УДК 621.39

А. Н. Ткаченко, к. т. н., доц.; О. Ф. Грийо Тукало; О. В. Дзись; С. М. Лаховец

МЕТОД КЛАСТЕРИЗАЦИИ НА ОСНОВЕ ПОСЛЕДОВАТЕЛЬНОГО ЗАПУСКА K-СРЕДНИХ С УСОВЕРШЕНСТВОВАННЫМ ВЫБОРОМ КАНДИДАТА НА НОВУЮ ПОЗИЦИЮ ВСТАВКИ

В работе предложен усовершенствованный метод кластеризации k-средних, который, в отличие от классического, позволяет получить решение, приближенное к глобальному минимуму искажения путем последовательного запуска k-средних для центроидов $1, 2, \ldots, k$. *Уменьшение искажения достигается за счет улучшенной процедуры определения векторов-кандидатов на выбор позиции вставки нового центроида без значительного замедления времени работы.*

Ключевые слова: *кодовые книги, кластеризация, метод k-средних, центроиды, kd-деревья.*

Введение

Кластеризация – это разбивка определенного множества объектов на непересекающиеся подмножества (кластеры), таким образом, чтобы каждый кластер содержал похожие объекты, а объекты разных кластеров отличались между собой. Кластеризация часто используется, в частности, при статистическом анализе данных [1], векторной квантизации [2], распознавании образов [3] и т. п. В области сжатия речи алгоритмы кластеризации применяются для создания кодовых книг – специальных таблиц, содержащих наиболее репрезентативные наборы данных. Задача кластеризации данных является важным элементом общей проблемы обработки данных, для решения которой существует множество подходов и алгоритмов: от сугубо интуитивных и эвристических до строго математических. Задачу кластеризации можно сформулировать так: заданный набор из n векторов, каждый из которых имеет размерность d, необходимо разбить на подмножества в соответствии с заданным критерием оптимизации. Как правило, таким критерием является минимизация искажения. Существуют различные пути оценивания искажения, но в большинстве прикладных реализаций используют сумму среднеквадратических Евклидовых расстояний между вектором и центром кластера (центроида), к которому он принадлежит [2, 4].

Метод кластеризации k-средних является наиболее распространенным и наиболее исследованным среди всех методов кластеризации. Он минимизирует вышеупомянутое искажение, распределяя данные между регионами, которые не пересекаются и идентифицируются по их центрам. Распространенность метода k-средних обусловлена его основными преимуществами: простотой, гибкостью, быстрой сходимостью. Однако практическое применение метода существенно ограничивается его недостатками: результаты кластеризации по методу k-средних в значительной степени зависят от выбора начальной конфигурации центроидов (инициализации), работа алгоритма существенно замедляется при кластеризации больших объемов данных; алгоритм может сходиться к локальному минимуму целевой функции.

Чтобы избежать этих недостатков был предложен ряд модификаций метода k-средних. В методе k-средних++ (k-means++) введено усовершенствованную процедуру инициализации, что позволяет улучшить результаты кластеризации за счет специального выбора начальной конфигурации центроидов [5]. Для ускорения процесса вычисления расстояний от точек до центроидов в [6] предложено исключать из рассмотрения статические центроиды, то есть оставшиеся на своих позициях на текущей итерации. В [7] с целью уменьшения вычислительной сложности метода k-средних использована особая структура данных – kd-деревья, что позволило существенно уменьшить вычислительную сложность метода. С

целью предотвращения локальной сходимости в [8] предложен итеративный алгоритм, позволяющий приблизиться к глобальному оптимуму путем пошагового последовательного запуска k-средних.

Постановка задачи

В данной работе сочетаются преимущества рассмотренных подходов с целью усовершенствования метода кластеризации k-средних, в котором решение, приближенное к глобальному минимуму, получают путем последовательного запуска k-средних для 1, 2, ..., k центроидов, и уменьшение искажения достигается за счет улучшенной процедуры определения векторов-кандидатов на выбор позиции вставки нового центроида.

Классический алгоритм k-средних

Кластеризация методом k-средних распределяет входной набор векторов по k кластерам $\mathbf{S}_i (i = 1, 2, ..., k)$, с каждым из которых связан центроид $\mathbf{c}_i$. Обозначим множество входных векторов $\mathbf{S} = \{\mathbf{x}\}, |\mathbf{S}| = n$. Пусть $D(\mathbf{x}, \mathbf{c})$ – расстояние между вектором $\mathbf{x}$ и центроидом $\mathbf{c}$. В этой статье используется невзвешенное Евклидовое расстояние между вектором $\mathbf{x} = (x_1, x_2, ..., x_d)$ и центроидом $\mathbf{c} = (c_1, c_2, ..., c_d)$:

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

Обозначим множество центроидов, полученных на итерации t, $\mathbf{SC}_t = \{\mathbf{c}_i\}$. Алгоритм кластеризации k-средних в его обычном варианте описывают так:

1. Устанавливаем $t = 0$ и задаем начальное расположение центроидов $\mathbf{SC}_0$.

2. Для заданного множества центроидов $\mathbf{SC}_t$ выполняем действия, описанные в пунктах 2.1 и 2.2 и получаем улучшенное множество центроидов $\mathbf{SC}_{t+1}$:

2.1 Находим такое разбиение $\mathbf{S}$, которое распределяет $\mathbf{S}$ по k кластерам $\mathbf{S}_i (i = 1, 2, ..., k)$ и удовлетворяет условие

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

2.2 Вычисляем центроид $\mathbf{c}_i$ для каждого кластера $\mathbf{S}_i (i = 1, 2, ..., k)$, чтобы получить новое множество центроидов $\mathbf{SC}_{t+1}$:

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

где m_i – количество векторов, принадлежащих кластеру $\mathbf{S}_i$.

3. Вычисляем суммарное искажение $E^2 = \sum_{\mathbf{x} \in S} D^2(\mathbf{x}, \mathbf{c})$ для $\mathbf{SC}_{t+1}$. Если оно отличается от полученного на предыдущей итерации на достаточно малую величину, прекращаем процесс. В противном случае присваиваем $t \leftarrow t + 1$ и возвращаемся к шагу 2.

Алгоритм гарантированно сходится за конечное число итераций. Погрешность кластеризации и число итераций зависит от первоначального выбора центроидов, поэтому обычной практикой является запуск k-средних несколько раз с различными начальными кандидатами в центроиды [9].

Глобальный алгоритм k-средних

Как отмечалось выше, алгоритму кластеризации k-средних присущи определенные недостатки. С целью их преодоления в [10] предложен усовершенствованный вариант

кластеризации k-средних, названный авторами жадным глобальным алгоритмом k-средних (greedy global k-means algorithm). В его основе лежит предположение, что глобальный оптимум может быть достигнут путем запуска k-средних, когда $(k-1)$ центроид находится в оптимальных позициях, полученных как решение задачи кластеризации для $(k-1)$ центроида, а k-й центроид должен быть размещен в соответствующей позиции, которую и надо определить. Оптимальную кластеризацию для $k=1$ легко получить, вычислив координаты первого центроида как среднеарифметическое соответствующих координат всех векторов множества $\mathbf{S}$. Таким образом, для реализации данного подхода получения k центроидов необходим последовательный запуск k-средних для 1, 2, ..., k центроидов.

Глобальный алгоритм k-средних предусматривает, что поиск надлежащей позиции i-го центроида, которая неизвестна, когда известны позиции предыдущих $(i-1)$ центроидов, требует запуска k-средних для каждого вектора $\mathbf{x}_i \in \mathbf{S}$ из набора входных векторов, который рассматривают как кандидата на позицию вставки нового центроида. Окончательно выбирают тот вариант, который обеспечивает минимальное суммарное искажение для всех i центроидов. Это означает, что для практических применений, где количество векторов составляет несколько десятков или сотен тысяч, а число кластеров – несколько тысяч, время работы алгоритма является слишком длительным.

Сложность алгоритма можно значительно уменьшить, если запуск алгоритма k-средних осуществлять не для каждого входного вектора $\mathbf{x}_i \in \mathbf{S}$, а для определенного множества векторов $\mathbf{X} = \{\mathbf{x}_0\} \subset \mathbf{S}$, которые будут использоваться в качестве кандидатов на позицию вставки нового центроида. Выбор множества $\{\mathbf{x}_0\}$ может выполняться, например, по схеме, которая применяется в алгоритме k-means+ +. Понятно, что мощность множества кандидатов в центроиды $\{\mathbf{x}_0\}$ будет существенно влиять на суммарное искажение и время работы алгоритма. Собственно, определение такого количества кандидатов, которое обеспечивало бы минимальное искажение без значительного замедления работы алгоритма и будет предметом исследования в данной статье.

В нашем случае получение k центроидов происходит путем последовательного запуска k-средних для $1, 2, ..., k$ центроидов, поэтому, выбрав кандидата на вставку следующего центроида, операцию пересчета нового положения центроида можно выполнять (выбор позиции i-го центроида с пересчетом нового положения центроидов), а можно не выполнять (выбор позиции i-го центроида без пересчета положения центроидов). Это обусловлено тем, что в результате пересчета координаты центроидов изменяются существенно лишь поначалу, когда количество определенных центроидов небольшое, в дальнейшем же, с увеличением количества определенных центроидов, перераспределения точек между центроидами почти не происходит, то есть пересчет уже не имеет такого значения, поскольку координаты центроидов изменяются незначительно.

В [10] предлагается для поиска надлежащей позиции i-го центроида ограничиться выбором того вектора, который обеспечивает минимальное искажение при добавлении его в качестве исходной позиции нового центроида вместо запуска k-средних для каждого вектора. Это позволяет достичь значительного уменьшения времени работы алгоритма. Полученный результат при этом близок к глобальному оптимуму.

Суммарное искажение для k центроидов вычисляют по формуле:

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

Вычисление суммарного искажения согласно формуле (2) требует выполнения такого количества операций: $N_{op} = 2 \cdot d \cdot N_i \cdot k$, где N_i – количество точек, принадлежащих

центроиду $\mathbf{c}_i$.

Выражение для искажения e_i центроида $\mathbf{c}_i = (c_1, c_2, ..., c_d)$ можно свести к виду:

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

Вместо хранения координат центроида $\mathbf{c}_i$ вычисление искажения согласно формуле (3) требует хранения количества и суммы координат точек, принадлежащих ему по каждой размерности d (то есть $\sum_{j=1}^{N_i} x_{jm}^{\,2}$). Понятно, что имея $\sum_{j=1}^{N_i} x_{jm}^{\,2}$, всегда можно получить координаты центроида.

В общем случае для выбора лучшего кандидата на $(k+1)$-й центроид необходимо оценить суммарное искажение $E_{k+1}^{\,2}$ для каждого кандидата в центроиды и выбрать вариант, обеспечивающий минимальное значение. Но поскольку $E_k^{\,2}$ является одинаковым для всех кандидатов, то достаточно оценить разницу Δ_{k+1}^2, полученную в результате введения данного кандидата как $(k+1)$-го центроида:

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

где M_i – количество точек $\mathbf{c}_i$, что отойдут центроиду $\mathbf{c}_{k+1}$.

Заметим, что во многих случаях (особенно с увеличением числа определенных центроидов) количество точек $\mathbf{c}_i$, которое отойдут центроиду $\mathbf{c}_{k+1}$, небольшое, то есть $M_i << N_i$.

Итак, поскольку сумма координат точек $\sum_{j=1}^{N_i} x_{jm}{}^2$, принадлежащих центроиду $\mathbf{c}_i$, сохраняется, а $\sum_{j=1}^{M_i} x_{jm}{}^2$ можно получить в процессе определения принадлежности точек центроиду $\mathbf{c}_{k+1}$, получаемый выигрыш согласно (4) можно вычислить за относительно небольшое время. В этом случае выражение количества операций имеет вид: $N_{op} = (M_i + 5) \cdot d \cdot k$.

Метод k-средних с вычислением расстояний до активных центроидов

Прежде всего отметим, что наиболее трудоемкой частью вычислений является нахождение центроида, ближайшего к данному вектору, поскольку это требует вычисления расстояний от каждого вектора до каждого центроида.

С выполнением каждой следующей итерации t все меньше центроидов меняют свое положение (активные центроиды $\mathbf{SC}_t^{(a)}$), а большинство из них остается на своих позициях (пассивные центроиды $\mathbf{SC}_t^{(p)}$). Таким образом, если хранить для каждой точки расстояния до всех центроидов на итерации t, на итерации $t+1$ достаточно вычислить расстояния только к активным центроидам. При этом выигрыш во времени будет тем больше,чем меньше относительная доля r_t активных центроидов среди всех центроидов на итерации t:

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

где $\left|\mathbf{SC}_t^{(a)}\right|$ и $\left|\mathbf{SC}_t\right|$ – мощности множеств $\mathbf{SC}_t^{(a)}$ и $\mathbf{SC}_t$ соответственно.

На рис. 1 показано, как изменяется доля активных центроидов r с ростом общего количества центроидов $|\mathbf{SC}|$, полученная в результате кластеризации 75000 векторов размерностью $d = 5$. Данные по обеим осям приведены в логарифмическом масштабе. Для сглаживания кривой, представленной на рисунке, данные усреднялись за 1000 итераций.

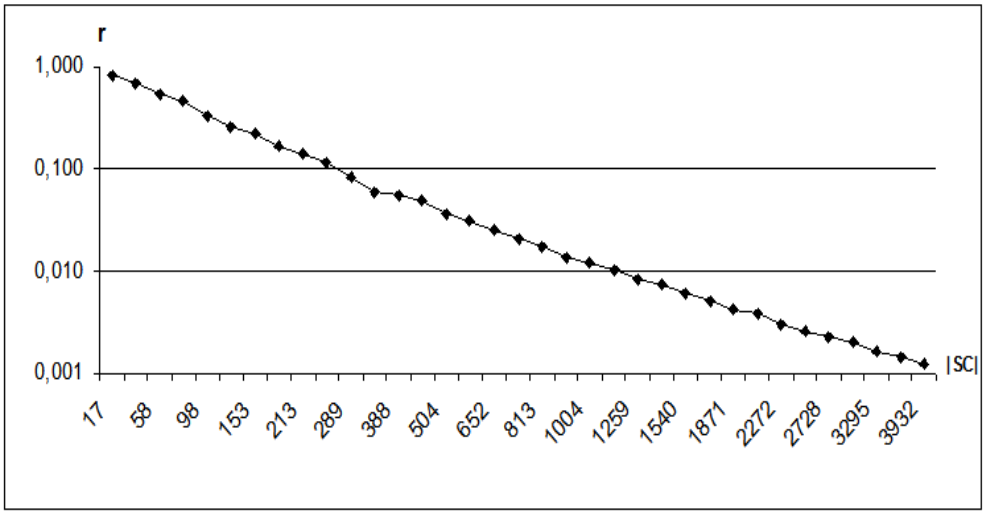

Рис. 1. Зависимость доли активных центроидов от общего количества центроидов

Как можно увидеть, по мере роста $|\mathbf{SC}|$ от 2 до 40000, r постепенно уменьшается от 0,9 до 0,0012. Средняя доля активных центроидов r_{av} составляет примерно 0,015. Таким

образом, количество вычислений расстояний должно уменьшиться в 50 – 100 раз по сравнению с глобальным алгоритмом k-средних. Отметим, что центры кластеров не будут отличаться от полученных при применении алгоритма глобальной кластеризации.

Указанный выигрыш в быстродействии достигается за счет существенного увеличения затрат памяти, поскольку для каждого вектора необходимо хранить расстояния до всех центроидов. Однако эти расходы можно значительно сократить, если для каждого вектора хранить расстояния не до всех, а только до m ближайших центроидов.

Метод k-средних с вычислением расстояний до активных центроидов осуществляется следующим образом:

1. Определяем множество точек $\{\mathbf{x}_0\} \subset \mathbf{S}$, которые будут использоваться для определения начальной позиции вставки нового центроида.

2. Присвоив $k = 1$ вычисляем координаты первого центроида как среднее значение координат всех векторов:

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

3. Выполняем $k \leftarrow k+1$ и находим исходную позицию для вставки центроида $\mathbf{c}_k$ путем выбора вектора $\mathbf{x} \in \{\mathbf{x}_0\}$, что обеспечивает минимальное искажение $E^2 = \sum_{\mathbf{x} \in S} D^2(\mathbf{x}, \mathbf{c})$.

4. Запускаем алгоритм k-средних для k центроидов, для чего выполняем шаги 4.1 – 4.3:

4.1 Разделяем множество $\mathbf{SC}$ на подмножества $\mathbf{SC}^{(a)}$ и $\mathbf{SC}^{(p)}$, состоящие соответственно из активных и пассивных центроидов.

4.2 Для каждого вектора $\mathbf{x} \in S$ определяем $\mathbf{W} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_m\}$, содержащее множество из m ближайших до $\mathbf{x}$ центроидов. Для каждого $\mathbf{c}_i \in \mathbf{SC}^{(a)}$ вычисляем расстояние $r_i = D(\mathbf{x}, \mathbf{c}_i)$. Если $\mathbf{c}_i \in \mathbf{W}$ корректируем соответствующее значение расстояния r_i. Если $\mathbf{c}_i \notin \mathbf{W}$ проверяем выполнение условия $r_i < r_{\max}$, где $r_{\max} = \max_j D(\mathbf{x}, \mathbf{c}_j)$ $j = 1, 2, ..., m$. Если условие выполняется, добавляем $\mathbf{c}_i$ к множеству $\mathbf{W}$.

4.3 Используя (1), вычисляем новое положение центроидов. Если условие сходимости не выполняется, возвращаемся к 4.1.

5. Проверяем, получено ли заданное количество центроидов $k_{\max}$. Если $k < k_{\max}$, возвращаемся к 3.

Приведенный метод требует дополнительных затрат памяти для хранения индексов ближайших центроидов и расстояний до них для каждого вектора. Кроме того, для ускорения проверки $\mathbf{c}_i \in \mathbf{W}$ поиск подходящего элемента можно организовать с помощью хеширования, что также требует дополнительной памяти.

Как видно из рис. 1, значение должно быть больше для малых k и меньше для больших значений k. Однако небольшое число $m = 4$ позволяет избежать увеличения искажения и может быть рекомендовано для практического применения.

Применение kd-деревьев для определения кандидатов на выбор позиции вставки нового центроида

Еще одним способом уменьшения вычислительной сложности является использование многомерных бинарных деревьев поиска (kd-деревьев) [11, 12]. kd-дерево строят на основе входных векторов, которым в процессе спуска по дереву назначают ближайшие центроиды. Уменьшения количества вычислений при этом можно достичь за счет того, что к узлу дерева может принадлежать много входных векторов. Таким образом, можно выполнять вычисления не для конкретных точек, а для всех сразу, которые принадлежат определенному узлу дерева. Поскольку kd-дерево определяют для входных векторов, а не для центроидов,

нет необходимости обновлять эту структуру данных, то есть ее построение выполняют один раз. Это позволяет хранить в структуре дерева информацию, которая вычисляется в процессе построения дерева и может быть полезной в дальнейшем: количество векторов, которые ассоциируются с узлом дерева, и сумма координат векторов в узле.

В данной работе kd-деревья применяют для определения кандидата на исходную позицию центроида. Для каждого узла дерева поддерживается список кандидатов в центроиды, количество которых в процессе спуска по дереву будет уменьшаться за счет исключения тех, которые не могут быть ближе для одной из точек данного узла. Если узел, который достигнут является терминальным (количество векторов в котором не превышает заданного), то выполняются вычисления расстояний до всех кандидатов для каждой из точек, которые ассоциируются с данным узлом дерева, и каждую точку соотносят с определенным центроидом. В этом случае производительность поиска определяется количеством кандидатов в списке для данного узла. В случае, когда с узлом ассоциируется только один кандидат, его можно считать ближайшим центроидом для всех точек узла. Как видим, в обоих случаях наблюдается значительное уменьшение количества вычислений расстояний. Следовательно, применение kd-деревьев позволяет достичь дальнейшего ускорения операции поиска центроида, ближайшего к данному вектору, что является основной составляющей вычислений, которая выполняется на каждой итерации k-средних.

Экспериментальные результаты

Исследования проводились на наборе векторов LSF-параметров [13], полученных с речевой базы данных TIMIT, в количестве $n = 50000$ для размерностей $d = 5$ и $d = 10$. Все вычисления выполнялись на компьютере Intel Core 2 2.0GHz с 2Гб памяти.

В данной работе исследовалось влияние количества векторов-кандидатов в центроиды на качество кластеризации и быстродействие алгоритма. Рассматривались два варианта поиска надлежащей позиции i-го центроида: на основе выбора вектора-кандидата, который обеспечивает минимальное искажение с пересчетом нового положения центроидов (RC), и без пересчета центроидов (WRC). В обоих случаях (RC, WRC) эффективность оценивалась по суммарному искажению (Error) и времени работы алгоритма в секундах (Time).

В таблице 1 и на рис. 2 (1000 центроидов) и 3 (4000 центроидов) показана зависимость искажения и времени работы от количества кандидатов в центроиды для обоих вариантов выбора i-го центроида (RC иWRC).

Таблица 1

Суммарное искажение для разного количества кандидатов: *n*=50000

К-во центроидов, k	Размерность	Вибор i-го центроида	Количество векторов-кандидатов в центроиды, |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			$1{,}188\cdot10^9$	$1{,}167\cdot10^9$	$1{,}155\cdot10^9$	$1{,}146\cdot10^9$

Из таблицы 1 видно, что увеличение числа кандидатов на выбор места первоначального

расположения нового центроида позволяет достичь уменьшения искажения. При этом имеет место экспоненциальный характер зависимости времени от количества кандидатов, что показано на рис. 2 и 3. С увеличением числа кандидатов более стремительный рост времени наблюдается для RC. Кроме того, при использовании более грубой оценки положения центроидов по WRC, наблюдается незначительное увеличение искажения (около 1%), тогда как время уменьшается на 5 – 20% (в зависимости от количества кандидатов). С ростом размерности характер зависимости сохраняется, абсолютные же значения искажения возрастают на порядок.

Итак, можно сделать выводы, что увеличение числа кандидатов позволяет достичь лишь незначительного уменьшения суммарного искажения за счет возрастания времени вдвое. Поэтому большей эффективности можно достичь, используя RC, с количеством кандидатов $|C| = 4 \cdot k$ (т. е. 4000 кандидатов для генерации 1000 центроидов и 16000 кандидатов для получения 4000 центроидов), поскольку это позволяет достичь меньшего искажения без значительного замедления работы алгоритма по сравнению с WRC, где примерно такое же значение искажения достигается при $|C| = 8 \cdot k$.

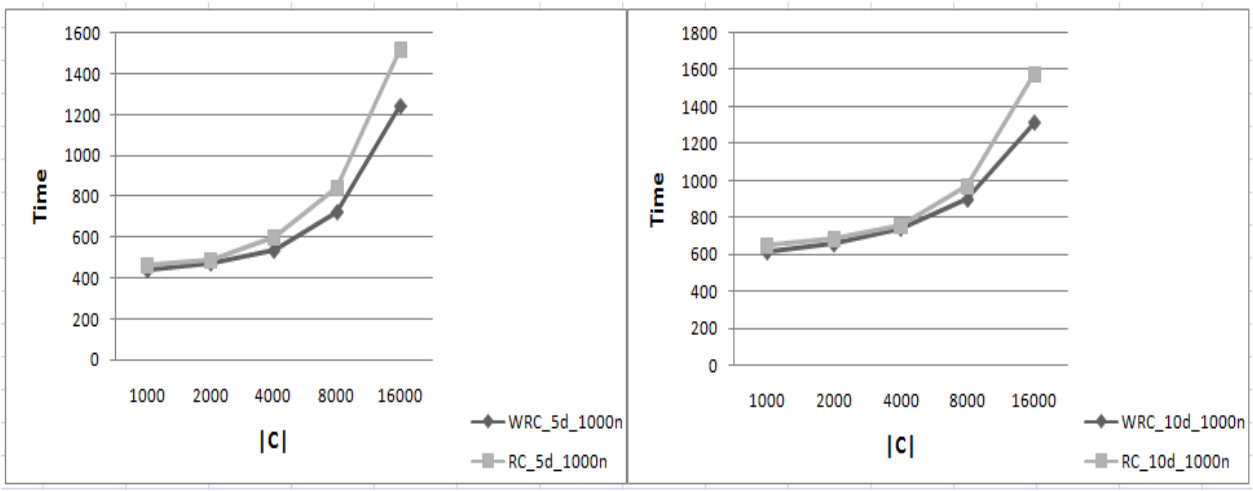

Рис. 2. Зависимость времени работы от количества входных векторов

а) $d = 5, n = 1000$ б) $d = 10, n = 1000$

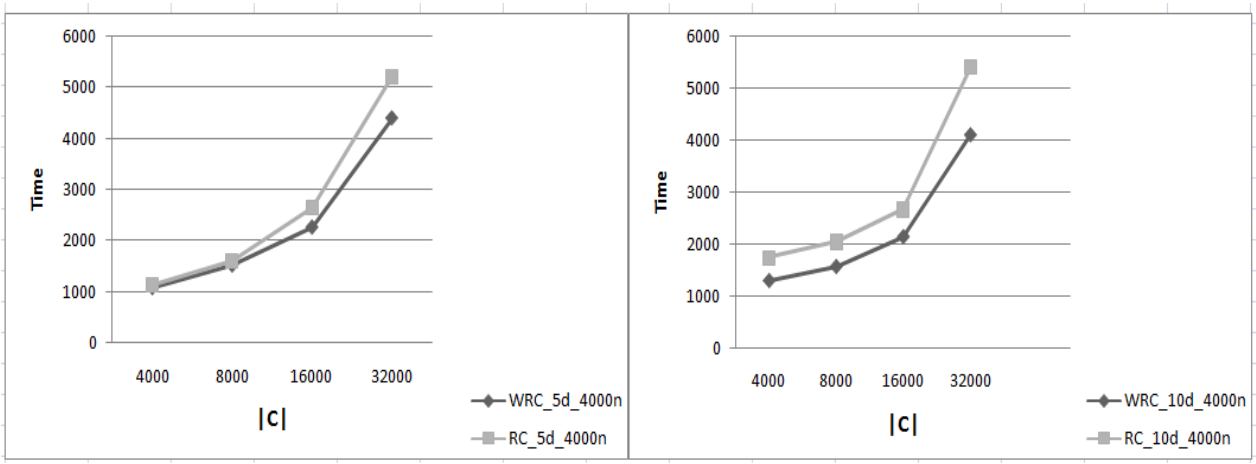

Рис. 3. Зависимость времени работы от количества входных векторов

а) $d = 5, n = 4000$ б) $d = 10, n = 4000$

Предложенный в данной работе алгоритм кластеризации (RC при $|\mathbf{C}| = 16000$) также сравнивали с двумя модификациями алгоритма k-средних (что показано на рис. 4): классическим алгоритмом k-средних (далее k-means), реализованным в MATLAB без

ограничения количества итераций и реализованным на основе kd-деревьев с максимальным количеством итераций 500 (далее hybrid), предложенным в [12]. Специфика алгоритма hybrid в том, что для приближения к глобальному минимуму в нем соединяются классический алгоритм k-средних и локальный поиск (обмен между существующими центроидами и кандидатами в центроиды при условии, что такой обмен приводит к уменьшению среднего искажения).

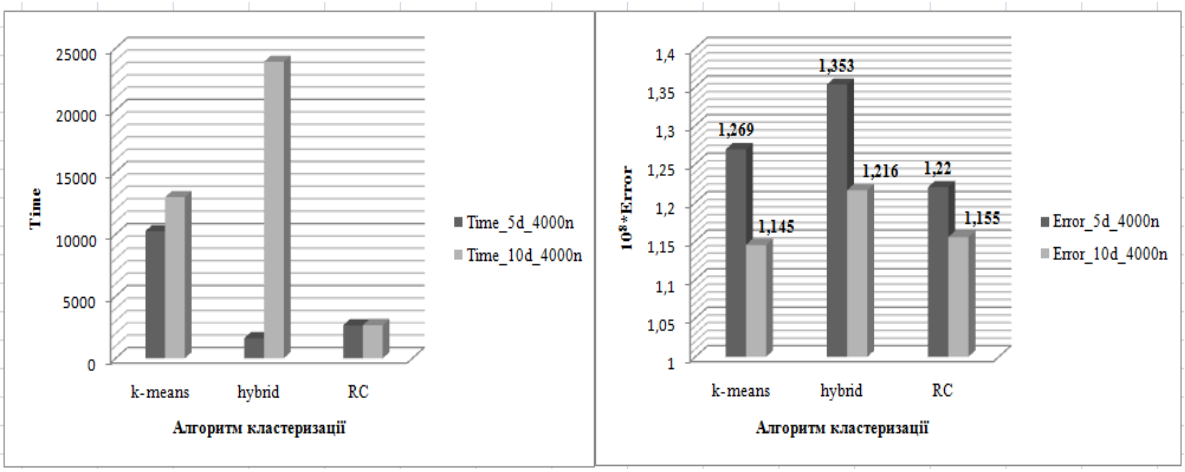

Рис. 4. Сравнение алгоритмов по сумарному искажению (слева) и времени работы (справа): $n = 4000$

Результаты, приведенные слева на рис. 4, показывают, что для обеих размерностей наибольшее значение суммарного искажения наблюдается при применении алгоритма кластеризации на основе kd-деревьев, наименьшее же искажение позволяют получить RC для $d = 5$ и k-means (MATLAB) для $d = 10$.

Справа на рис. 4 эффективность алгоритмов оценивают по времени работы. Так, для $d = 5$ наибольшее быстродействие достигается алгоритмом с использованием kd-деревьев, однако с увеличением размерности для hybrid наблюдается стремительный рост времени работы, что обусловлено экспоненциальным характером зависимости времени от размерности. Отметим, что эффективность предложенного в работе алгоритма практически не зависит от размерности.

Выводы

Предложенный в работе усовершенствованный метод кластеризации k-средних позволяет получить решение, приближенное к глобальному минимуму искажения. Уменьшение суммарного искажения составляет до 11% по сравнению с алгоритмом hybrid и до 5% при использовании алгоритма k-means (MATLAB). По скорости работы предложенный метод дает результаты лучше в 4 – 8 раз, чем алгоритм k-means (MATLAB) для размерностей $d = 5$ и $d = 10$ соответственно, и в 9 раз, чем hybrid для $d = 10$, в случае, когда $d = 5$, он несколько проигрывает hybrid в быстродействии (в 1,6 раз). Отметим, что эффективность предложенного в работе метода практически не зависит от размерности. Уменьшение погрешности кластеризации достигается за счет улучшенной процедуры выбора векторов-кандидатов на вставку нового центроида (RC) и определения оптимального количества кандидатов, которое составляет $|C| = 4 \cdot k$ и приводит к замедлению работы алгоритма в 1,5 – 2 раза.

СПИСОК ЛИТЕРАТУРЫ

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 Александр Николаевич – к. т. н., доцент кафедры вычислительной техники, тел. 59-84-13, e-mail: ant@vstu.vinnica.ua.

Грийо Тукало Оксана Франсисковна – студентка института информационных технологий и компьютерной инженерии, тел. 0662819319, e-mail: xxmargoxx@gmail.com.

Дзись Алексей Викторович – студент института информационных технологий и компьютерной инженерии, e-mail: alexdz47@gmail.com.

Лаховец Сергей Максимович – студент института информационных технологий и компьютерной инженерии, e-mail: selema.tsx@gmail.com.

Винницкий национальный технический университет.