УДК 621.39

О. М. Ткаченко, к. т. н., доц.; О. Ф. Грійо Тукало; О. В. Дзісь; С. М. Лаховець

МЕТОД КЛАСТЕРИЗАЦІЇ НА ОСНОВІ ПОСЛІДОВНОГО ЗАПУСКУ K-СЕРЕДНІХ З УДОСКОНАЛЕНИМ ВИБОРОМ КАНДИДАТА НА НОВУ ПОЗИЦІЮ ВСТАВКИ

У роботі запропоновано вдосконалений метод кластеризації k-середніх, який, на відміну від класичного, дозволяє отримати розв'язок, наближений до глобального мінімуму спотворення шляхом послідовного запуску k-середніх для центроїдів $1, 2, ..., k$. *Зменшення спотворення досягається за рахунок покращеної процедури визначення векторів-кандидатів на вибір позиції вставки нового центроїда без значного сповільнення часу роботи.*

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

Вступ

Кластеризація – це розбиття певної множини об'єктів на підмножини (кластери), що не перетинаються, таким чином, щоб кожен кластер містив схожі об'єкти, а об'єкти різних кластерів відрізнялися між собою. Кластеризацію часто використовують, зокрема, під час статистичного аналізу даних [1], векторної квантизації [2], розпізнаванні образів [3] тощо. У галузі ущільнення мовлення алгоритми кластеризації застосовують для створення кодових книг – спеціальних таблиць, що містять найбільш репрезентативні набори даних. Задача кластеризації даних є важливим елементом загальної проблеми обробки даних, для розв'язання якої існує безліч підходів і алгоритмів: від суто інтуїтивних і евристичних до строго математичних. 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 кластеру (центроїдом), до якого він належить [2, 4].

Метод кластеризації k-середніх є найрозповсюдженішим і найбільшдослідженим серед усіх методів кластеризації. Він мінімізує вищезгадане спотворення, розподіляючи дані між регіонами, що не перетинаються та ідентифікуються за їхніми центрами. Поширеність методу k-середніх зумовлена його головними перевагами: простотою, гнучкістю, швидкою збіжністю. Проте практичне застосування 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 [5]. Для прискорення процесу обчислення відстаней від точок до центроїдів у [6] запропоновано відкидати з розгляду статичні центроїди, тобто такі, що залишилися на своїх позиціях на поточній ітерації. В [7] з метою зменшення обчислювальної складності методу k-середніх застосовано особливу структуру даних – kd-дерева, що дозволило суттєво зменшити обчислювальну складність методу. З метою запобігання локальній збіжності у [8] запропоновано ітеративний алгоритм, що дозволяє наблизитися до глобального оптимуму шляхом покрокового послідовного запуску k-середніх.

Постановка задачі

У цій роботі поєднано переваги розглянутих підходів з метою вдосконалення методу кластеризації k-середніх, у якому розв'язок, наближений до глобального мінімуму, отримують шляхом послідовного запуску k-середніх для $1, 2, ..., k$ центроїдів, і зменшення спотворення досягають за рахунок покращеної процедури визначення векторів-кандидатів на вибір позиції вставки нового центроїда.

Класичний алгоритм k-середніх

Кластеризація за методом k-середніх розподіляє вхідний набір векторів за k кластерами $\mathbf{S}_i (i = 1, 2, ..., k)$, з кожним із 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 використано незважену Евклідову відстань між вектором $\mathbf{x} = (x_1, x_2, ..., x_d)$ та центроїдом $\mathbf{c} = (c_1, c_2, ..., c_d)$:

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

Позначимо множину центроїдів, отриманих на ітерації t, $\mathbf{SC}_t = \{\mathbf{c}_i\}$. Алгоритм кластеризації k-середніх у його звичайному варіанті описують так:

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

2. Для заданої множини центроїдів $\mathbf{SC}_t$ виконуємо дії, зазначені в пунктах 2.1 та 2.2, і отримуємо поліпшену множину центроїдів $\mathbf{SC}_{t+1}$:

2.1 Знаходимо таке розбиття $\mathbf{S}$, що розподіляє $\mathbf{S}$ за k кластерами $\mathbf{S}_i (i = 1, 2, ..., k)$ та задовольняє умову

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

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

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

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.

Алгоритм гарантовано збігається за кінцеве число ітерацій. Похибка кластеризації і число ітерацій залежить від початкового вибору центроїдів, тому звичайною практикою є запуск k-середніх кілька разів з різними початковими кандидатами в центроїди [9].

Глобальний алгоритм k-середніх

Як зазначалося вище, алгоритму кластеризації k-середніх властиві певні недоліки. З метою їхнього подолання в [10] 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 розташовано в оптимальних позиціях, отриманих як розв'язок задачі кластеризації для $(k-1)$ центроїда, а k-й центроїд має бути розміщений у відповідній позиції, яку і треба визначити. Оптимальну кластеризацію

для $k=1$ легко отримати, обчисливши координати першого центроїда як середньоарифметичне відповідних координат всіх векторів множини $\mathbf{S}$. Таким чином, реалізація цього підходу для отримання k центроїдів потребує послідовного запуску k-середніх для 1, 2, ..., k центроїдів.

Глобальний алгоритм k-середніх передбачає, що пошук належної позиції i-го центроїда, яка є невідомою, коли відомі позиції попередніх $(i-1)$ центроїдів, потребує запуску k-середніх для кожного вектора $\mathbf{x}_i \in \mathbf{S}$ з набору вхідних векторів, який розглядають як кандидата на позицію вставки нового центроїда. Остаточно вибирають той варіант, який забезпечує мінімальне сумарне спотворення для всіх i центроїдів. 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.

Складність алгоритму можна значно зменшити, якщо запуск алгоритму k-середніх здійснювати не для кожного вхідного вектора $\mathbf{x}_i \in \mathbf{S}$, а для визначеної множини векторів $\mathbf{X} = \{\mathbf{x}_0\} \subset \mathbf{S}$, що будуть використовуватися як кандидати на позицію вставки нового центроїда. Вибір множини $\{\mathbf{x}_0\}$ можна виконувати, наприклад, за схемою, що застосовується в алгоритмі k-means++. Зрозуміло, що потужність множини кандидатів у центроїди $\{\mathbf{x}_0\}$ буде суттєво впливати на сумарне спотворення та час роботи алгоритму. Власне, визначення такої кількості кандидатів, яка забезпечувала б мінімальне спотворення без значного уповільнення роботи алгоритму і буде предметом дослідження в цій статті.

У нашому випадку отримання k центроїдів відбувається шляхом послідовного запуску k-середніх для $1, 2, ..., k$ центроїдів, тому, обравши кандидата на вставку наступного центроїда, операцію перерахунку нового положення центроїда можна виконувати (вибір позиції i-го центроїда з перерахунком нового положення центроїдів), а можна не виконувати (вибір позиції i-го центроїда без перерахунку положення центроїдів). Це зумовлено тим, що в результаті перерахунку координати центроїдів змінюються суттєво лише спочатку, коли кількість визначених центроїдів є невеликою, у подальшому ж, зі збільшенням кількості визначених центроїдів, перерозподіл точок між центроїдами майже не відбувається, тобто перерахунок вже не має такого важливого значення, оскільки координати центроїдів змінюються несуттєво.

В [10] 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. Це дозволяє досягти значного зменшення часу роботи алгоритму. Отримуваний результат при цьому близький до глобального оптимуму.

Сумарне спотворення для k центроїдів обчислюють за формулою:

$$E_k{}^2 = \sum_{i=1}^{k} e_i^2 = \sum_{i=1}^{k}\sum_{j=1}^{N_i} D^2(\mathbf{x}_j, \mathbf{c}_i) = \sum_{i=1}^{k}\sum_{j=1}^{N_i}\sum_{m=1}^{d} (x_{jm} - c_{im})^2 . \quad (2)$$

Обчислення сумарного спотворення згідно з формулою (2) вимагає виконання такої кількості операцій: $N_{op} = 2 \cdot d \cdot N_i \cdot k$, де N_i – кількість точок, що належать центроїду $\mathbf{c}_i$.

Вираз для спотворення e_i центроїда $\mathbf{c}_i = (c_1, c_2, ..., c_d)$ можна звести до вигляду:

$$e_i^2 = \sum_{j=1}^{N_i}\sum_{m=1}^{d}(x_{jm} - c_{im})^2 = \sum_{j=1}^{N_i}\sum_{m=1}^{d}(x_{jm} - c_{im})^2 = \sum_{j=1}^{N_i}\sum_{m=1}^{d}(x_{jm}^2 - 2\cdot x_{jm}\cdot c_{im} + c_{im}^2) =$$
$$= \sum_{m=1}^{d}(\sum_{j=1}^{N_i} x_{jm}^2 - 2\cdot c_{im}\cdot\sum_{j=1}^{N_i} x_{jm} + \sum_{j=1}^{N_i} c_{im}^2) = \qquad (3)$$
$$= \sum_{m=1}^{d}(\sum_{j=1}^{N_i} x_{jm}^2 - \frac{2}{N_i}\cdot(\sum_{j=1}^{N_i} x_{jm})^2 + \frac{N_i}{N_i^2}\cdot(\sum_{j=1}^{N_i} x_{jm})^2) = \sum_{m=1}^{d}(\sum_{j=1}^{N_i} x_{jm}^2 - \frac{1}{N_i}\cdot(\sum_{j=1}^{N_i} x_{jm})^2),$$

Замість зберігання координат центроїда $\mathbf{c}_i$ обчислення спотворення згідно з формулою (3) потребує зберігання кількості та суми координат точок, що йому належать за кожною розмірністю d (тобто $\sum_{j=1}^{N_i} x_{jm}^2$). Зрозуміло, що маючи $\sum_{j=1}^{N_i} x_{jm}^2$, завжди можна отримати координати центроїда.

В загальному випадку для вибору кращого кандидата на $(k+1)$-й центроїд необхідно оцінити сумарне спотворення E_{k+1}^2 для кожного кандидата в центроїди та вибрати варіант, що забезпечує мінімальне значення. Але оскільки E_k^2 є однаковим для всіх кандидатів, то достатньо оцінити різницю Δ_{k+1}^2, отриману в результаті введення цього кандидата як $(k+1)$-го центроїда:

$$\Delta_{k+1}^2 = E_k^2 - E_{k+1}^2 = \sum_{i=1}^{k}(e_i)^2 - \sum_{i=1}^{k+1}(e_i')^2 =$$
$$= \sum_{i=1}^{k}\left[\sum_{m=1}^{d}(\sum_{j=1}^{N_i} x_{jm}^2 - \frac{1}{N_i}\cdot(\sum_{j=1}^{N_i} x_{jm})^2) - \sum_{m=1}^{d}(\sum_{j=1}^{N_i - M_i} x_{jm}^2 - \frac{1}{N_i - M_i}\cdot(\sum_{j=1}^{N_i} x_{jm})^2)\right] -$$
$$- \sum_{m=1}^{d}\left(\sum_{i=1}^{k}(\sum_{j=1}^{M_i} x_{jm}^2 - \frac{1}{\sum_{i=1}^{k} M_i}\cdot\sum_{i=1}^{k}\sum_{j=1}^{M_i}(x_{jm})^2\right) =$$
$$= \sum_{m=1}^{d}\left[\sum_{i=1}^{k}\left(\sum_{j=1}^{N_i} x_{jm}^2 - (\sum_{j=1}^{N_i} x_{jm}^2 - \sum_{j=1}^{M_i} x_{jm}^2) + \sum_{j=1}^{M_i} x_{jm}^2\right) - \right.$$
$$\left. - \sum_{i=1}^{k}\frac{1}{N_i}\cdot(\sum_{j=1}^{N_i} x_{jm})^2 + \sum_{i=1}^{k}\frac{1}{N_i - M_i}\cdot(\sum_{j=1}^{N_i - M_i} x_{jm})^2 + \frac{1}{\sum_{i=1}^{k} M_i}\cdot\sum_{i=1}^{k}(\sum_{j=1}^{M_i} x_{jm})^2\right] = \qquad (4)$$
$$= \sum_{m=1}^{d}\left\{\sum_{i=1}^{k}\left[\frac{1}{N_i - M_i}\cdot\left(\sum_{j=1}^{N_i} x_{jm} - \sum_{j=1}^{M_i} x_{jm}\right) - \frac{1}{N_i}\cdot(\sum_{j=1}^{M_i} x_{jm})^2\right] + \frac{1}{N_{k+1}}\cdot\sum_{i=1}^{k}(\sum_{j=1}^{M_i} x_{jm})^2\right\},$$

де M_i – кількість точок $\mathbf{c}_i$, що відійдуть центроїду $\mathbf{c}_{k+1}$.

Зауважимо, що в багатьох випадках (особливо зі збільшенням числа визначених центроїдів), кількість точок $\mathbf{c}_i$, що відійдуть центроїду $\mathbf{c}_{k+1}$ є невеликою, тобто $M_i << N_i$.

Отже, оскільки сума координат точок $\sum_{j=1}^{N_i} x_{jm}^2$, що належить центроїду $\mathbf{c}_i$, зберігається, а $\sum_{j=1}^{M_i} x_{jm}^2$ можна також отримати в процесі визначення належності точок центроїду $\mathbf{c}_{k+1}$,

отримуваний виграш згідно з (4) можна обчислити за відносно невеликий час. У цьому випадку вираз кількості операцій має вигляд: $N_{op} = (M_i + 5) \cdot d \cdot k$.

Метод k-середніх з обчисленням відстаней до активних центроїдів

Насамперед зазначимо, що найтрудомісткішим складником обчислень є знаходження центроїда, найближчого до цього вектора, оскільки це потребує обчислення відстаней від кожного вектора до кожного центроїда.

Із виконанням кожної наступної ітерації t все менше центроїдів змінюють свої положення (активні центроїди $\mathbf{SC}_t^{(a)}$), а більшість з них залишається на своїх позиціях (пасивні центроїди $\mathbf{SC}_t^{(p)}$). Отже, 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:

$$r_t = \frac{\left|\mathbf{SC}_t^{(a)}\right|}{\left|\mathbf{SC}_t\right|}, \tag{5}$$

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

На рис. 1 показано, як змінюється частка активних центроїдів r із зростанням загальної кількості центроїдів $|\mathbf{SC}|$, отримана в результаті кластеризації 75000 векторів розмірністю $d = 5$. Дані по обох осях наведено в логарифмічному масштабі. Для згладжування кривої, представленої на рисунку, дані усереднювалися за 1000 ітерацій.

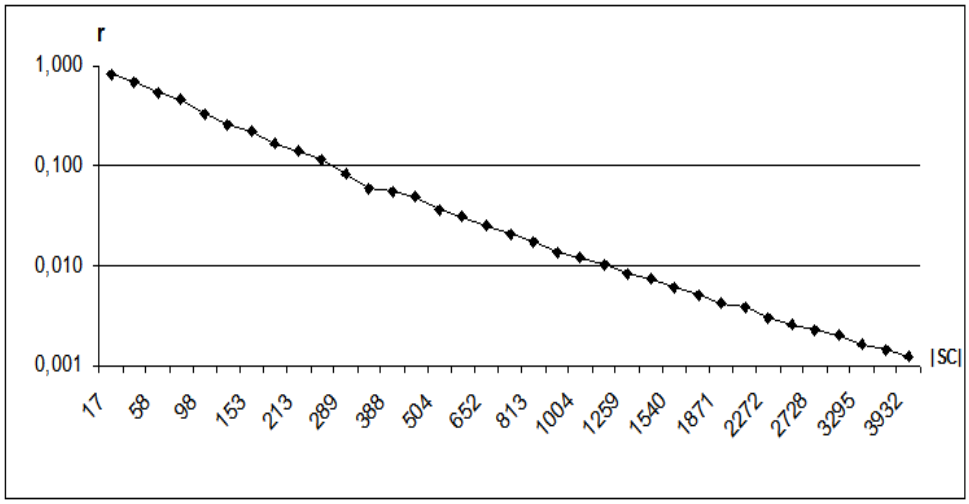

Рис. 1. Залежність частки активних центроїдів від загальної кількості центроїдів

Як можна побачити, із зростанням $|\mathbf{SC}|$ 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. Зазначимо, що центри кластерів не будуть відрізнятися від тих, що отримані під час застосування алгоритму глобальної кластеризації.

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.

Метод k-середніх з обчисленням відстаней до активних центроїдів здійснюють таким чином:

1. Визначаємо множину точок $\{\mathbf{x}_0\} \subset \mathbf{S}$, що будуть використовуватися для визначення початкової позиції 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 $\mathbf{c}_i \in \mathbf{W}$, коригуємо відповідне значення відстані r_i. Якщо $\mathbf{c}_i \notin \mathbf{W}$, перевіряємо виконання умови $r_i < r_{\max}$, де $r_{\max} = \max_j D(\mathbf{x}, \mathbf{c}_j)$ $j = 1, 2, ..., m$. 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.

5. 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 та 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, 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 рекомендованим для практичного застосування.

Застосування kd-дерев для визначення кандидатів на вибір позиції вставки нового центроїда

Ще один способом зменшення обчислювальної складності є використання багатовимірних бінарних дерев пошуку (kd-дерев) [11, 12]. kd-дерево будують на основі вхідних векторів, яким у процесі спуску по дереву призначають найближчі центроїди. Зменшення кількості обчислень під час цього можна досягти за рахунок того, що до вузла дерева може належати багато вхідних векторів. Отже, можна виконувати обчислення не для конкретних точок, а для усіх одразу, що належить певному вузлу дерева. Оскільки kd-дерево обчислюють для вхідних векторів, а не для центроїдів, немає необхідності оновлювати цю структуру даних, тобто її побудова виконують один раз. Це дозволяє зберігати в структурі дерева інформацію, що обчислюється в процесі побудови дерева і може бути корисною в подальшому, зокрема: кількість векторів, що асоціюються з вузлом дерева, та сума координат векторів у вузлі.

В цій роботі kd-дерева застосовують для визначення кандидата на початкову позицію центроїда. Для кожного вузла дерева підтримується список кандидатів у центроїди, кількість

яких у процесі спуску по дереву буде зменшуватися за рахунок виключення тих, що не можуть бути ближчими для жодної з точок цього вузла. Якщо вузол, який досягнули, є термінальним (кількість векторів у якому не перевищує заданої), то виконують обчислення відстаней до усіх кандидатів для кожної з точок, що асоціюються з цим вузлом дерева, і кожну точку співвідносять з певним центроїдом. В такому випадку продуктивність пошуку визначають кількістю кандидатів у списку для цього вузла. У випадку, коли з вузлом асоціюється лише один кандидат, його можна вважати найближчим центроїдом для усіх точок вузла. Як бачимо, в обох випадках спостерігається значне зменшення кількості обчислень відстаней. Отже, застосування kd-дерев дозволяє досягти подальшого прискорення операції пошуку центроїда, найближчого до цього вектора, що є основним складником обчислень, який виконують на кожній ітерації k-середніх.

Експериментальні результати

Дослідження проводили на наборі векторів LSF-параметрів [13], отриманих з мовленнєвої бази даних TIMIT, у кількості n=50000 для розмірностей d=5 та d=10. Усі обчислення виконували на комп'ютері Intel Core 2 2.0GHz з 2Гб пам'яті.

У цій роботі досліджено вплив кількості векторів-кандидатів у центроїди на якість кластеризації та швидкодію алгоритму. Розглянуто два варіанти пошуку належної позиції i-го центроїда: на основі вибору вектора-кандидата, який забезпечує мінімальне спотворення з перерахунком нового положення центроїдів (RC), і без перерахунку центроїдів (WRC). В обох випадках (RC, WRC) ефективність оцінювалась за сумарним спотворенням (Error) та часом роботи алгоритму в секундах (Time).

В таблиці 1 та на рис. 2 (1000 центроїдів) і 3 (4000 центроїдів) показано залежність спотворення та часу роботи від кількості кандидатів у центроїди для обох варіантів вибору i-го центроїда (RC та WRC).

Таблиця 1

Сумарне спотворення для різної кількості кандидатів: n=50000

К-сть центроїдів, k	Розмірність	Вибір i-го центроїда	Кількість векторів-кандидатів у центроїди, |C|					
			1000	2000	4000	8000	16000	32000
1000	5	WRC	$2{,}642\cdot10^8$	$2{,}632\cdot10^8$	$2{,}623\cdot10^8$	$2{,}614\cdot10^8$	$2{,}610\cdot10^8$	
		RC	$2{,}628\cdot10^8$	$2{,}619\cdot10^8$	$2{,}611\cdot10^8$	$2{,}604\cdot10^8$	$2{,}598\cdot10^8$	
	10	WRC	$1{,}928\cdot10^9$	$1{,}919\cdot10^9$	$1{,}914\cdot10^9$	$1{,}910\cdot10^9$	$1{,}909\cdot10^9$	
		RC	$1{,}925\cdot10^9$	$1{,}915\cdot10^9$	$1{,}909\cdot10^9$	$1{,}905\cdot10^9$	$1{,}902\cdot10^9$	
4000	5	WRC			$1{,}279\cdot10^8$	$1{,}252\cdot10^8$	$1{,}238\cdot10^8$	$1{,}232\cdot10^8$
		RC			$1{,}262\cdot10^8$	$1{,}235\cdot10^8$	$1{,}220\cdot10^8$	$1{,}213\cdot10^8$
	10	WRC			$1{,}205\cdot10^9$	$1{,}185\cdot10^9$	$1{,}173\cdot10^9$	$1{,}164\cdot10^9$
		RC			$1{,}188\cdot10^9$	$1{,}167\cdot10^9$	$1{,}155\cdot10^9$	$1{,}146\cdot10^9$

З таблиці 1 видно, що збільшення числа кандидатів на вибір місця початкового розташування нового центроїду дозволяє досягти зменшення спотворення. При цьому має місце експоненціальний характер залежності часу від кількості кандидатів, що показано на рис. 2 та 3. Зі збільшенням числа кандидатів більш стрімке зростання часу спостерігається для RC. Крім того, при використанні більш грубої оцінки положення центроїдів за WRC, спостерігається незначне збільшення спотворення (близько 1%), тоді як час зменшується на 5 – 20% (залежно від кількості кандидатів). Зі зростанням розмірності характер залежності зберігається, абсолютні ж значення спотворення зростають на порядок.

Отже, можна зробити висновки, що збільшення кількості кандидатів дозволяє досягти лише незначного зменшення сумарного спотворення за рахунок зростання часу вдвічі.Тому більшої ефективності можна досягти, використовуючи RC, з кількістю кандидатів $|C|=4\cdot k$ (тобто 4000 кандидатів для генерації 1000 центроїдів та 16000 кандидатів для отримання 4000 центроїдів), оскільки це дозволяє досягнути меншого спотворення без значного уповільнення роботи алгоритму порівняно з WRC, де приблизно таке ж значення спотворення досягається при $|C|=8\cdot k$.

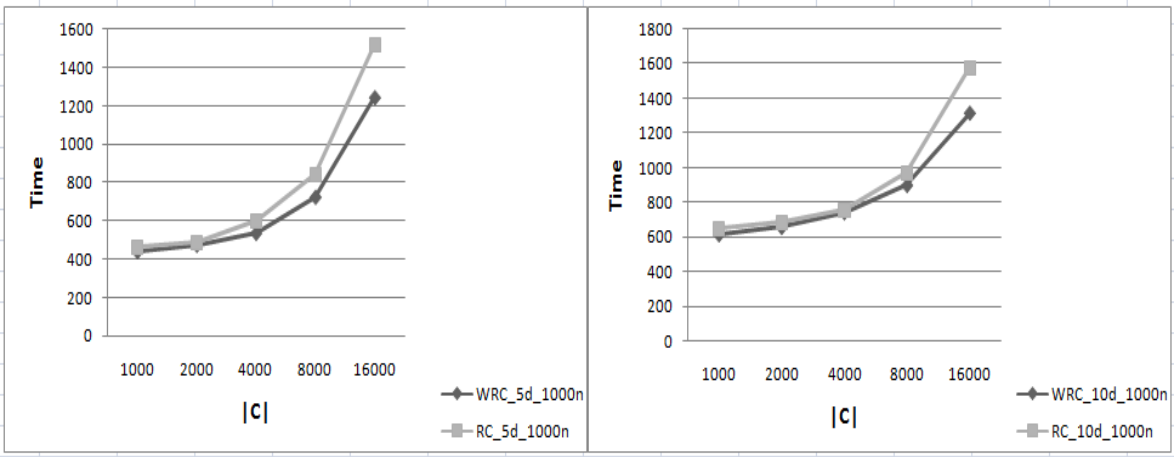

Рис. 2. Залежність часу роботи від кількості вхідних векторів

а) $d=5, n=1000$ б) $d=10, n=1000$

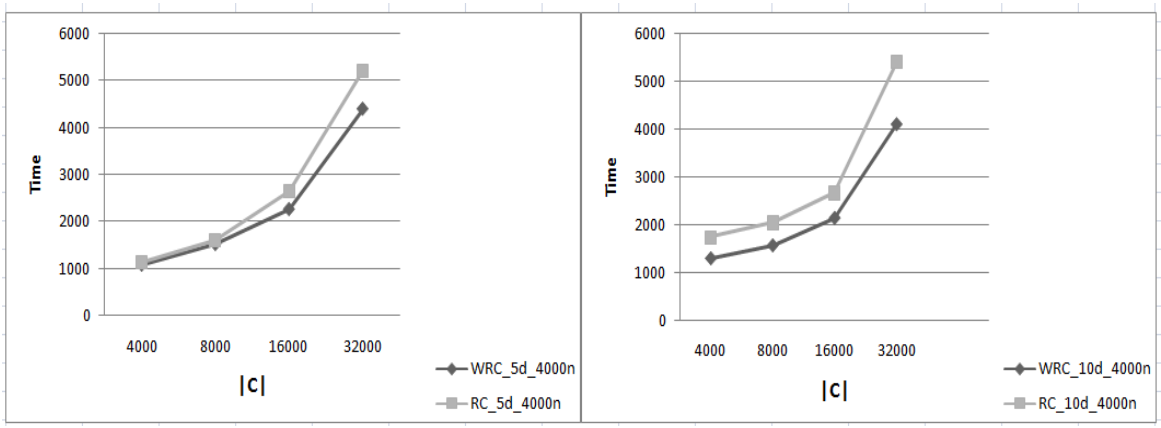

Рис. 3. Залежність часу роботи від кількості вхідних векторів

а) $d=5, n=4000$ б) $d=10, n=4000$

Запропонований в цій роботі алгоритм кластеризації (RC при $|\mathbf{C}|=16000$) також порівнювали з двома модифікаціями алгоритму k-середніх (що показано на рис. 4): 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 в [12]. Специфіка алгоритму hybrid полягає в тому, що для наближення до глобального мінімуму в ньому поєднано класичний алгоритм k-середніх та локальний пошук (обмін між існуючими центроїдами та кандидатами на центроїди за умови, що такий обмін призводить до зменшення середнього спотворення).

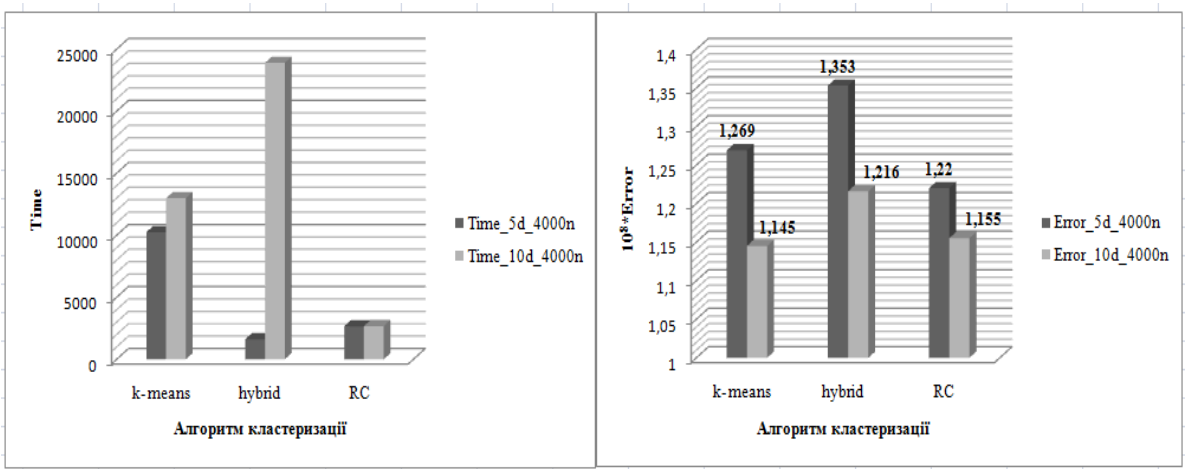

Рис. 4. Порівняння алгоритмів за сумарним спотворенням (зліва) та часом роботи (справа): $n = 4000$

Результати, наведені зліва на рис. 4, показують, що для обох розмірностей найбільше значення сумарного спотворення спостерігається під час застосування алгоритму кластеризації на основі kd-дерев, найменше ж спотворення дозволяють отримати RC для $d = 5$ та k-means (MATLAB) для $d = 10$.

Справа на рис. 4 ефективність алгоритмів оцінюють за часом роботи. Так, для $d = 5$ найбільша швидкодія досягається алгоритмом з використанням kd-дерев, проте зі збільшенням розмірності для hybrid спостерігається стрімке зростання часу роботи, що зумовлено експоненціальним характером залежності часу від розмірності. Зазначимо, що ефективність запропонованого в роботі алгоритму практично не залежить від розмірності.

Висновки

Запропонований у роботі вдосконалений метод кластеризації k-середніх дозволяє отримати розв'язок, наближений до глобального мінімуму спотворення. Зменшення сумарного спотворення складає до 11% порівняно з алгоритмом hybrid та до 5% під час використання алгоритму k-means (MATLAB). За швидкістю роботи запропонований метод дає результати кращі в 4 – 8 разів, ніж алгоритм k-means (MATLAB) для розмірностей $d = 5$ і $d = 10$ відповідно, та в 9 разів, ніж hybrid для $d = 10$, у випадку, коли $d = 5$, він дещо програє hybrid у швидкодії (у 1,6 разів). Зазначимо, що ефективність запропонованого в роботі методу практично не залежить від розмірності. Зменшення похибки кластеризації досягається за рахунок покращеної процедури вибору векторів-кандидатів на вставку нового центроїда (RC) та визначення оптимальної кількості кандидатів, яка становить $|C| = 4 \cdot k$ і призводить до уповільнення роботи алгоритму в 1,5 – 2 рази.

СПИСОК ЛІТЕРАТУРИ

1. Fayyad U. M. Advances in Knowledge Discovery and Data Mining / U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy // AAAI/MIT Press. – 1996. – 611 p.

2. Gersho A. Vector Quantization and Signal Compression. / A. Gersho, R. M. Gray // Boston: Kluwer Academic. – 1992. – 760 p.

3. Duda R. O. Pattern Classification and Scene Analysis / R. O. Duda, P. E. Hart // New York: John Wiley & Sons. – 1973. – 512 p.

4. Jain A. K. Algorithms for Clustering Data / A. K. Jain, R. C. Dubes // Englewood Cliffs, N.J.: Prentice Hall. – 1988. – 334 p.

5. 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. 1027 – 1035.

6. 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. – No 11, vol. 42. – P. 2551 – 2556.

7. Kanungo T. An Efficient k-means clustering algorithm: analysis and implementation / T. Kanungo, D. M.

Mount, N. S. Netanyahu, C. Piatko, R. Silverman and A.Y. Wu // IEEE Transactions On Pattern Analysis And Machine Intelligence. – 2002. – No. 7, vol. 24. – P. 881 – 892.

8. Likas A. The global k-means clustering algorithm / Aristidis Likas, Nikos Vlassis, Jacob J. Verbeek // Pattern Recognition. – 2002. – No 2, vol. 36. – P. 451 – 461.

9. Refining initial points for KMeans clustering : (Conference on Machine Learning) [Електронний ресурс] / P. S. Bradley, U. M. Fayyad // Proceedings of Fifteenth Intl. – 1998. – P. 91 – 99. / Режим доступу: ftp://ftp.research.microsoft.com/pub/tr/tr-98-36.pdf.

10. Hussein N. A Fast Greedy K-Means Algorithm / Master's Thesis Nr:9668098 N. Hussein. – University of Amsterdam Faculty of Mathematics, Computer Sciences, Physics and Astronomy Euclides Building Plantage muidergracht 24. – 2002. – p. 62.

11. Moore Andrew William Efficient memory based learning for robot control / Moore Andrew William. – PhD thesis Nr: UCAM-CL-TR-209. – 1990. – p. 248.

12. Kanungo T. A Local Search Approximation Algorithm for k-Means Clustering / T. Kanungo, D. M. Mount, N. S. Netanyahu, C. Piatko, R. Silverman, A. Y. Wu // Computational Geometry: Theory and Applications. – 2004. – No 2. – P. 89 – 112.

13. 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.

Ткаченко Олександр Миколайович – к. т. н., доцент кафедри обчислювальної техніки, тел. 59-84-13, e-mail: ant@vstu.vinnica.ua.

Грійо Тукало Оксана Франсисківна – студентка інституту інформаційних технологій та комп'ютерної інженерії, тел. 0662819319, e-mail: xxmargoxx@gmail.com.

Дзісь Олексій Вікторович – студент інституту інформаційних технологій та комп'ютерної інженерії,
e-mail: alexdz47@gmail.com.

Лаховець Сергій Максимович – студент інституту інформаційних технологій та комп'ютерної інженерії, e-mail: selema.tsx@gmail.com

Вінницький національний технічний університет.