Реализация модуля определения параметров сложной нагрузки на самоадаптирующиеся контейнеры данных

Д. Р. Потапов

Воронежский Государственный университет, г. Воронеж, Россия

Аннотация. В работе приведены результаты реализации модуля определения параметров сложной нагрузки на самоадаптирующиеся контейнеры данных. Представлено обоснование выбора модификации алгоритма EM и алгоритма инициализации kmeans++, дано краткое описание работы программы. Помимо этого, проведен анализ качества кластеризации (для одного и нескольких кластеров, смещения и устаревания данных) и времени работы модуля. Исходя из результатов тестирования, можно сказать, что модуль хорошо справляется с задачей определения параметров сложной нагрузки и может быть эффективно использован в самоадаптирующихся контейнерах данных.

Ключевые слова: хранение данных, эффективность кэша, самоадаптирующийся контейнер данных, нагрузка на контейнер, смесь нормальных распределений, кластеризация, EM алгоритм, алгоритм k-средних.

DOI 10.14357/20718632190108

Введение

Самоадаптирующийся контейнер данных [1] – структура данных, которая изменяет логику своей работы в зависимости от структуры нагрузки [2]. Под нагрузкой понимается последовательность операций вставки, удаления и выборки из контейнера данных. Самоадаптирующийся контейнер данных может быть реализован как путем использования различных структур данных для различных нагрузок [3, 4], так и с использованием кэша [5]. В этом случае размер кэша будет изменяться в зависимости от разницы скоростей между кэшем и основным хранилищем и параметров нагрузки на контейнер. В статье [5] выявлено, что основным параметром нагрузки является распределение ключей, которое в общем случае может быть сведено к нормальному. Однако на практике обычно нагрузка представляет собой набор простых нагрузок, так как запросы к хранилищу могут приходить из разных источников или для различных прикладных задач. Таким образом, необходимо выявлять параметры простых нагрузок для максимально эффективного использования кеша.

1. Анализ задачи

Модель данных, в которой учитывается несколько одномерных нормальных распределений с различными весами, называется смесью нормальных распределений (Gaussian mixture model - GMM) [6, 7]. Для заданной выборки χ объемом N смесью p(x), состоящей из K компонент $N(\mu,\ \sigma^2)$ с весами π_k , называется формула:

$$p(x) = \sum_{k=1}^{K} \pi_k N(\mu_k, \sigma_k^2), \quad (1)$$

где $\sum_{k=1}^{K} \pi_k = 1$.

Задача нахождения параметров таких распределений (сигма σ и матожидание μ) сводится к задаче кластеризации, для решения которой в основном используется алгоритм EM (Expectation Maximization) и его различные модификации [8, 9]. Цель данного алгоритма состоит в том, чтобы найти вектор параметров $\theta = (\pi_1, \dots, \pi_k, \mu_1, \dots, \mu_k, \sigma_1, \dots, \sigma_k)$, при котором функция правдоподобия (2) достигает максимума.

$$\ln p(\chi|\pi, \mu,) = \sum_{n=1}^{N} \ln\left\{\sum_{k=1}^{K} \pi_k N(x_n|\mu_k, \sigma_k^2)\right\}. \quad (2)$$

Для этого вводится вспомогательный вектор скрытых переменных $\boldsymbol{G}$, который позволяет упростить задачу максимизации. Данный вектор состоит из элементов g_{nk}, которые определяют некую апостериорную вероятность того, что элемент выборки x_n принадлежит кластеру k. EM алгоритм состоит из последовательного выполнения 2-х шагов:

1. Шаг E (Expectation) – пересчет вектора $\boldsymbol{G}$ на основе текущего значения вектора θ по формуле:

$$g_{nk} = \frac{\pi_k N(x_n|\mu_k, \sigma_k^2)}{\sum_{j=1}^{K} \pi_j N(x_n|\mu_j, \sigma_j^2)}, k = 1..K. \quad (3)$$

2. Шаг M (Maximization) – максимизация функции 2 и вычисление нового значения вектора θ на основе вектора $\boldsymbol{G}$ и текущего значения вектора θ (для $k = 1..K$) по формулам:

$$\pi_k^{new} = \frac{1}{K} \sum_{n=1}^{K} g_{nk}$$

$$\mu_k^{new} = \frac{1}{K\pi_k^{new}} \sum_{n=1}^{K} g_{nk} x_n$$

$$\sigma_k^{2\,new} = \frac{1}{K\pi_k^{new}} \sum_{n=1}^{K} g_{nk} (x_n - \mu_k^{new})^2$$

Итерации происходят до сходимости (норма разности векторов скрытых переменных или изменение логарифмического правдоподобия на каждой итерации не будет превышать заданную константу) или достижения максимального числа итераций.

Одним из важных недостатков алгоритма EM является то, что данный алгоритм находит локальный экстремум функции правдоподобия, значение которого может оказаться гораздо ниже, чем глобальный максимум. Таким образом, в зависимости от выбора начального приближения алгоритм может сходиться к разным точкам. Существует несколько подходов к инициализации начальных параметров (матожидание, дисперсия и вес для каждого одномерного нормального распределения $N(\mu_k, \sigma_k^2)$ k=1...K) [10, 11]. Наиболее простой способ задания начальных значений это случайный, когда вес задается как 1/K, дисперсия и матожидания выбираются случайным образом. Данный способ был протестирован в первую очередь, так как он не требует дополнительных затрат, однако даже при небольшом количестве компонент в большинстве случаев результирующие кластеры сильно отличались от входных.

Кроме того, были рассмотрены алгоритмы rndem и emEm [12, 13]. Данные алгоритмы также случайным образом задают входные параметры, но выполняет несколько неполных итераций основного алгоритма, после чего выбирают начальные значения с наибольшим значением функции правдоподобия. Однако в случае большого объема данных, выполнение нескольких итераций может занять существенное количество времени. Другие популярные подходы основаны на агломеративной иерархической кластеризации (hierarchical agglomerative clustering (HAC)) [14], однако эти методы работают даже медленнее чем предыдущие.

Одним из самых эффективных согласно исследованию [10], является алгоритм kmeans++ [15]. Поэтому именно он был реализован в модуле определения параметров сложной нагрузки. Кроме того, в условиях big data могут быть использованы масштабируемые версии данного алгоритма [16] или даже версия с использованием технологии MapReduce [17, 18]. kmeans++ представляет собой алгоритм выбора начальных значений (центров кластеров, называемых также центроидами) для проведения кластеризации. Для поиска таких значений необходимо выполнить несколько шагов:

1. случайным образом из выборки χ выбирается первый центроид c_1;

2. для всех оставшихся данных выборки χ находится расстояние $D(x)$ до ближайшего центроида;

3. выбирается новый центроид c_i таким образом, что элемент $x \in \chi$ может быть выбран в качестве центроида с вероятностью $\frac{D(x)^2}{\sum_{x \in \chi} D(x)^2}$;

4. шаги 2 и 3 необходимо повторять пока не будут выбраны все центроиды.

Кроме того, необходимо учитывать, что данные постоянно поступают в контейнер, зачастую в большом количестве и на большой скорости. Следовательно, необходимо применять специальные алгоритмы кластеризации на потоке [19, 20]. К таким алгоритмам относится EM алгоритм с использованием буфера. Модуль самоадаптирующегося контейнера данных был реализован с применением данного алгоритма: сначала данные в режиме онлайн накапливаются в буфере, затем в режиме оффлайн через некоторые заданные промежутки времени/заданное количество вновь прибывших элементов или по запросу контейнера (в случае если кластеризацию нужно провести немедленно) выполняется EM алгоритм кластеризации на буфере. Размер буфера является настраиваемым параметром и определяет, насколько быстро устаревают данные.

2. Программное обеспечение

Модуль определения параметров сложной нагрузки на самоадаптирующиеся контейнеры данных был реализован на языке JavaScript с использованием библиотеки gaussian-mixture-model [21], которая реализует модификацию алгоритма EM на потоке с использованием буфера. Кроме того, данная библиотека позволяет учитывать уменьшение актуальности данных с течением времени.

Для тестирования подпрограммы был реализован алгоритм генерации смеси нормальных гауссовских распределений и средства визуализации кластеризации в одномерном пространстве. Для генерации одного распределения используется центральная предельная теорема [22]. Для вычисления времени работы модуля использовалось Web Performance API языка JavaScript.

Тестирование модуля проводилось на следующей программно-аппаратной платформе:

1. Processor: Intel Core i7 6500U @ 2.50 GHz (2 cores).
2. RAM: 16 GB.
3. OS: Windows 10.
4. Веб-браузеры Google Chrome 68.0.3440.84 и Opera 54.0.2952.64.

3. Эксперименты

В первую очередь было проведено тестирование на одном кластере. Данная ситуация часто возникает в приложении, когда долгое время данные используются только для одной задачи или одним источником запросов. Например, это различные аналитические системы с задачами, требующими много данных и времени выполнения, работающие в режиме реального времени. Размер буфера был выбран $1.5*10^6$ элементов. Была сгенерирована выборка в количестве 10^6 ключей по закону нормального распределения N (-25, 5.5) и в потоковом режиме ключи поступали в модуль. На Рис. 1 изображены ключи и границы кластера, построенного модулем. На изображении кластер обозначен как окружность с радиусом 4σ, т.к. в интервал $(\underline{x} - 4\sigma; \underline{x} + 4\sigma)$ попадает

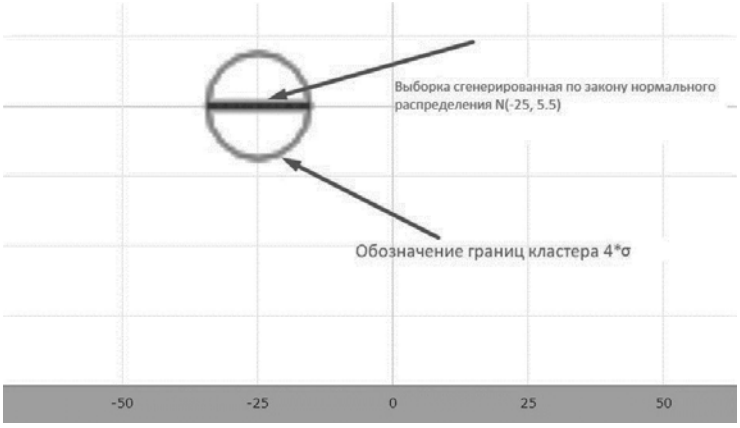

Рис. 1. Первичная выборка и кластер

99.993669986724854% всех ключей, принадлежащих кластеру [23], что является достаточным при таком количестве ключей в выборке.

Было проведено 10^6 таких тестов. В каждом были вычислены сигма и матожидание для полученной выборки. Параметры распределения, полученные в результате обработки данных ключей модулем кластеризации, совпали с вычисленными, что говорит о том, что алгоритм вычисления одного кластера работает верно.

Далее был протестирован случай, когда параметры распределения данных изменяются динамически. Для этого для заданного кластера в модуль было добавлено $5*10^5$ ключей в диапазоне [-51; -49] таким образом, что закон распределения кластера поменялся на N (-33, 142). На Рис. 2 изображены ключи и границы нового кластера, построенного модулем.

После этого была исследована ситуация с устареванием данных. Такая ситуация так же часто встречается в реальных приложениях при смене источников запросов или смене задач, требующих большего количества данных и времени выполнения. Для этого было добавлено еще 10^6 ключей из диапазона [-51; -49], что привело к тому, что модуль перестал учитывать данные из первоначального распределения N (-25, 5.5) и построил кластер N (-50, 0.5). На Рис. 3 наглядно изображен результат работы модуля после добавления нового набора данных.

Далее была исследована кластеризация нескольких кластеров. Данная ситуация наиболее часто встречается в современных приложениях с большим числом задач и источников запросов, например, веб-приложениях, высоконагруженных системах. Для этого было сгенерировано 5

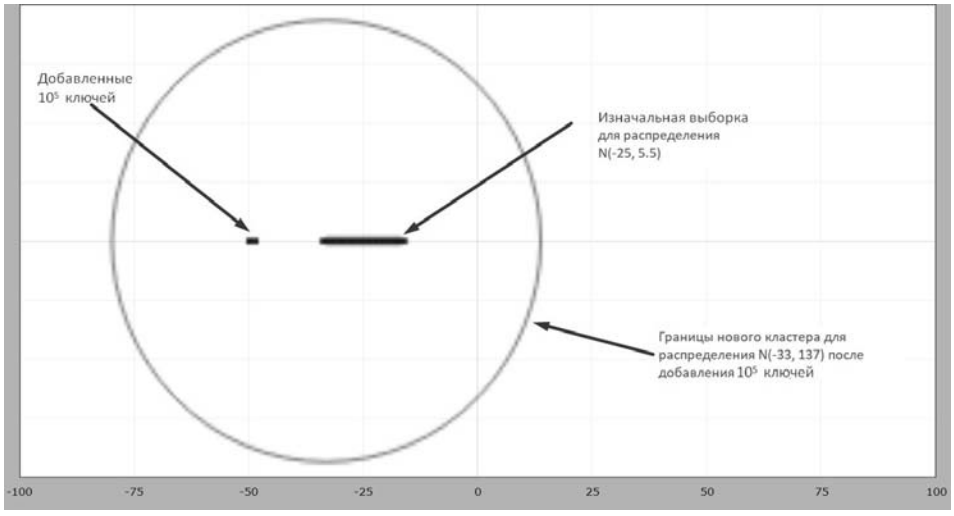

Рис. 2. Начальная выборка, добавленные 5*105 ключи и обновленный кластер

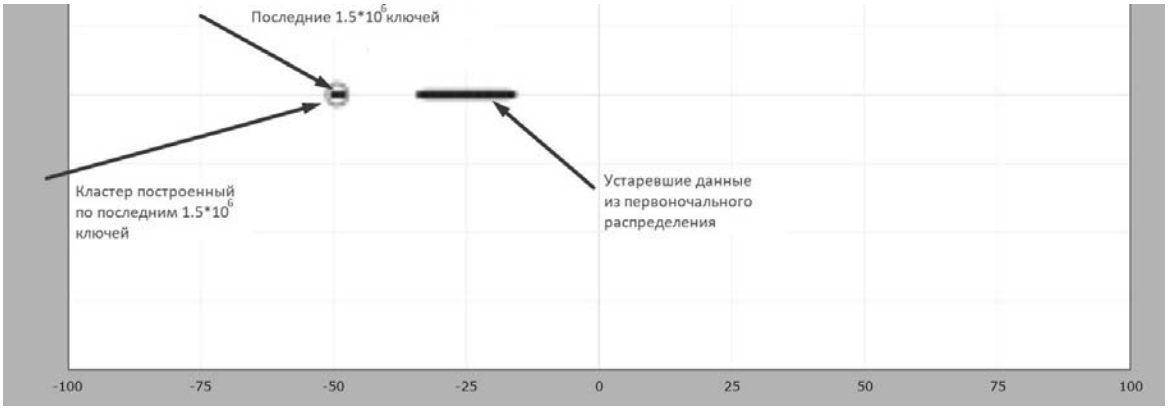

Рис. 3. Кластер после добавления 1.5*106 ключей и изначальная (неактуальная) выборка

выборок по законам нормального распределения N (-80, 2.35), N (-40, 2.35), N (0, 2.35), N (40, 2.35), N (80, 2.35) в количестве 10^7 ключей каждая и в потоковом режиме ключи поступали в модуль. Размер буфера был выбран 10^8 элементов. В первую очередь была протестирована реализация модуля со случайной инициализацией. Было проведено 10^6 таких тестов. Однако при использовании такого подхода в большей части тестов кластеризация была проведена некорректно (успешно проведено было только 16% тестов). Примеры результатов такой кластеризации изображены на Рис. 4 (только на последнем графике изображена правильная кластеризация).

Далее на тех же данных модуль был протестирован с использованием kmeans++ инициализации. В результате для заданной смеси распределений кластеризация во всех 10^6 тестах была произведена достаточно корректно (доля ошибочных кластеризаций составила 4%). Кроме того, было протестировано пересечение кластеров (для смеси N (-50,22.5), N (-20, 22.5), N (0,5.55), N (20,12.3), Рис. 5), проверены ситуации со смещением кластеров и устареванием данных для нескольких кластеров.

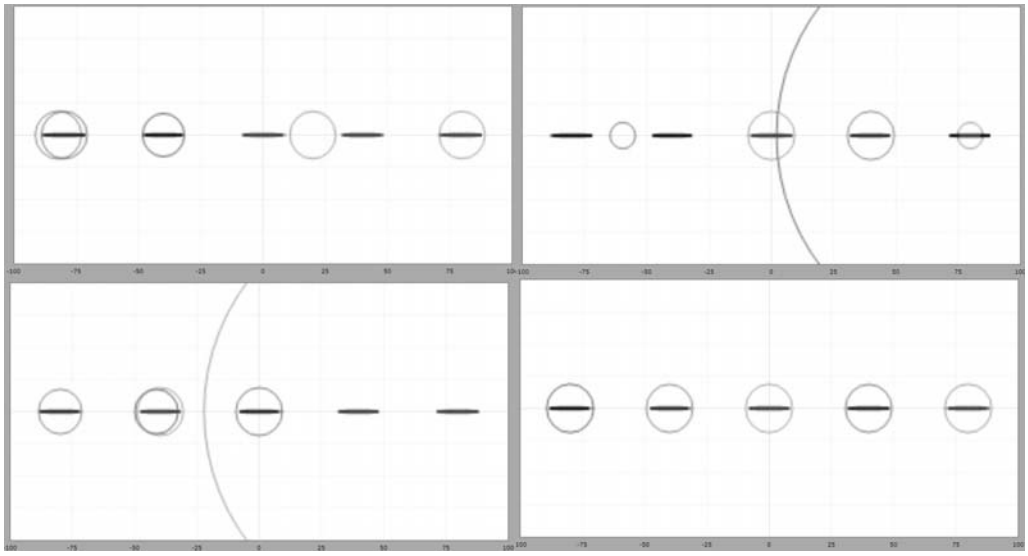

Рис. 4. Результаты кластеризации при случайной инициализации модуля

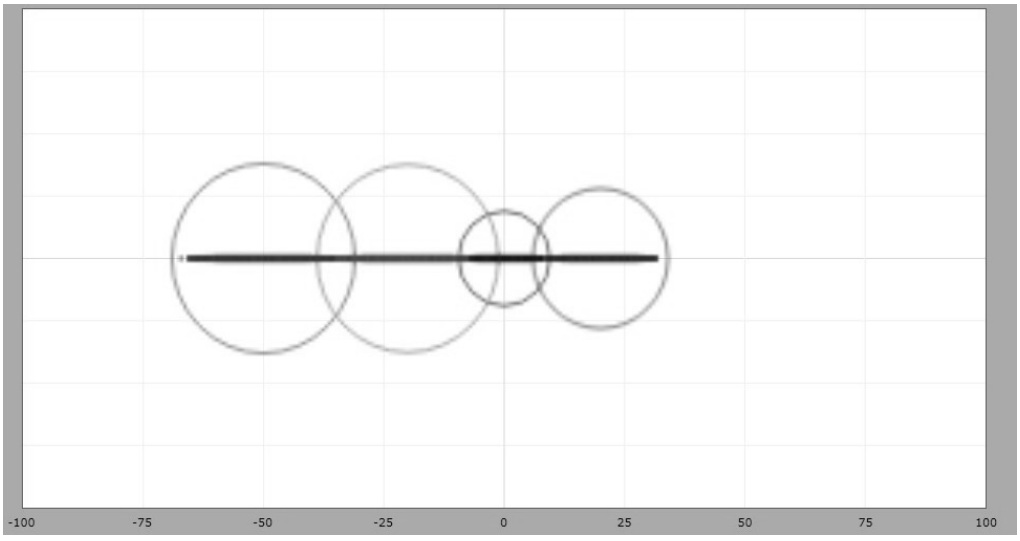

Рис. 5. Результаты кластеризации при пересечении кластеров

4. Анализ времени выполнения

Время выполнения EM кластеризации зависит линейно от количества итераций (i), количества кластеров (k) и количества ключей (n) в анализируемом буфере (сложность O(i*k*n)). Данная закономерность была подтверждена тестами, результат которых представлен в таблицах (Табл. 1-Табл. 3). Исходя из этих данных, можно прогнозировать скорость работы в зависимости от количества данных и кластеров.

Необходимо отметить, что кластеризация даже порядка 10^6 ключей для 4 кластеров занимает около 30 секунд (аналогичные результаты для алгоритма EM с буферизацией были получены в исследовании [24]). Несмотря на довольно долгое выполнение алгоритма EM, эффективность модуля определения параметров сложной нагрузки даже в условиях большой нагрузки будет высокой. Это связано в первую очередь с тем, что выполнение кластеризации не должно происходить слишком часто, потому что изменение кэша в самоадаптирующемся контейнере на основе полученных от модуля параметров не должно происходить слишком часто, так как перестроение кэша для большого объема данных требует много ресурсов. Кроме того, производительность модуля может быть улучшена несколькими способами:

1. увеличением размера буфера;
2. созданием временного буфера для поступающих во время кластеризации ключей.

Заключение

В статье приведены результаты реализации модуля определения параметров сложной нагрузки на самоадаптирующиеся контейнеры данных. В данной работе показано, что модель сложной нагрузки может быть рассмотрена как смесь нормальных распределений, а для определения параметров такой модели может быть использована потоковая версия алгоритма кластеризации EM (Expectation-Maximization). Кроме того, предоставлен краткий обзор алгоритмов инициализации алгоритма EM, обоснование выбора kmeans++ и краткое описание общей схемы работы модуля.

Табл. 1. Зависимость времени кластеризации от количества кластеров

Количество итераций и ключей в буфере	Количество кластеров	Среднее время кластеризации (в миллисекундах)
i=2 $n=6*10^4$	k=1	527
	k=2	872
	k=3	1253
	k=4	1567
	k=5	1891

Табл. 2. Зависимость времени кластеризации от количества ключей в буфере

Количество итераций и кластеров	Количество ключей в буфере	Среднее время кластеризации (в миллисекундах)
i=2 k=4	$n=6*10^4$	1567
	$n=12*10^4$	3043
	$n=24*10^4$	6244
	$n=48*10^4$	13395
	$n=96*10^4$	27129

Табл. 3. Зависимость времени кластеризации от количества итераций

Количество кластеров и ключей в буфере	Количество итераций	Среднее время кластеризации (в миллисекундах)
$n=6*10^4$ k=4	i=2	1567
	i=20	12566
	i=200	145004

Помимо этого, в статье представлены результаты тестирования модуля для одного и нескольких кластеров, смещения и устаревания данных. Точность кластеризации модуля при использовании k-means++ составила 100% для одного кластера и 96% для нескольких. Также приведены результаты экспериментов реализации модуля со случайной инициализацией (точность кластеризации нескольких кластеров составила 16%). Кроме того, в статье приведены результаты экспериментов по определению зависимости времени работы модуля от количества кластеров, количества итераций и количества ключей в буфере, которые позволяют прогнозировать время работы модуля. Исходя из результатов тестирования, можно сказать, что данный модуль хорошо справляется с задачей определения параметров сложной нагрузки и может быть эффективно использован в самоадаптирующихся контейнерах данных.

Литература

1. Потапов Д. Р., Артемов М.А., Барановский Е.С. Обзор условий адаптации самоадаптирующихся ассоциативных контейнеров данных // Вестник вгу, серия: системный анализ и информационные технологии. Воронеж, 2017. №1. С. 112-119.
2. Зобов В. В., Селезнев К.Е. Инструмент для моделирования нагрузки на контейнеры данных // Материалы четырнадцатой научно-методической конференции «Информатика: проблемы, методология, технологии». — Воронеж, 2014. – Т. 3. – С. 154–161.
3. Потапов Д. Р. Обзор методов построения многомерных контейнеров данных «ключ-значение» для использования в самоадаптирующихся контейнерах данных // Прикладная информатика, 2018. №2(74). С. 69-82.
4. Потапов Д. Р., Артемов М.А., Барановский Е.С., Селезнев К.Е. Обзор методов построения контейнеров данных «ключ-значение» для использования в самоадаптирующихся контейнерах данных // Кибернетика и программирование, 2017. №5. С. 14-45.
5. Потапов Д. Р. Исследование эффективности применения кеша для использования в самоадаптирующихся контейнерах данных // ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ // Принято к печати.
6. Bishop C. Pattern Recognition and Machine Learning. — Heidelberg: Springer, 2006. 738 p.
7. McLachlan G., Peel D. Finite Mixture Models. — NY: John Wiley & Sons, 2004. 419 p.
8. Королёв В.Ю. EM-алгоритм, его модификации и их применение к задаче разделения смесей вероятностных распределений. Теоретический обзор. — М.: Изд-во ИПИ РАН, 2007. 102 с.
9. McLachlan G., Krishnan T. The EM algorithm and extensions. Wiley series in probability and statistics. — NY: John Wiley & Sons, 1997. 400 p.
10. Blomer J., Bujna K. Simple methods for initializing the EM algorithm for Gaussian mixture models. // Computing Research Repository, 2013. http://arxiv.org/abs/1312.5946
11. Baudry J.-P., G. Celeux. EM for Mixtures. // Statistics and Computing, 2015. Vol. 25. No. 4. P. 713–726.
12. Melnykov V., Melnykov I. Initializing the EM algorithm in Gaussian mixture models with an unknown number of components. // Computational Statistics & Data Analysis, 2012. Vol. 56. No.6. P. 1381-1395.
13. Biernacki C., Celeux G., Govaert G. Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models. // Computational Statistics & Data Analysis, 2003. Vol. 41. No. 3-4. P. 561–575.
14. Meila M., Heckerman D. An Experimental Comparison of Several Clustering and Initialization Methods. // Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, 1998. P. 386–395.
15. Arthur D., Vassilvitskii S. K-means++: The Advantages of Careful Seeding. // Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms, 2007. P. 1027–1035.
16. Bahmani B., Moseley B., Vattani A., Kumar R., Vassilvitskii S. Scalable k-means++. // Proceedings of the VLDB Endowment, 2012. Vol.5. No.7. P.622-633.
17. Zhao W., Ma H., He Q. Parallel K-means clustering based on mapReduce. // Proceedings of the 1st International Conference on Cloud Computing, 2009. Vol. 5931. P. 674–679.
18. Xu Y., Qu W., Li Z., Ji C., Li Y., Wu Y. Fast Scalable k-means++ Algorithm with MapReduce. // Algorithms and Architectures for Parallel Processing. ICA3PP 2014, 2014. Vol. 8631. P. 15-28.
19. Aggarwal C.C., Han J., Wang J., Yu P.S. A framework for clustering evolving data streams. // Proceedings of the 29th international conference on Very large data bases, 2003. P. 81-92.
20. Liang P., Klein D. Online EM for unsupervised models // Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2009. P. 611-619.
21. Unsupervised machine learning with multivariate Gaussian mixture model which supports both offline data and real-time data stream. https://github.com/lukapopijac/gaussian-mixture-model
22. Круглов В. М., Королев В. Ю. Предельные теоремы для случайных сумм. М.: Изд-во Моск. ун-та, 1990. 269 с.
23. Гмурман В. Е. Теория вероятностей и математическая статистика: учебник для прикладного бакалавриат. М.: Издательство Юрайт, 2014. 479 с.
24. Bradley P.S., Fayyad U. M., Reina C. A. Scaling EM (Expectation-Maximization) Clustering to Large Databases. Microsoft Research Technical Report MSR-TR-98-35, 1999.

Потапов Данила Романович. Воронежский Государственный Университет (ВГУ), г. Воронеж, Россия. Аспирант. Количество печатных работ: 8. Область научных интересов: информационные технологии. e-mail: potapovd36@gmail.com

Implementation of the module for determining complex load parameters for self-adapting data containers

D. R. Potapov

Voronezh State University, Voronezh, Russia

Abstract. In applications with a large amount of the static data or data which is using for reading mostly cache applying improves performance greatly. To achieve maximum efficiency in an adaptive data storage implementation cache size can be changed dynamically during execution based on difference between speed of a main container and the cache, and container load. The main parameter of load is a set of requesting data, which in common case can be described as Gaussian distribution. But in a real world the container load is a set of simple loads mostly, because requests to data storage can be made by many applications or different tasks. Thus, parameters of such loads should be identified to achieve cache maximum efficiency. This paper provides implementation of the module for determining complex load parameters for self-adapting data containers results. The choice of EM modification, k-means++ initialization, and module structure brief description are also explained in this work. Clustering quality (for one and many clusters, concepts drift and time frame) and module execution time in this research are analyzed. Based on tests results, it can be said, that this module is good enough for determining complex load parameters and can be used in self-adapting data containers effectively.

Keywords: store the data, cache efficiency, optimal data storage, adaptive data container, container load, gaussian mixture model, clustering, EM, k-means.

DOI 10.14357/20718632190108

References

1. Potapov, D. R., M. A. Artemov, and E. S. Baranovskii. 2017. Obzor uslovii adaptatsii samoadaptiruyushchikhsya assotsiativnykh konteinerov dannykh [Review adaptation conditions of adaptive associative data storages]. Vestnik Voronezhskogo gosudarstvennogo universiteta. Seriya: Sistemnyi analiz i informatsionnye tekhnologii 1: 112-119.
2. Zobov, V. V., and K. E. Seleznev. 2014. Instrument dlya modelirovaniya nagruzki na konteinery dannykh [Tool for modeling the load on data containers]. Materialy chetyrnadtsatoi nauchno-metodicheskoi konferentsii «Informatika: problemy, metodologiya, tekhnologii 3: 154–161.
3. Potapov, D. R. 2018. Existing methods of multidimensional «key-value» storages construction for using in adaptive data storages review. JOURNAL OF APPLIED INFORMATICS 2(74): 69-82.
4. Potapov, D. R., M. A. Artemov, E. S. Baranovskii, and K.E. Seleznev. 2017. Obzor metodov postroenija kontejnerov dannyh «kljuch-znachenie» dlja ispol'zovanija v samoadaptirujushhihsja kontejnerah dannyh [Existing methods of "key-value" storages construction for using in adaptive data storages review]. Kibernetika i programmirovanie. 5:14-45.
5. Potapov, D. R. Issledovanie jeffektivnosti primenenija kesha dlja ispol'zovanija v samoadaptirujuwihsja kontejnerah dannyh [Cache efficiency research for using in adaptive data storage]. (In Russian, Unpubl.)
6. Bishop, C. 2006. Pattern Recognition and Machine Learning. Heidelberg: Springer. 738 p.
7. McLachlan, G., and D. Peel. 2004. Finite Mixture Models. NY: John Wiley & Sons. 419 p.
8. Korolev, V.U. 2007. EM-algoritm, ego modifikacii i ih primenenie k zadache razdelenija smesej verojatnostnyh raspredelenij. Teoreticheskij obzor [EM-algorithm, its modifications and their application to the problem of separation of mixtures of probability distributions. Theoretical review]. Moscow: IPI RAN. 102 p.
9. McLachlan, G., and T. Krishnan. 1997. The EM algorithm and extensions. Wiley series in probability and statistics. NY: John Wiley & Sons. 400 p.
10. Aggarwal, C.C., J. Han, J. Wang, and P.S. Yu. 2003. A framework for clustering evolving data streams. Proceedings of the 29th international conference on Very large data bases. Berlin. 81-92.
11. Liang, P., and D. Klein. 2009. Online EM for unsupervised models. Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Boulder. 611-619.
12. Blomer, J., and K. Bujna. 2013. Simple methods for initializing the EM algorithm for Gaussian mixture models. Computing Research Repository. Vol. abs/1312.5946.
13. Baudry, J.-P., and G. Celeux. 2015. EM for Mixtures. Statistics and Computing 25(4): 713–726.
14. Melnykov, V., and I. Melnykov. 2012. Initializing the EM algorithm in Gaussian mixture models with an unknown number of components. Computational Statistics & Data Analysis. 56(6): 1381-1395.
15. Biernacki, C., G. Celeux, and G. Govaert. 2003. Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models. Computational Statistics & Data Analysis. 41(3-4): 561–575.

16. Meila, M., and D. Heckerman. 1998. An Experimental Comparison of Several Clustering and Initialization Methods. Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. San Francisco. 386–395.
17. Arthur, D., and S. Vassilvitskii. 2007. K-means++: The Advantages of Careful Seeding. Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms. New Orleans. 1027–1035.
18. Bahmani, B., B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. 2012. Scalable k-means++. Proceedings of the VLDB Endowment. 5(7): 622-633.
19. Zhao, W., H. Ma, and Q. He. 2009. Parallel K-means clustering based on mapReduce. Proceedings of the 1st International Conference on Cloud Computing. Heidelberg. 674-679.
20. Xu, Y., W. Qu, Z. Li, C. Ji, Y. Li, and Y. Wu. 2014. Fast Scalable k-means++ Algorithm with MapReduce. Algorithms and Architectures for Parallel Processing. ICA3PP 2014 8631: 15-28.
21. Unsupervised machine learning with multivariate Gaussian mixture model which supports both offline data and real-time data stream. https://github.com/lukapopijac/gaussian-mixture-model
22. Kruglov, V. M., and V. U. Korolev. 1990. Predel'nye teoremy dlja sluchajnyh sum [Limit theorems for random sums]. Moscow: Moscow University Publishing. 269 p.
23. Gmurman, V. E. 2014. Teoriya veroyatnostej i matematicheskaya statistika : uchebnik dlya prikladnogo bakalavriata [Theory of Probability and Mathematical Statistics: A Textbook for Applied Bachelor Degree]. Moscow: Urait. 479p.
24. Bradley, P.S., U. M. Fayyad, and C. A. Reina. 1999. Scaling EM (Expectation-Maximization) Clustering to Large Databases. Microsoft Research Technical Report MSR-TR-98-35.

Potapov D. R. postgraduate student, Voronezh State University, Universitetskaya pl., 1, Voronezh 394006, Russia, email: potapovd36@gmail.com