ПОСЛЕДОВАТЕЛЬНЫЙ И ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМЫ КЛАСТЕРИЗАЦИИ СИМВОЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

О.И. Нечаева

Новосибирский государственный университет

Введение

Во многих областях науки возникает задача, в которой требуется разбить множество некоторых объектов на подмножества, элементы которых близки в некотором смысле. Такого рода задачи называются задачами *кластеризации* множества объектов, а получившиеся подмножества – *кластерами*. При этом следует заметить, что понятие «кластер» в данном случае определяет подмножество близких между собой элементов, а не вычислительную систему, состоящую из нескольких компьютеров.

В данной работе в качестве объектов выступают символьные последовательности, т.е. последовательности символов некоторого конечного алфавита.

Кластеризация применяется в самых разнообразных областях. Например, в медицине – для систематики заболеваний и их лечения, в генетике – для изучения взаимосвязи между структурой генетической цепочки и информацией, зашифрованной в различных её частях, в археологии – для установления таксономии различных находок, в музыке – для выявления структуры мелодии или установления плагиата и т.д.

Одним из важнейших методов кластеризации является «итерационный метод динамических ядер» [1], или метод «k-средних».

В статье рассматривается последовательная и параллельная реализация алгоритма кластеризации символьных последовательностей методом k-средних с использованием расстояния Левенштейна.

Задача кластеризации и последовательный алгоритм её решения методом k-средних заключаются в следующем. Пусть Y – это некоторый *алфавит* символов, Y* – множество всевозможных последовательностей символов из Y. Рассмотрим некоторое множество символьных последовательностей $X \subseteq Y^*$, $X = \{ x^1, \ldots, x^m \}$, где $x^i = [y^i_1, y^i_2, \ldots, y^i_{Ni}]$, $y^i_j \in Y$, N_i – длина i-й последовательности.

Для того чтобы проводить кластеризацию множества символьных последовательностей, нужно определить меру близости между его элемен-

тами. Для символьных последовательностей одной из наиболее распространённых мер близости является расстояние Левенштейна.

Расстоянием Левенштейна $d(x^1, x^2)$ между двумя последовательностями $x^1 = [\, y^1_1, y^1_2, \ldots, y^1_{N1}\,]$ и $x^2 = [\, y^2_1, y^2_2, \ldots, y^2_{N2}\,]$ называется неотрицательное целое число, равное минимальному количеству операций «вставок», «удалений» и «замен» символов в x^1, необходимых, чтобы из x^1 получить x^2. Вычислить расстояние Левенштейна можно, например, с помощью алгоритма Вагнера–Фишера [2], который основан на методе динамического программирования.

В процессе вычисления расстояния между последовательностями $[\, y^1_1, \ldots, y^1_i, \ldots, y^1_{N1}\,]$ и $[\, y^2_1, \ldots, y^2_j, \ldots, y^2_{N2}\,]$ заполняется массив $\{d_{i,j}\}$ размером $(N_1 + 1)\times(N_2 + 1)$ по рекуррентной формуле

$$d_{i,j} = \begin{cases} i, & j = 0 \\ j, & i = 0 \\ \min\left\{d_{i-1\,j} + 1, d_{ij-1} + 1, d_{i-1\,j-1} + \delta\left(y^1_i, y^2_j\right)\right\}, & i > 0, j > 0 \end{cases} \qquad (1)$$

где $d_{i,j} = d([\, y^1_1, y^1_2, \ldots, y^1_i\,], [\, y^2_1, y^2_2, \ldots, y^2_j\,])$, $\delta(y^1_i, y^2_j)=1$, если $y^1_i \neq y^2_j$, и $\delta(y^1_i, y^2_j)=0$ в противном случае. Искомым расстоянием Левенштейна между x^1 и x^2 является число, равное последнему элементу массива $\{d_{i,j}\}$, т.е. $d(x^1, x^2)=d_{N1,N2}$.

Для решения задачи кластеризации применим метод kсредних [1], описание которого приводится ниже.

Пусть $P_1, \ldots, P_k$, где k – зафиксировано, $k \le m$, разбиение множества X на кластеры, при этом $X = P_1 \cup P_2 \cup \ldots \cup P_k$, $P_i \cap P_i = \varnothing$ для всех пар (i, j). С каждым кластером P_i связан некоторый элемент a^i из Y* – «типичный представитель», или *ядро* кластера, и отнесение очередной последовательности к кластеру проводится путём сравнения её с ядрами всех кластеров и выбора ближайшего. Ядро a^i является аналогом центра тяжести кластера P_i и должно удовлетворять соотношению

$$a^i = \arg\min_{a \in A} \sum_{x \in P_i} d(a,x),\ i = 1,\ldots,k, \qquad (2)$$

где $A \subseteq Y^*$ – множество ядер, которое в общем случае не совпадает с множеством X.

Математическая постановка метода k-средних заключается в том, чтобы найти набор ядер $a^1, \ldots, a^k$ и разбиение множества X на кластеры $P_1, P_2, \ldots, P_k$, минимизирующие следующую величину:

$$D = \sum_{i=1}^{k} \sum_{x \in P_i} d\left(a^i, x\right). \qquad (3)$$

Ниже приведены основные шаги последовательного варианта итерационного алгоритма кластеризации методом k-средних. Будем обозначать через n номер итерации.

Последовательный алгоритм:

0. Фиксируется начальное разбиение $P_1(0), ..., P_k(0)$ произвольно либо по какому-нибудь эвристическому правилу, и в каждом множестве $P_i(0)$ вычисляется ядро $a^i(0)$, $i = 1...k$.

1. На итерации с номером n для каждой последовательности x^j, $j = 1...m$, вычисляются расстояния до всех k ядер и выбирается среди них минимальное:

$$d(x^j, a^i(n)) = min \{ d(x^j, a^1(n)), ..., d(x^j, a^k(n)) \}. \quad (4)$$

Последовательность x^j добавляется к множеству $P_i(n+1)$. В случае, когда минимум достигается при нескольких значениях i, выбор между ними может быть сделан произвольно.

2. Если $P_i(n+1)=P_i(n)$ для каждого $i = 1...k$, то алгоритм останавливается. Иначе в каждом множестве $P_i(n+1)$ вычисляется новое ядро $\hat{a}^i$, и делается переход к шагу 1 с найденными ядрами: $a^i(n+1) := \hat{a}^i$, $i = 1...k$.

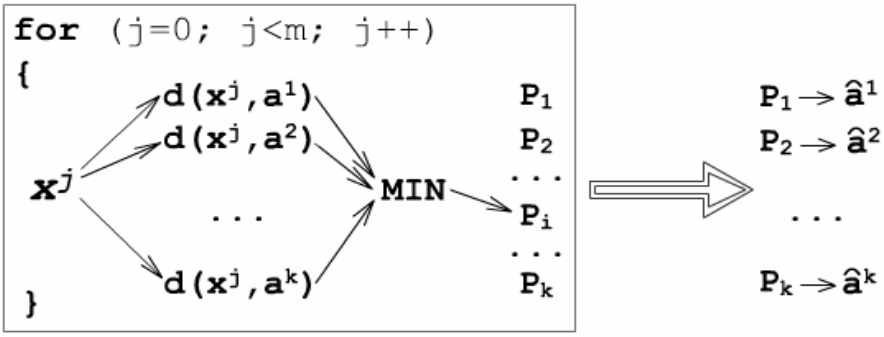


Рис. 1 Схема одной итерации алгоритма

На каждой итерации алгоритма уменьшается критерий качества D, отсюда следует сходимость алгоритма – после конечного числа шагов разбиение $P_1, P_2, ..., P_k$ уже не меняется.

Главная вычислительная сложность в методе k-средних с расстоянием Левенштейна – это *поиск ядра в кластере*. Задача нахождения «центра тяжести» с использованием этого расстояния, вообще говоря, является NP-полной.

Если искать ядро полным перебором, то сложность будет порядка $O(tN^2|\Sigma|^N)$, где t – количество последовательностей в кластере P; N – это максимум среди длин N_i, $i = 1...t$; Σ – множество символов, встречающихся в последовательностях кластера, $\Sigma \subseteq Y$, $|\Sigma|$ – количество этих символов. Для уменьшения сложности можно использовать эвристический алгоритм, который основан на том, что ядро выбирается среди последовательностей кластера.

Пусть $x^1, ..., x^t$ – последовательности кластера P. Тогда ядром кластера будем называть $x^p \in P$, сумма расстояний от которого до всех элементов кластера минимальна:

$$x^p = arg \min_{x \in P} \sum_{i=1}^{t} d(x, x^i). \quad (5)$$

Сложность нахождения ядра в кластере P равна $O(N^2 t^2)$, так как вычисление расстояния Левенштейна для двух последовательностей x^i и x^j имеет сложность $O(N_i N_j)$ и вычисляется t^2 раз.

Однако если проводится кластеризация большого числа длинных последовательностей, этот эвристический алгоритм также практически неприменим. Существуют методы, позволяющие уменьшить время вычислений. Например, можно построить более эффективный по качеству или по времени алгоритм нахождения ядра [3], но это возможно сделать, лишь обладая информацией о некоторых специфических свойствах рассматриваемых последовательностей. Ещё один метод основан на переходе от последовательностей к их частотным словарям, а значит, в евклидово пространство, в котором все вычисления на каждой итерации линейны [4]. Однако этот метод эффективен только для последовательностей с небольшим алфавитом, таких как последовательности ДНК с алфавитом из четырёх символов.

В основном такого рода методы требуют ограничений на последовательности. Более универсальным способом ускорения вычислений является распараллеливание алгоритма метода k-средних.

Параллельная реализация метода k-средних

Если проанализировать последовательный алгоритм метода k-средних, то можно заметить в нем внутренний параллелизм, который заключается, во-первых, в том, что при распределении входных последовательностей по кластерам вычислять расстояния от ядер до элементов X можно одновременно. Во-вторых, вычисление ядер в кластерах может происходить независимо друг от друга. На основании этого можно построить параллельный алгоритм кластеризации методом k-средних.

Пусть $X = \{ x^1, ..., x^m \}$ – множество последовательностей, которое нужно разбить на k кластеров. Тогда для параллельной реализации потребуется $k+1$ процессоров, среди которых имеется один управляющий и k рабочих процессоров. Все процессоры соединены по принципу «каждый с каждым», и кластеру P_j соответствует j-й рабочий процессор.

Управляющий процессор фиксирует начальное разбиение $P_1(0), ..., P_k(0)$ произвольно либо по какому-нибудь эвристическому правилу и каждому j-му рабочему процессору посылает множество $P_j(0)$ соответственно. После остановки параллельного итерационного алгоритма результат кластеризации возвращается в управляющий процессор.

Параллельный итерационный алгоритм для рабочих процессоров

0. Исходными данными в j-м процессоре является множество $P_j(0)$, $j = 1...k$.

На n-м шаге выполняются следующие операции:

1. Каждый j-й процессор вычисляет ядро $a^j(n)$ множества $P_j(n)$, $j = 1...k$.

2. Все процессоры обмениваются найденными ядрами так, что в результате обмена в каждом из них оказываются все k ядер.

3. В каждом j-м процессоре элементы множества $P_j(n)$ распределяются по подмножествам $P_1'(n), ..., P_k'(n)$ следующим образом: для каждой последовательности $x \in P_j(n)$ вычисляются расстояния до всех k ядер и x помещается в $P_i'(n)$, у которого ядро имеет минимальное расстояние до x по (4). В случае, когда минимум достигается при нескольких значениях i, выбор между ними может быть сделан произвольно.

4. Каждый j-й процессор рассылает множества $P_i'(n)$, $i = 1...k$, $i \neq j$, соответствующим рабочим процессорам и от каждого из них получает подмножество $P_j'(n)$.

5. В каждом j-м процессоре множество $P_j(n+1) = \cup P_i'$, $i = 1...k$, является исходным для следующего итерационного шага.

6. Если во всех процессорах $P_j(n+1) = P_j(n)$, $j = 1...k$, то $P_1(n), ..., P_k(n)$ является результатом кластеризации.

Пункты 1 и 3 алгоритма являются наиболее затратными по времени и выполняются одновременно во всех рабочих процессорах. Поэтому использование этого параллельного алгоритма метода k-средних позволяет уменьшить сложность вычислений примерно в k раз.

Заключение

Исследована последовательная реализация метода k-средних с расстоянием Левенштейна для кластеризации символьных последовательностей.

Сделан вывод о том, что для большого числа символьных последовательностей этот метод практически неприменим, и сократить время вычислений зачастую возможно только для узкого класса последовательностей.

Параллельная реализация позволяет обойти эти ограничения и ускорить вычисления в среднем во столько раз, сколько кластеров нужно получить.

Литература

1. Albayrak S. Unsupervised Clustering Methods for Medical Data: An Application to Thyroid Gland Data // Lecture Notes in Compute Science. – Berlin: Springer-Verlag, 2003. – Vol. 2714. – P. 695.

2. Lee C.S. George. Neural Fuzzy Systems. – New Jersey: Prentice Hall P T R, 1996.
3. Gorban A.N., Porova T.G., Sadovsky M.G. Classification of Symbol Sequences over their Frequency Dictionaries: Towards the Connection Between Structure and Natural Taxonomy // Open Sys. Inform. Dyn. – 2000. – № 7. – P.1.
4. Кузнецов П.Г. Обучение при распознавании символьных последовательностей // Вычислительные системы. – 1994. – № 150. – С. 164.