорівнює кількості вершин початкового графу, що були об'єднані для утворення цієї гіпервершини.

В алгоритмі SaTyr на першому етапі процесу огрубіння обирається ряд вершин з найбільшим ступенем (тобто з найбільшою кількістю інцидентних ребер) та кожну з них об'єднується з будь-яким сусідом, що не потрапив до обраного ряду. Вершини об'єднуються лише попарно. Етап завершується, коли жодна пара вершин не може бути об'єднаною. На інших етапах вершини відвідуються у випадковому порядку та кожна вершина u зливається з вершиною v такою, що вага гіперребра (u, v) є максимальною. Вершина, об'єднана з іншою, вже не може бути використана на даному етапі.

Традиційно вага гіперребра між двома вершинами u і v грубого графу обчислюється як кількість ребер початкового графу, що поєднують вершини підмножини u з вершинами підмножини v. Але ця величина не дає ніякого уявлення про близькість між цими підмножинами, оскільки при побудові гіперграфу вершини відвідуються у випадковому порядку, незалежно від того, на якій відстані від інших вершин вони знаходяться. У нашому алгоритмі вага гіперре-

бра між вершинами u і v обчислюється як сума ваг усіх ребер, що поєднують вершини підмножини u з вершинами підмножини v. Чим більше ця величина, тим ближчими є підмножини u і v.

Процес огрубіння на кожному рівні зупиняється, коли кількість вершин грубого графу зменшується в 1,7 раз. (рис. 1б)

Поділ гіперграфу на частини.

На цьому етапі відбувається поділ гіперграфу на набір маленьких гіперграфів, при цьому кількість вершин у кожному підграфі має бути достатньою для подальшого застосування динамічного підходу до об'єднання. Для поділу гіперграфу використовується багаторівневий алгоритм k-way. Процес поділу починається з вибору k найважчих вершин, де k = 8, 16, 32. Після цього до кожної з обраних вершин одна за одною приєднуються вершини-сусіди, доки усі вершини не будуть об'єднані у k груп. Таким чином створюється початковий поділ з урахуванням балансу. Наступним етапом є покращення поділу таким чином, щоб значення цільової функції було мінімальним. Цільовою функцією є edge-cut (на відміну від традиційного підходу [4], використовується вага ребер, що поєднують вершини, які належать до різних частин).

Задача знаходження оптимального поділу гіперграфу є NP-повною. Одним з найбільш точних алгоритмів для покращення якості поділу є алгоритм Kernighan-Lin/Fiduccia – Mattheyses. Для кожної вершини обчислюється вигода її переміщення в іншу частину. При класичному підході [4] вигода обчислюється як різниця між кількістю ребер, що поєднують дану вершину з вершинами інших класів, та сумою ваг ребер, що поєднують дану вершину з вершинами цього ж класу. У нашому алгоритмі вигода замість кількості ребер використовується їх вага, це дозволяє враховувати схожість між об'єктами. На кожному проході алгоритму знаходиться вершина з найбільшою позитивною вигодою та переміщується в іншу частину.

Процес завершується, коли переміщення будь-якої вершини не покращує edge-cut, або коли вигода усіх вершин від'ємна.

Об'єднання гіперграфів для отримання фінальної кластеризації.

Під час цієї фази відбувається послідовне об'єднання гіперграфів, отриманих на стадії поділу. Процес об'єднання відбувається на основі ієрархічного агломеративного підходу.

В алгоритмі Хамелеон Для оцінки схожості між двома підкластерами використовується відносна близькість – RC та взаємопов'язаність – RI. Але цей підхід має певні недоліки:

- не враховується схожість підкласів за їх щільністю,

- на кожному етапі необхідно знаходити min-cut bisector (тобто зважену суму ребер, які поділяють підграф на дві приблизно рівні частини) для нового об'єднаного гіперграфу.

Для подолання цих недоліків було запропоновано модифікацію даного підходу.

В алгоритмі SaTyr для визначення схожості між підкласами використовується величина CS (Cluster Similarity):

$$CS = \frac{|c_{ij}|}{\min(|c_i|,|c_j|)} * \left(\frac{s_{ij}}{\frac{|c_i|}{|c_i|+|c_j|} s_i + \frac{|c_j|}{|c_i|+|c_j|} s_j} \right)^{\alpha} * \left(\frac{\min(s_i, s_j)}{\max(s_i, s_j)} \right)^{\beta}, \quad (2)$$

де $|c_{ij}|$ – кількість ребер, що поєднують вершини підкласу i та вершини підкласу j;

$|c_i|, |c_j|$ – кількість ребер всередині підкласу i та j відповідно;

$|s_{ij}|$ – середня довжина ребер, що поєднують вершини підкласу i та вершини підкласу j;

$|s_i|, |s_j|$ – середня довжина ребер всередині підкласу i та j відповідно;

α, β – задаються користувачем.

Перша частина формули являє собою кількість ребер, що поєднують два класи по відношенню до кількості ребер у меншому з класів. Це дозволяє враховувати пов'язаність двох підграфів. Вираз у перших дужках є аналогічним до відносної близькості, але замість минкат бісектору використовується сума ваг усіх ребер всередині підкласу. Вираз у других дужках показує, наскільки схожими є два підкласи. Чим більше значення цього виразу, тим більш схожими є підкласи. На кожному етапі для об'єднання обирається та пара підкласів, для який ця міра є максимальною. Процес об'єднання завершується, коли кількість класів дорівнює заданій користувачем величині, або якщо немає пов'язаних між собою класів (це можливо, якщо початковий k-NN граф був не повністю зв'язним).

Наведений алгоритм дозволяє проводити кластеризацію на складних вибірках (різна щільність класів, лінійно нероздільні випадки) з високою точністю.

5. Експериментальні дослідження

У своїх експериментах ми використовували вибірку "disk in disk" (два лінійно неподільні класи, що мають різну щільність та кругову форму), вибірки, запропоновані авторами алгоритму Хамелеон, та синтетичні вибірки, в яких змодельовано випадки, складні для кластеризації.

Як вже було зазначено вище, традиційний алгоритм має певні недоліки. Так на етапі поділу гіперграфу не враховується близькість між вершинами грубого графу. В алгоритмі SaTyr нам вдалося подолати цей недолік. На рис. 2 (а) показано результати, отримані при використанні класичного алгоритму та (б) результати, отримані при використанні підходу, запропонованого у SaTyr. Видно, що підхід, використаний у SaTyr здійснює поділ по природній границі між класами, в той час як використання традиційного підходу не дає можливості чітко визначити границю між кластерами.

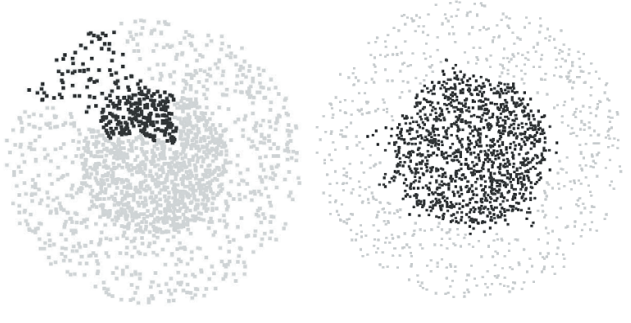
На етапі об'єднання підкластерів традиційний алгоритм Хамелеон не враховує схожість підкласів між собою, а перевіряє лише, щоб їх близькість (RC) та взаємопов'язаність (RI) були високими. У алгоритмі SaTyr формулу для обчислення схожості підкласів було модифіковано таким чином, щоб окрім цих двох мір враховувати схожість між підкласами. На рис. 3 наведено фінальну кластеризацію методами Хамелеон та SaTyr. Наочно видно, що із кластеризацією лінійно неподільних випадків, де

кластери мають різну щільність SaTyr справляється краще.

а) б)

Рис. 2. Вибірка “disk in disk”, етап розбиття: а)використання класичного методу; б) використання алгоритму SaTyr

Експерименти, проведені на багатьох вибірках показали, що алгоритм SaTyr має дуже низьку чутливість до вхідних параметрів (кількість сусідів для побудови k-NN графу, мінімальний розмір підкласу після етапу поділу). Вірний результат було отримано вже при кількості сусідів k=5, збільшення цього параметру призводить до незначного підвищення точності. Єдиною вимогою до розміру підграфу є достатня кількість його вершин для обчислення міри схожості CS.



а) б)

Рис. 3. Вибірка “disk in disk”, фінальна кластеризація, кількість сусідів k =5: а) використання алгоритму Хамелеон; б) використання алгоритму SaTyr

На рис. 4 та 5 наведена фінальна кластеризація для синтетичних вибірок, що містять кластери різної щільності.

Рис. 4. Вибірка “2 ромби”, k =5

Рис. 5. Фігури, що перетинаються, k =5

Експерименти показали, що якість кластеризації знижується на 10% при наявності великої кількості шумів.

Для реалізації алгоритму SaTyr було розроблено відповідне програмне забезпечення.

6. Висновки

Виділення та кластеризація складних форм образів при наявності великої кількості шумів у вихідній вибірці є складною актуальною науково-дослідною задачею. На сьогоднішній день в області кластеризації існує досить велика кількість методів, однак лише деякі з них здатні вирішувати задачі кластеризації з достатньою точністю. Найбільш ефективним методом є Хамелеон, однак і він має свої обмеження в точності на певних формах кластерів. У роботі проведено детальне експериментальне дослідження ефективності методу Хамелеон, виділено об’єктивні причини та форми кластерів на яких даний метод не може одержати точну границю між кластерами. В роботі запропоновано модифікацію даного підходу для покращення якості кластеризації на складних вибірках.

Литература

1. Karypis G. Multilevel hypergraph partitioning: Application in VLSI domain [Text] / G. Karypis , R. Aggarwal, V. Kumar, Sh. Shekhar // Proceedings of the Design and Automation Conference – 1997.
2. Zhang T. BIRCH: an efficient data clustering method for very large databases/ T. Zhang, R. Ramakrishnan, M. Livny// SIGMOD’96 – 1996 – pp. 103-114.
3. Karypis G. CHAMELEON: A Hierarchical Clustering Algorithms Using Dynamic Modeling [Text] / G. Karypis, E.-H. Han, V. Kumar // IEEE Computer – 1999 – 32(8) – pp. 68- 75.
4. Karypis G. Multilevel k-way hypergraph partitioning [Text]/ G. Karypis, V. Kumar // Proceedings of the Design and Automation Conference –1999.