УДК 621.39

О. М. Ткаченко, Н. О. Біліченко,
О. Ф. Грійо Тукало, О. В. Дзісь
Вінницький національний технічний університет
Хмельницьке шосе, 95, 21021 Вінниця, Україна

Метод кластеризації на основі послідовного запуску *k*-середніх з обчисленням відстаней до активних центроїдів

Розглянуто один із варіантів розв'язку задачі кластеризації на основі алгоритму k-середніх, який широко застосовується в багатьох сферах науки і техніки. Головними недоліками алгоритму k-середніх є залежність результатів кластеризації від вибору початкової конфігурації центроїдів (ініціалізації) та збіжність до локального мінімуму цільової функції. Запропонований в роботі вдосконалений метод k-середніх дозволяє отримати розв'язок, наближений до глобального мінімуму спотворення шляхом послідовного запуску k-середніх для $1,2,\ldots,k$ *центроїдів. Значне прискорення роботи досягається за рахунок обчислення відстаней лише до активних центроїдів, а також зменшення кількості векторів-кандидатів на вибір місця початкового розташування нового центроїду. Перевага даного підходу суттєво зростає за великих обсягів даних і зі збільшенням розмірності. Запропонований алгоритм доцільно використовувати в задачах кластеризації мовленнєвих даних при створенні кодових книг.*

Ключові слова: *кодові книги, кластеризація, k-середніх, центроїди, kd-дерева.*

Вступ

Кластеризація даних є відомою науковою та практичною задачею, а саме: як розподілити експериментально отримані набори векторів за групами, або кластерами. Кластеризація часто використовується, зокрема, при статистичному аналізі даних, векторній квантизації, розпізнаванні образів тощо. В галузі ущільнення мовлення алгоритми кластеризації застосовуються для створення кодових книг — спеціальних таблиць, що містять найбільш репрезентативні набори даних.

Задачу кластеризації можна сформулювати так: заданий набір з n векторів, кожен з яких має розмірність d ; необхідно розбити на підмножини відповідно до

заданого критерію оптимізації. Як правило, таким критерієм є мінімізація спотворення. Існують різні шляхи оцінювання спотворення, але в більшості прикладних реалізацій використовують суму середньоквадратичних Евклідових відстаней між вектором і центром кластера (центроїдом), до якого він належить.

Метод кластеризації *k*-середніх є найбільш розповсюдженим і найкраще дослідженим серед усіх методів кластеризації. Він мінімізує вищезгадане спотворення, розподіляючи дані між регіонами, що не перетинаються та ідентифікуються за їхніми центрами. Поширеність методу *k*-середніх зумовлено його головними перевагами: простотою, гнучкістю, швидкою збіжністю. Проте привабливість методу суттєво обмежується його недоліками, зокрема:

— результати кластеризації за методом *k*-середніх значною мірою залежать від вибору початкової конфігурації центроїдів (ініціалізації);

— робота алгоритму суттєво уповільнюється під час кластеризації великих обсягів даних;

— алгоритм може сходитися до локального мінімуму цільової функції.

Щоб позбутися цих недоліків було запропоновано низку модифікацій методу *k*-середніх. У методі *k*-середніх++ (*k*-means++) введено вдосконалену процедуру ініціалізації, що дозволяє покращити результати кластеризації за рахунок спеціального вибору початкової конфігурації центроїдів [1]. Для прискорення процесу обчислення відстаней від точок до центроїдів у [2] запропоновано відкидати з розгляду статичні центроїди, тобто такі, що залишилися на своїх позиціях на поточній ітерації. З метою запобігання локальній збіжності в [3] запропоновано ітеративний алгоритм, що дозволяє наблизитися до глобального оптимуму шляхом покрокового послідовного запуску *k*-середніх. У [4] даний метод було розвинуто шляхом застосування особливої структури даних — *kd*-дерев, що дозволило суттєво зменшити обчислювальну складність методу.

У даній роботі поєднуються переваги розглянутих підходів з метою вдосконалення методу кластеризації *k*-середніх, в якому розв'язок, що наближений до глобального мінімуму, отримується шляхом послідовного запуску *k*-середніх для $1,2,...,k$ центроїдів, і значне прискорення роботи досягається завдяки обраховуванню відстаней лише до тих центроїдів, що змінили своє розташування на попередній ітерації, а також зменшенню кількості векторів-кандидатів на вибір місця початкового розташування нового центроїда.

Глобальний алгоритм *k*-середніх

Кластеризація за методом *k*-середніх розподіляє вхідний набір векторів по k кластерах $\mathbf{S}_i (i = 1, 2, ..., k)$, з кожним з яких пов'язаний центроїд $\mathbf{c}_i$. Позначимо множину вхідних векторів $\mathbf{S} = \{\mathbf{x}\}, |\mathbf{S}| = n$. Нехай $D(\mathbf{x}, \mathbf{c})$ — відстань між вектором $\mathbf{x}$ та центроїдом $\mathbf{c}$. У цій статті використовується незважена Евклідова відстань між вектором $\mathbf{x} = (x_1, x_2, ..., x_d)$ та центроїдом $\mathbf{c} = (c_1, c_2, ..., c_d)$:

$$D^2(\mathbf{x}, \mathbf{c}) = \sum_{i=1}^{d} (x_i - c_i)^2 .$$

Основу кластеризації за методом *k*-середніх складає процес перетворення множини центроїдів в іншу множину, поліпшену за критерієм мінімізації спотворення шляхом перерозподілу вхідних векторів між кластерами. Кластеризація розпочинається з деякого початкового задання центроїдів, і процес перетворення повторюється, поки не задовольняється умова завершення.

Позначимо множину центроїдів, отриманих на ітерації t, $\mathbf{SC}_t = \{\mathbf{c}_i\}$. Алгоритм кластеризації *k*-середніх в його звичайному варіанті описується наступним чином.

1. Встановлюємо $t = 0$ та задаємо початкове розташування центроїдів $\mathbf{SC}_0$.

2. Для заданої множини центроїдів $\mathbf{SC}_t$ для отримання поліпшеної множини центроїдів $\mathbf{SC}_{t+1}$ виконуємо пункти 2.1 та 2.2.

2.1. Знаходимо таке розбиття $\mathbf{S}$, що розподіляє $\mathbf{S}$ по k кластерах $\mathbf{S}_i (i = 1,2,...,k)$ та задовольняє умові:

$$\mathbf{S}_i = \{\mathbf{x} \mid D(\mathbf{x}, \mathbf{c}_i) \le D(\mathbf{x}, \mathbf{c}_j) \ \forall j \ne i\}.$$

2.2. Обчислюємо центроїд $\mathbf{c}_i$ для кожного кластера $\mathbf{S}_i (i = 1,2,...,k)$, щоб отримати нову множину центроїдів $\mathbf{SC}_{t+1}$:

$$c_i = \frac{1}{m_i} \cdot \left(\sum_{l=1}^{m_i} x_{lj}\right), \ j = 1,2,...,d, \qquad (1)$$

де m_i — кількість векторів, що належать кластеру $\mathbf{S}_i$.

3. Обчислюємо сумарне спотворення $E^2 = \sum_{\mathbf{x} \in S} D^2(\mathbf{x}, \mathbf{c})$ для $\mathbf{SC}_{t+1}$. Якщо воно відрізняється від отриманого на попередній ітерації на достатньо малу величину, припиняємо процес. В іншому випадку присвоюємо $t \leftarrow t + 1$ та повертаємося до кроку 2.

Як зазначалося вище, наведеному алгоритму кластеризації *k*-середніх притаманні певні недоліки. З метою їхнього подолання в [3] запропоновано вдосконалений варіант кластеризації *k*-середніх, названий авторами жадібним глобальним алгоритмом *k*-середніх (greedy global *k*-means algorithm). В його основі лежить припущення, що глобальний оптимум може бути досягнутий шляхом запуску *k*-середніх, коли $(k-1)$ центроїд розташований в оптимальних позиціях, що отримані як розв'язок задачі кластеризації для $(k-1)$ центроїда, а k-й центроїд має бути розміщений у відповідній позиції, яку й треба визначити. Оптимальну кластеризацію для $k = 1$ легко отримати, обчисливши координати першого центроїда як середньоарифметичне відповідних координат усіх векторів множини $\mathbf{S}$. Таким чином, реалізація даного підходу для отримання k центроїдів потребує послідовного запуску *k*-середніх для $1,2,...,k$ центроїдів. Окремо слід зазначити, що пошук належної позиції *i*-го центроїда, яка є невідомою, коли відомі позиції попередніх

$(i-1)$ центроїдів, потребує запуску *k*-середніх для кожного вектора $\mathbf{x}_i \in \mathbf{X}$, який розглядається як кандидат на позицію вставки нового центроїда. Остаточно вибирається той варіант, який забезпечує мінімальне сумарне спотворення для всіх i центроїдів.

Метод *k*-середніх з обчисленням відстаней до активних центроїдів

Неважко показати, що, на відміну від звичайної кластеризації *k*-середніх, складність якої оцінюється як $O(nk)$, складність жадібного глобального алгоритму *k*-середніх становить $O(n^2k^2)$. Це означає, що для практичних застосувань, де кількість векторів складає кілька десятків або сотень тисяч, а кількість кластерів — кілька тисяч, час роботи алгоритму є занадто тривалим. Тому в [3] було запропоновано для пошуку належної позиції i-го центроїда обмежитися вибором того вектора, який забезпечує мінімальне спотворення при додаванні його як початкової позиції нового центроїда замість запуску *k*-середніх для кожного вектора. Крім того, зменшення обчислювальної складності можливо за рахунок використання *kd*-дерев для генерації нових центроїдів, а також для знаходження центроїда, найближчого до даного вектора [5, 6].

У даній роботі пропонується інший підхід, що має на меті зменшення часу роботи глобального алгоритму *k*-середніх.

Насамперед відзначимо, що найбільш трудомісткою складовою частиною обчислень є знаходження центроїда, найближчого до даного вектора, оскільки це потребує обчислення відстаней від кожного вектора до кожного центроїда. Проте в міру роботи алгоритму відносно невелика частка центроїдів змінюють свої положення, а більшість з них залишається на своїх позиціях. Надалі будемо називати центроїди, що змінюють своє положення на ітерації t, активними, а ті центроїди, які залишаються на своїх позиціях — пасивними. Відповідні множини позначимо як $\mathbf{SC}_t^{(a)}$ та $\mathbf{SC}_t^{(p)}$. Таким чином, якщо зберігати для кожної точки відстані до всіх центроїдів на ітерації t, на ітерації $t+1$ достатньо обчислити відстані лише до активних центроїдів $\mathbf{SC}_t^{(a)}$. При цьому виграш у часі буде тим більший, чим менша відносна частка r_t активних центроїдів серед усіх центроїдів на ітерації t:

$$r_t = \frac{\left|\mathbf{SC}_t^{(a)}\right|}{\left|\mathbf{SC}_t\right|}, \tag{2}$$

де $\left|\mathbf{SC}_t^{(a)}\right|$ та $\left|\mathbf{SC}_t\right|$ — потужності множин $\mathbf{SC}_t^{(a)}$ та $\mathbf{SC}_t$ відповідно.

Для досліджень у галузі ущільнення мовлення типовою є ситуація, коли для створення кодових книг розміром 1024–4096 центроїдів використовуються тренувальні набори, що складаються з 60000–200000 векторів.

На рис. 1 показано, як змінюється доля активних центроїдів *r* із зростанням загальної кількості центроїдів $\left|\mathbf{SC}\right|$, яка отримана в результаті кластеризації 75000 векторів розмірністю $d = 5$. Дані по обох осях наведено у логарифмічному масш-

табі. Для згладжування кривої, що представлена на рис. 1, дані усереднювалися за 1000 ітерацій.

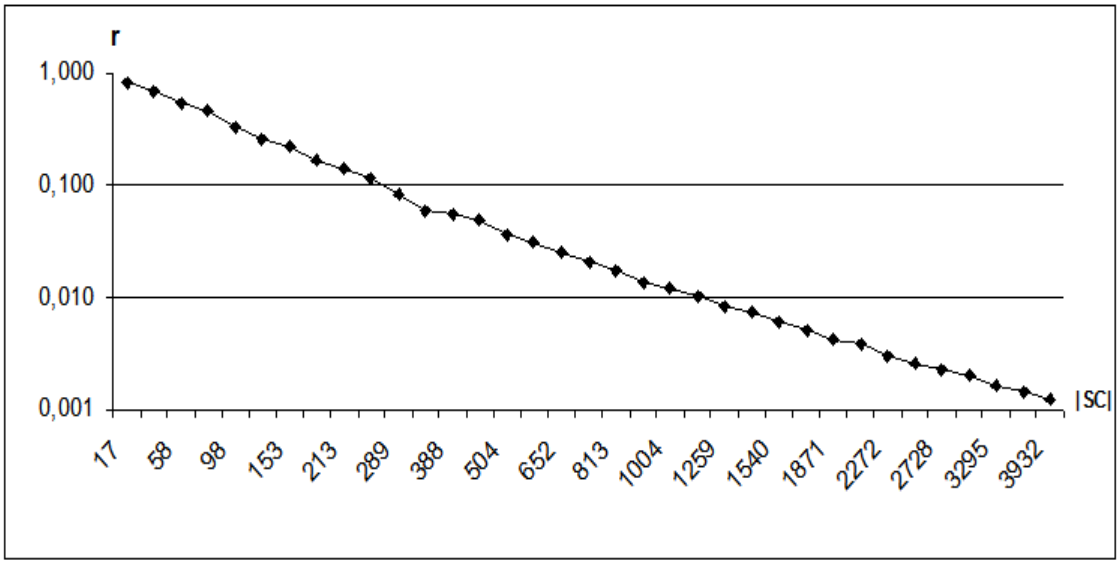


Рис. 1. Залежність долі активних центроїдів від загальної кількості центроїдів

Як можна побачити, в міру зростання $|\mathbf{SC}|$ від 2 до 40000, r поступово зменшується від 0,9 до 0,0012. Середня доля активних центроїдів r_{av} складає приблизно 0,015. Таким чином, кількість обчислень відстаней має зменшитися в 50–100 разів порівняно з глобальним алгоритмом *k*-середніх. Важливо відзначити, що центри кластерів не будуть відрізнятися від тих, що отримані при застосуванні алгоритму глобальної кластеризації.

Разом з тим слід зауважити, що зазначений виграш у швидкодії досягається за рахунок суттєвого збільшення витрат пам'яті, оскільки для кожного вектора необхідно зберігати відстані до всіх центроїдів. Проте ці витрати можна значно скоротити, якщо для кожного вектора зберігати відстані не до всіх, а тільки до m найближчих центроїдів.

Позначимо положення *i*-го центроїда на поточній та наступній ітераціях відповідно як $\mathbf{c}_i$ та $\mathbf{c}'_i$. Нехай $\mathbf{W} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_m\}$ — множина центроїдів, найближчих до вектора $\mathbf{x}$ на поточній ітерації. Позначимо $D_{\max}$ максимальну з відстаней від вектора $\mathbf{x}$ до центроїдів з $\mathbf{W}$: $D_{\max} = D(x, c_i) \geq D(x, c_j), \forall j \neq i, \quad i, j = 1, 2, ..., m$. Нехай $\mathbf{W}$ складається з активних $\mathbf{W}^{(a)}$ і пасивних $\mathbf{W}^{(p)}$ центроїдів. Таким чином, $\mathbf{W} = \mathbf{W}^{(a)} \bigcup \mathbf{W}^{(p)}$. Пропонується обирати центроїд, найближчий до точки $\mathbf{x}$, з множини $\mathbf{SC}^{(a)} \bigcup \mathbf{W}^{(p)} = \mathbf{SC}^{(a)} \setminus \mathbf{W}^{(a)} \bigcup \mathbf{W}$. Для кожного кластера $\mathbf{c}_i \in \mathbf{SC}^{(a)}$ обчислюється відстань $D(\mathbf{x}, \mathbf{c}_i)$. Якщо $D(\mathbf{x}, \mathbf{c}_i) < D_{\max}$, центроїд $\mathbf{c}_i$ включається до $\mathbf{W}_t = \mathbf{W}_t \bigcup \mathbf{W}$. Найближчий до вектора $\mathbf{x}$ центроїд $\mathbf{c}_j$ вибирається з $\mathbf{W}_t$, виходя-

чи з умови $D(\mathbf{x},\mathbf{c}_j) \le D(\mathbf{x},\mathbf{c}_i), \forall i \ne j$. Після цього з $\mathbf{W}_t$ формується множина $\mathbf{W}$ з m центроїдів, найближчих до вектора $\mathbf{x}$ для наступної ітерації.

При даному підході може виникати помилка у визначенні найближчого центроїда, що ілюструється на рис. 2 для $m = 2$. Припустимо, найближчі до $\mathbf{x}$ центроїди $\mathbf{c}_1$ та $\mathbf{c}_2$ будуть активними та пересунуться на позиції $\mathbf{c}_1'$ та $\mathbf{c}_2'$ відповідно, в результаті чого найближчим до $\mathbf{x}$ стане пасивний центроїд $\mathbf{c}_j$, відстань до якого не зберігалась і не обчислювалася. Проте найближчим до $\mathbf{x}$ буде вважатися центроїд $\mathbf{c}_1'$.

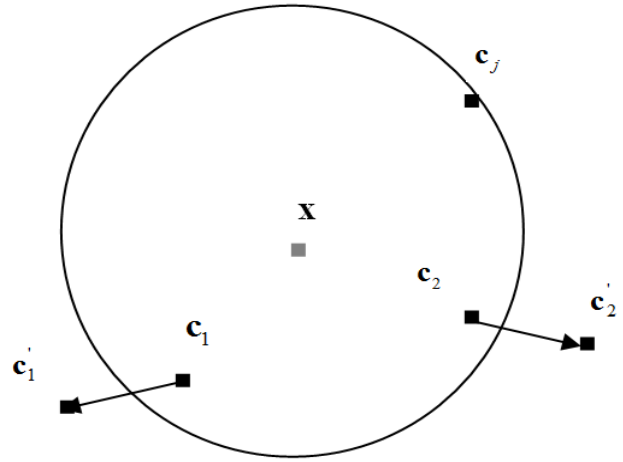


Рис. 2. Можлива помилка при визначенні найближчого центроїда

Припускаючи, що події є незалежними, ймовірність $p^{(er)}$ такої помилки на ітерації t для заданої величини m можна оцінити як

$$p_t^{(er)} = p(\mathbf{c}_j \in \mathbf{SC}_t^{(p)}) \cdot \prod_{i=1}^{m} p(\mathbf{c}_i \in \mathbf{SC}_t^{(a)}). \qquad (3)$$

Замінюючи ймовірності в (3) на їхні статистичні оцінки $p(\mathbf{c}_i \in \mathbf{SC}_t^{(p)}) = \frac{|\mathbf{SC}_t^{(p)}|}{|\mathbf{SC}_t|}$, $p(\mathbf{c}_i \in \mathbf{SC}_t^{(a)}) = \frac{|\mathbf{SC}_t^{(a)}|}{|\mathbf{SC}_t|}$, з урахуванням (2) отримаємо:

$$p_t^{(er)} = \frac{|\mathbf{SC}_t^{(p)}|}{|\mathbf{SC}_t|} \cdot \prod_{i=1}^{m} \frac{|\mathbf{SC}_t^{(a)}|}{|\mathbf{SC}_t|} = \frac{|\mathbf{SC}_t| - |\mathbf{SC}_t^{(a)}|}{|\mathbf{SC}_t|} \cdot \prod_{i=1}^{m} \frac{|\mathbf{SC}_t^{(a)}|}{|\mathbf{SC}_t|} = (1 - r_t) \cdot r_t^m. \qquad (4)$$

Функція (4) має максимум при $r_t = \frac{m}{m+1}$, який дорівнює $p_{\max}^{(er)} = \frac{m^m}{(m+1)^{m+1}}$.

При $r_{av} = 0,015$ $p_{av}^{(er)} = 0,985 \cdot 0,015^m$. Таким чином, обираючи, наприклад, $m = 4$, можна отримати $p_{av}^{(er)} \approx 5 \cdot 10^{-8} << 1$.

Метод *k*-середніх з обчисленням відстаней до активних центроїдів полягає в наступному.

1. Визначаємо множину точок $\{\mathbf{x}_0\} \subset S$, що будуть використовуватися для визначення початкової позиції для вставки нового центроїда.

2. Присвоївши $k = 1$, обчислюємо координати першого центроїда як середнє значення координат усіх векторів:

$$c_{kj} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_{ij}, \ j = 1, 2, ..., d \ .$$

3. Виконуємо $k \leftarrow k + 1$ і знаходимо початкову позицію для вставки центроїда $\mathbf{c}_k$ шляхом вибору вектора $\mathbf{x} \in \{\mathbf{x}_0\}$, що забезпечує мінімальне спотворення $E^2 = \sum_{\mathbf{x} \in S} D^2(\mathbf{x}, \mathbf{c})$.

4. Запускаємо алгоритм *k*-середніх для *k* центроїдів, для чого виконуємо кроки 4.1–4.3.

4.1. Розділяємо множину $\mathbf{SC}$ на підмножини $\mathbf{SC}^{(a)}$ та $\mathbf{SC}^{(p)}$, що складаються відповідно з активних і пасивних центроїдів.

4.2. Для кожного вектора $\mathbf{x} \in S$ визначаємо $\mathbf{W} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_m\}$, що містить множину з m найближчих до $\mathbf{x}$ центроїдів. Для кожного $\mathbf{c}_i \in \mathbf{SC}^{(a)}$ обчислюємо відстань $r_i = D(\mathbf{x}, \mathbf{c}_i)$. Якщо $\mathbf{c}_i \in \mathbf{W}$, корегуємо відповідне значення відстані r_i. Якщо $\mathbf{c}_i \notin \mathbf{W}$, перевіряємо виконання умови $r_i < r_{\max}$, де $r_{\max} = \max_j D(x, c_j)$, $j = 1, 2, ..., m$. Якщо умова виконується, додаємо $\mathbf{c}_i$ до множини $\mathbf{W}$.

4.3. Використовуючи (1), обчислюємо нове положення центроїдів. Якщо умова збіжності не виконується, повертаємося до 4.1.

4.4. Перевіряємо, чи отримано задану кількість центроїдів $k_{\max}$. Якщо $k < k_{\max}$, повертаємося до 3.

Наведений метод потребує додаткових витрат пам'яті для зберігання індексів m найближчих центроїдів і відстаней до них для кожного вектора.

Крім того, для прискорення перевірки $\mathbf{c}_i \in \mathbf{W}$ пошук відповідного елемента можна організувати за допомогою хешування, що також потребує додаткової пам'яті.

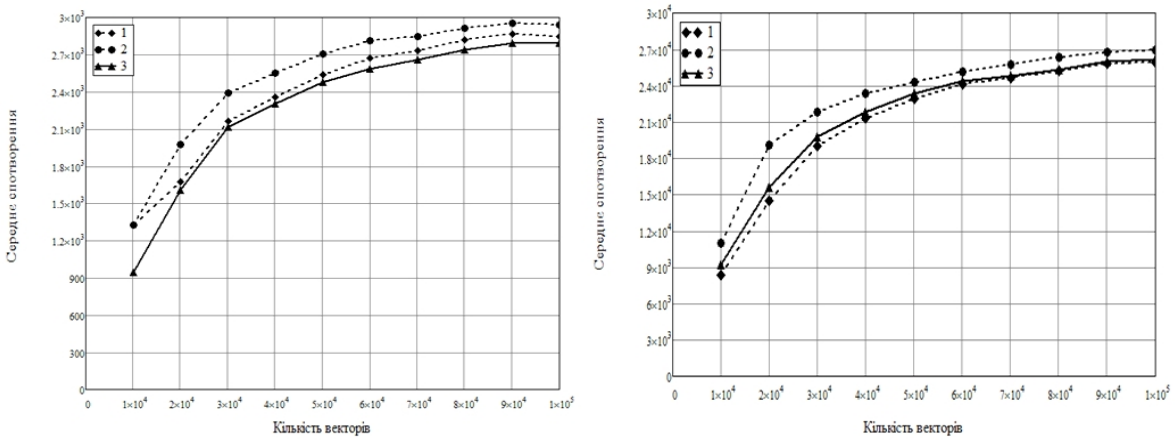
Як видно з рис. 1, з урахуванням (4), для виконання $p^{(er)} << 1 \approx \text{const}$ значення m має бути більшим для малих k та меншим для великих значень k. Проте невелике число $m = 4$ дозволяє уникнути збільшення спотворення та може бути рекомендовано для практичного застосування. Вибір множини $\{\mathbf{x}_0\}$ у пункті 1 може виконуватися за схемою, що застосовується в алгоритмі *k*-means ++.

Експериментальні результати

Ефективність запропонованого в даній статті алгоритму кластеризації з обчисленням відстані тільки до активних центроїдів (Distance Calculation to Active Centroids, DCAC) оцінювалася за середнім спотворенням в перерахунку на один вхідний вектор та часом роботи алгоритму. Досліджувався вплив кількості векторів з вхідного набору на похибку та час кластеризації.

Запропонований алгоритм порівнювався з двома модифікаціями алгоритму *k*-середніх: класичним алгоритмом *k*-середніх (надалі *k*-means), реалізованим в MATLAB без обмеження кількості ітерацій, та реалізованим на основі *kd*-дерев з максимальною кількістю ітерацій 500 (надалі hybrid), запропонованим у [7]. Специфіка алгоритму hybrid полягає в тому, що для наближення до глобального мінімуму в ньому поєднано класичний алгоритм *k*-середніх і локальний пошук (обмін між існуючими центроїдами та кандидатами на центроїди за умови, що такий обмін призводить до зменшення середнього спотворення). Дослідження проводилося на наборі векторів LSF-параметрів [8], які отримані з мовленнєвої бази даних TIMIT. Всі обчислення виконувалися на комп'ютері Intel Core 2 2.0 GHz з 2 Гб пам'яті.

На рис. 3 показано, як змінюється середнє спотворення для різних значень вхідних векторів 5- та 10-вимірної розмірності (відповідно рис. 3,*а* та 3,*б*). Кількість векторів змінюється у діапазоні від 10000 до 100000, кількість згенерованих центроїдів — 4000. Оскільки алгоритм *k*-середніх передбачає велику кількість обчислень відстаней, які можна виконувати одночасно, DCAC також було реалізовано з розпаралелюванням операцій. Введемо позначення, що використовуються на нижче наведених рис. 3 та 4: 1 — *k*-means; 2 — hybrid; 3 — DCAC; 4 — DCAC з розпаралелюванням.



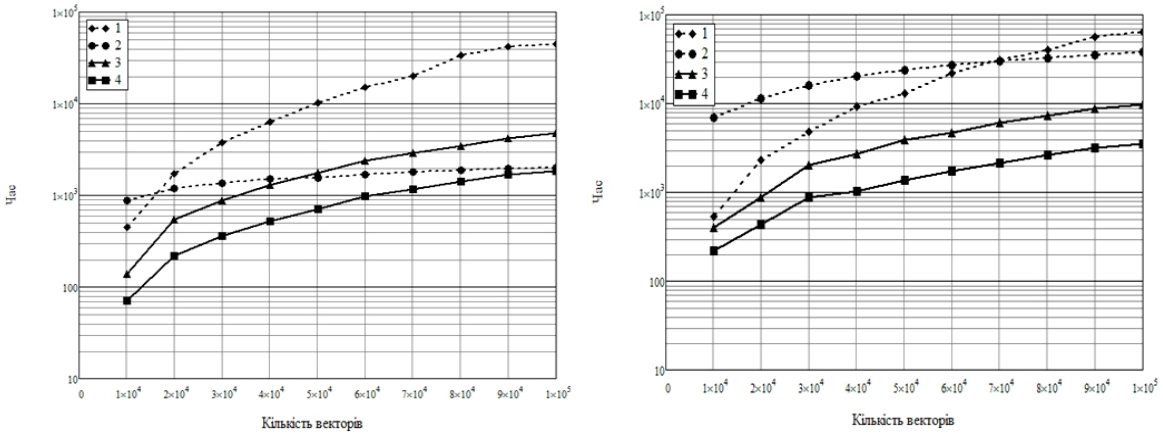
а) б)

Рис. 3. Залежність середнього спотворення від кількості вхідних векторів: а) $d = 5$; б) $d = 10$

З аналізу графіків можна зробити висновок, що в усіх трьох алгоритмах середнє спотворення повільно зростає. Найбільше спотворення виникає при застосуванні алгоритму кластеризації на основі *kd*-дерев, інші два алгоритми відрізня-

ються несуттєво. Зі зростанням розмірності характер залежності майже не змінився, абсолютні ж значення спотворення зросли на порядок.

Ще одним важливим критерієм оцінювання алгоритму є час його роботи. Залежність часу роботи від кількості векторів у вхідному наборі наведено на рис. 4 для розмірностей $d = 5$ і $d = 10$ (відповідно рис. 4,*а* та 4,*б*). Масштаб по осі ординат логарифмічний.



а) б)

Рис. 4. Залежність часу роботи алгоритмів від кількості вхідних векторів: а) $d = 5$; б) $d = 10$

Очевидно, що при збільшенні кількості вхідних векторів час роботи алгоритмів зростає. Найбільш стрімке зростання спостерігається для алгоритму, що реалізований у MATLAB. При застосуванні *kd*-дерев час на обробку 10000 векторів є найбільшим, але зі збільшенням кількості векторів час його роботи зростає найповільніше, і для $d = 5$ (рис. 4,*а*) після проходження точки в 50000 векторів є меншим, ніж у запропонованого в даній статті без розпаралелювання. Часові показники алгоритму DCAC з розпаралелюванням є найкращими.

Зі збільшенням розмірності для алгоритму з використанням *kd*-дерев спостерігається стрімке зростання часу роботи, що зумовлено експоненціальним характером залежності часу від розмірності. Проте для великої кількості вхідних векторів часові показники його роботи є кращими за MATLAB, але поступаються алгоритму DCAC. Зазначимо, що ефективність алгоритму DCAC практично не залежить від розмірності.

Висновки

Запропонований в роботі вдосконалений метод *k*-середніх дозволяє отримати розв’язок, що наближений до глобального мінімуму спотворення. В середньому похибка кластеризації складає на 10–15 % менше, ніж при використанні алгоритму hybrid і знаходиться приблизно на тому ж рівні, що при використанні алгоритму *k*-means (MATLAB). За швидкістю роботи запропонований алгоритм дає результати кращі в 10–15 разів, ніж алгоритм *k*-means (MATLAB), що є особливо важливим під час кластеризації великих обсягів даних. Порівняння з реалізацією

k-середніх на основі *kd*-дерев показує, що швидкість роботи представленого алгоритму є вищою, якщо відношення кількості точок до розмірності складає менше 20000. Вказані характеристики свідчать про доцільність використання наведеного алгоритму в задачах кластеризації мовленнєвих даних при створенні кодових книг.

1. *Arthur D. k*-Means++: The Advantages of Careful Seeding / D. Arthur, S. Vassilvitskii // ACM-SIAM Symposium on Discrete Algorithms (SODA 2007) Astor Crowne Plaza. — New Orleans (Louisiana). — 2007. — P. 1–11.

2. *Lai Jim Z.C.* Fast *k*-Means Clustering Algorithm Using Cluster Center Displacement / Jim Z. C. Lai, Tsung-Jen Huang, Yi-Ching Liaw // Pattern Recognition. — 2009. — N 42(11).

3. *Likas A.* The Global *k*-Means Clustering Algorithm / Aristidis Likas, Nikos Vlassis, Jacob J. Verbeek // Pattern Recognition. — 2003. — Vol. 36, N 2.

4. *Hussein N.* A Fast Greedy *k*-Means Algorithm / Master's Thesis Nr: 9668098 // University of Amsterdam Faculty of Mathematics, Computer Sciences, Physics and Astronomy Euclides Building Plantage Muidergracht 24. — November, 2002.

5. *Alsabti K.* An Efficient *k*-Means Clustering Algorithm / Khaled Alsabti, Sanjay Ranka, Vineet Singh // In Proc. of the First Workshop on High Performance Data Mining. — Orlando, FL. — March, 1998.

6. *Pelleg D.* Accelerating Exact *k*-Means Algorithms with Geometric Reasoning / Dan Pelleg, Andrew Moore // Technical Report CMU-CS-00105. — Carnegie Mellon University/ — Pittsburgh, PA.

7. *A Local* Search Approximation Algorithm for *k*-Means Clustering / [T. Kanungo, D.M. Mount, N.S. Netanyahu et al.] // Computational Geometry: Theory and Applications. — 2004. — N 2. — P. 89–112.

8. *Ткаченко О.М.* Ефективне векторне квантування LSF-параметрів при ущільненні мовних сигналів / О.М. Ткаченко, О.Д. Феферман, С.В. Хрущак // Інформаційні технології та комп'ютерна інженерія. — 2007. — № 1. — С. 124–129.

Надійшла до редакції 23.01.2012