Кластеризация неопределенного потока данных в режиме реального времени

А. З. Махмутова[1], И. В. Аникин[1], Kai-Uwe Sattler[2]
[1] Казанский национальный исследовательский технический университет имени А. Н. Туполева-КАИ
[2] Technische Universität Ilmenau

***Аннотация*. На сегодняшний день большой объем информации поступает с разных устройств, таких как сенсоры или датчики, в виде потока данных. Большая часть полученных данных связана с неопределенностью из-за неточности аппаратных средств и ошибок передачи. Поэтому обработка потоков данных и применение технологий интеллектуального анализа данных на них становится сложной процедурой со точки зрения скорости, точности и технических ограничений вычислительных устройств (организация распределенных вычислений, управление памятью и т. д.). Тем не менее, существуют современные технологии и инструменты, например, Storm, Samza, Spark Streaming и Flink, которые позволяют работать с потоками данных и берут на себя роль распределения вычислений, но они не предоставляют механизмов интеллектуального анализа для неопределенного потока данных. Кластеризация является одним из наиболее удобных методов интеллектуального анализа для потока данных, поскольку он обладает свойством адаптации под изменения данных в режиме реального времени, которое помогает системе отслеживать развитие записей без контроля со стороны человека. Неопределенность в данных добавляет сложности обработке и идентификации кластера. Таким образом, эта работа в основном фокусируется на исследованиях и разработке компонента кластеризации неопределенного потока данных в режиме реального времени. В рамках этой работы мы также проводим оценку неопределенных потоков данных для кластеризации и эффективности существующего решения для обработки потока данных.**

Ключевые слова: поток данных; неопределенность; кластеризация

I. Введение

На сегодняшний день большинство процессов было автоматизировано благодаря современным технологиям, что увеличило количество приложений, работающих в режиме реального времени и генерирующих потоки данных – непрерывный неограниченный по времени разнообразный по содержанию поток информации (текст, числа и т.д.). Обработка статичных и потоковых данных полностью различна вследствие поступления новой информации в определенный момент времени. Это особенность существенно ограничивает и накладывает определенные ограничения на обработку потоков не только с точки зрения аппаратных средств, но и анализа данных [1].

Методы интеллектуального анализа данных (англ. *Data mining*) доказали свою эффективность в поиске закономерностей и правил в масштабном количестве информации. Однако применение этих методов на потоковых данных накладывает дополнительные требования [2]:

- Мгновенная обработка полученных данных – без этого условия результаты анализа могут быть устаревшими.
- Компактное представление – необходимо разумно управлять постоянным потоком информации с точки зрения использования памяти.
- Обнаружение и обработка «выбросов» – запись, выходящая за пределы общей массы значений, может значительно воздействовать на результат, поэтому в потоковой обработке в режиме реального времени необходимо определить действия с этими данными.

Также можно выделить характеристики данных, затрудняющие процесс анализа: объем потока данных поступающий с многочисленных устройств, разнообразие поступающих данных (текст, численные значения, медиа), возможное изменение с течением времени, скорость обработки, достоверность данных (неопределенность и отклонения). Часть этих проблем является определением больших данных [4]. Для удовлетворения всех требований анализа потоковых данных следует использовать системы и инструменты для распределенной высокопроизводительной обработки больших данных, таких как Spark Streaming [5], Apache Flink [6] и т.п.

II. Подходы к кластеризации неопределенных потоковых данных

Потоки данных, содержащих неточности, ошибки или неполноту относят к неопределенным потокам. Для устранения неопределенности существуют два основных подхода [7]: предварительная обработка информации и исключение неопределенности или управление неопределенностью. Хотя первый подход представляется предпочтительным и более надежным, в случае потоков данных он не может применяться из-за отсутствия всего набора данных и требования оперативной обработки.

В базах данных может существовать неопределенность на уровне кортежей и атрибутов, что означает неопределенность в существование кортежа или значений атрибута соответственно. В первом случае метод [8], расширяющий исходную базу данных всеми возможными значениями и вычисляет вероятность для каждого из них. Неопределенность на основе атрибутов разделяется на два типа: нечет-

кое значение и значение неоднозначности [9]. В этой работе мы будем работать с значением неоднозначности, а это означает, что переменные определяются функцией плотности вероятности. В потоках наиболее необычным, но строгим форматом для записи является $\{D_t, f_t\}$, где t – метка времени, D –значение, а f – соответствующая непрерывная функция плотности. Другой случай, когда запись приходит в виде $\{(x(t_1), 0.15); (x(t_2); 0.40);\ldots (x(t_k); 0.05)\}$ относят к дискретному распределению. И третья модель показывает общее представление поступающей информации (обычно от датчиков), где известна стандартная ошибка r, что означает среднее отклонение данных $(x_t \pm r)$.

Адаптация алгоритмов кластеризации для неопределенных потоков может быть осуществлена в два этапа: сначала для неопределенных данных, а затем – для потока. В системах обработки в режиме реального времени кластеризация позволяет идентифицировать группировать записи в кластерах без каких-либо знаний о предстоящих данных [8].

А) Существуют несколько типов алгоритмов построения кластера: дистанционные (основаны на вычислении расстоянии между объектами), основанные на теории графов и плотностные алгоритмы [9]. Учитывая неопределенность в данных, известный метод кластеризации DBSCAN [10] был модифицирован в алгоритм кластеризации на нечетких объектах FDBSCAN [11]. Алгоритм OPTICS [12] преобразовался в алгоритм FOPTICS [13]. Самый популярный из использования в кластерной области алгоритм K-mean [14] был модифицирован в алгоритм UK-средств [15].

Во всех описанных алгоритмах для используемых в них концепциях добавляется оценкой и вычислением вероятности, что является основной трудностью. Одним из подходов является использование метода Монте-Карло [16] на выборке, но он неэффективен в рамках использовании на потоке данных. Идея уменьшения до минимума числа вычислений ожидаемого расстояния от точки до кластера представлена путем вычисления минимальных граничных прямоугольников (MBR) в дистанционных алгоритмах.

Б) Существующие методы делят кластеризацию потока на две фазы: онлайн и офлайн. Большое количество предстоящих записей не может быть сохранено в памяти, следовательно, онлайн-часть позволяет обрабатывать информацию за один проход [17] сбором сводной информации о полученных записях. В зависимости от того, какой тип информации был накоплен в онлайн-фазе, в автономном режиме могут быть выбраны различные методы кластеризации. В режиме реального времени информация о поступающих записях суммируется в *микрокластеры* [18], расширенную концепцию вектор-функции кластера (CFV) [19]. Многие методы кластеризации используют разделение на двухфазную обработку: DenStream [20], DCUStream [21], CluStream [18] – для детерминированных данных, Umicro [22], LuMicro [23] и EMicro [24] – для неопределенных.

При обработке массивного неопределенного потока данных необходимо обеспечить эффективный и масштабируемый подход для кластеризации данных в потоках с распределенной обработкой данных.

III. Программный комплекс для кластеризации неопределенного потока данных в режиме реального времени

Мы выбрали Apache Spark Streaming в качестве основы для разработки нашей инфраструктуры из-за его поддержки гибридной (пакетной и потоковой) обработки потока данных, а также стабильной и отказоустойчивой обработки. Также мы рассмотрели открытую библиотеку StreamDM предоставляющую два метода кластеризации потоков - *StreamKM++* [25] и *CluStream* [26]. Производительность онлайн-части второго метода мы будем сравнивать с нашей реализацией алгоритма *Umicro*.

Оба алгоритма используют понятие микро-кластеров и вычисляют принадлежность к кластеру на основе расстояния от центра. *CluStream* предпочтительнее для пакетной обработки, тогда как *Umicro* представляет собой истинное поточное представление обработки каждой приходящей записи. На этапе предварительной обработки *CluStream* необходимо определить центры кластеров, что обязывает систему как-то получить информацию о будущей структуре данных. *Umicro* может быть запущен без предварительной обработки, но на входе ему требуются многомерные записи с оцененными ошибками ψ для каждого измерения. Эта оценочная ошибка ψ также изменяет представление микро-кластера (CFV). Оба алгоритма имеют предопределенное ограниченное количество кластеров, и, если Umicro просто отслеживает временные метки и удаляет кластер с наибольшим временным штампом, CluStream имеет возможность объединить два самых старых кластера, близких по значению.

Реализация алгоритма Umicro была выполнена как расширение существующей библиотеки StreamDM, которая работает на движке Apache Spark Streaming [5]. Структура данных включает такую концепцию, как экземпляр – многомерный вектор данных. Для работы в реализации алгоритма экземпляры обернуты внутри объекта-примера, и они передаются как DStream. Он обеспечивает неизменную исходную обработку данных: каждая операция возвращает новый экземпляр с изменениями. Рис. 1 отображает общий рабочий процесс для онлайн-части кластерного подхода в разработанной структуре.

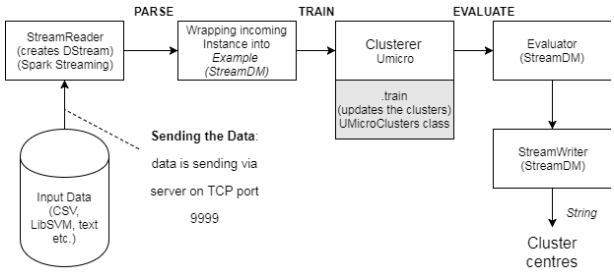

Рис. 1. Диаграмма процесса онлайн-кластеризации алгоритма Umicro

Работа программы может быть условно разделена на три части:

- шаг предварительной обработки – на этом этапе мы готовим данные к представлению, в котором они будет обрабатываться внутри реализации. Он включает в себя добавление части неопределенности (шума) к данным, получение Spark Streaming самих данные, и представление их в подходящий для следующей формы расчетов.
- стадия кластеризации – самая важная часть – формирование микро-кластеров, подсчёт расстояния точка-центр кластера и удаление кластера в случае превышения допустимого максимального уровня микро-кластеров.
- шаг оценки – эта часть содержит проверку корректности работы алгоритма. Основная концепция этого этапа будет дана в следующей части.

IV. Оценка производительности и эксперименты

Оценка работы проводилась по двум критериям: эффективность (аккуратность формирования кластеров) и производительность. Второй критерий направлен на оценку производительности и масштабируемости разработанной структуры и сравнение ее с оригинальным подходом. Для экспериментов в рамках разработанной структуры синтетические данные были созданы в виде 7-мерного кортежа tuple $< w; x_1; x_2; x_3; \psi_1; \psi_2; \psi_3 >$, где w – вес, который необходим, например, для обертывания, x_j – данные с предопределенным уровнем шума, а ψ_j содержит ошибку. Для данных без шума, чистых данных, ψ_j содержат ноль.

A. Оценка эффективности

С точки зрения и интересов исследователя кластеризация имеет разные интерпретации. Эффективность кластеризации можно оценить в двух основных категориях: внешнем и внутреннем. Первая учитывает сопоставление с некоторой внешней структурой, тогда как вторая использует только свои внутренние атрибуты для проверки. В случае неопределенных данных также должны учитываться значения вероятности самих точек. Чтобы обеспечить оценку эффективности работы алгоритма, на каждом шаге рассчитывали стандартное отклонение кластерных центров от первоначальных «чистых» значений при повышении уровня шума на каждом этапе:

$$S = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - x)^2} \qquad (1)$$

где x – представляет «чистое» значение центра кластера, с учетом каждого из измерений x_i представляет полученные значения с учетом каждого кластера и размерности, и i представляет собой количество подсчетов. Меньшее значение отклонения будет интерпретироваться как более точная работа алгоритма. Точность оценки была выполнена на небольшом наборе данных из 200 экземпляров для детального изучения работы обоих алгоритмов. Таблица содержит результаты оценки для каждого измерения. Из результатов оценки можно сделать вывод о том, что реализация UMicro обеспечивает более точный результат по данным с неопределенностью, чем алгоритм CluStream.

ТАБЛИЦА I СТАНДАРТНОЕ ОТКЛОНЕНИЕ ДЛЯ ДВУХ АЛГОРИТМОВ

Количество кластеров	CluStream			UMicro		
	Измерения			Измерения		
	1	2	3	1	2	3
1	28,775	9,9304	61,452	23,273	9,1824	41,888
2	23,345	21,700	32,181	21,876	6,9637	10,907
3	39,479	12,373	36,233	33,503	12,067	29,913
4	29,593	18,543	28,477	24,34	11,232	27,121
5	64,206	19,283	44,804	28,937	11,383	22,111

B. Оценка производительности

В случае этого исследования измерялась задержка по времени, необходимая на проведение вычислений. Результаты критериев оценки вычислительной мощности (каждый алгоритм работает с различными уровнями шума при различной вычислительной мощности – локальном и распределенном), представленном на рис. 2, и демонстрируют тенденцию, что при добавлении вычислительной мощности в систему подход *UMicro* показывает лучший результат работы.

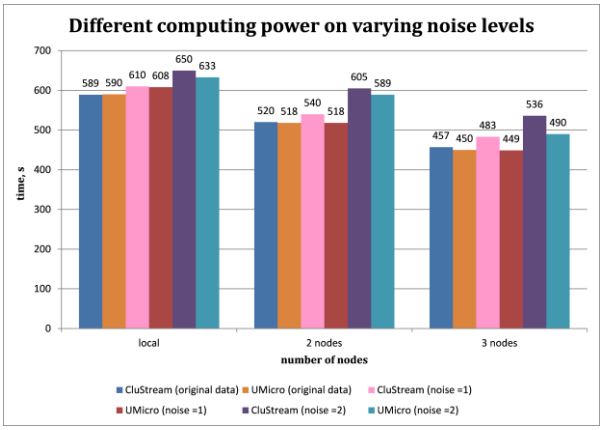

Рис. 2. Оценка производительности методов на разных уровнях вычислительной мощности при добавлении шума

V. Заключение

В этой работе мы представили программный комплекс для кластеризация неопределенного потока данных в режиме реального времени. Эта структура была реализована в среде Apache Spark Streming с внешней библиотекой StreamDM для обработки большого потока данных.

В будущем систему можно улучшить, добавив компонент отслеживания эволюции данных. В перспективе кластеры могут быть объединены вместо простого удаления. Кроме того, можно адаптировать различные модели окон из алгоритмов кластеризации на основе плотности для неопределенных данных.

Список литературы

[1] M. Stonebraker, Cetintemel, and S. Zdonik. The 8 requirements of real-time stream processing. Sigmod Record, 34(4):42–47, 2005.

[2] D. Barbar. Requirements for clustering data streams. ACM SIGKDD Explorations Newsletter, 3(2):23–27, 2002.

[3] B. Rupa and R. Soujanya. Data stream clustering issues and challenges-a survey. International Journal of Advanced research, 5(3):1644–1649, 2017.

[4] The four v's of big data. [Online] http://www.ibmbigdatahub.com/infographic/four-vs-big-data.

[5] Spark streaming. [Online]. https://spark.apache.org/streaming/.

[6] Apache flink. [Online]. https://flink.apache.org/.

[7] M. Lin. Uncertainty in data stream processing. Master's thesis, Technical University Ilmenau, 2015.

[8] S. Guha and R. Motwani. Clustering data streams: theory and practice. IEEE Transactions on Knowledge and Data Engineering, 15(3):515–528, 2002.

[9] H. Tan. Knowledge Discovery and Data Mining. Springer Berlin Heidelberg, 2012.

[10] E. Simoudis, J. Han, and U. M. Fayyadeds. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press., page 226-231, 1996.

[11] H.-P. Kriegel and M. Pfeifle. Density-based clustering of uncertain data. ACM KDD Conference Proceedings, 2005.

[12] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. ACM SIGMOD international conference on Management of data. ACM Press., pages 49–60, 1999.

[13] H.-P. Kriegel and M. Pfeifle. Hierarchical density based clustering of uncertain data. ICDM Conference, 2005.

[14] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. July 2002.

[15] W. Ngai, B. Kao, C. Chui, R. Cheng, M. Chau, and K. Y. Yip. Efficient clustering of uncertain data. ICDM Conference Proceedings, 2006.

[16] R. Jampani, F.Xu, and M.Wu. Mcdb: A monte carlo approach to managing uncertain data. Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 687–700, 2008.

[17] C.C. Aggarwal and C.K. Reddy. Data Clustering: Algorithms and Applications. Chapman and Hall/CRC, 2013.

[18] C.C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. Proceedings of the International Conference on Very Large Data Bases, 29:81–92, 2003.

[19] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. Proceedings of the 1996 ACM SIGMOD international conference on Management of data, pages 103–114, 1996.

[20] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. Proceeding The 2006 SIAM Conference on Data Mining, pages 328–339, April 2006.

[21] Y. Yang, Z. Liu, J. Zhang, and J. Yang. Dynamic density-based clustering algorithm over uncertain data streams. Proceedings. In 9th International Conference on Fuzzy Systems and Knowledge Discovery, pages 2664–2670, 2012.

[22] C. Aggarwal and P.S.Yu. A framework for clustering uncertain data streams. Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, pages 150–159, 2008.

[23] C. Zhang, M. Gao, and A. Zhou. Tracking high quality clusters over uncertain data streams. Proceedings. of the 2009 IEEE 25th International Conference on Data Engineering, pages 1641–1648, 2009.

[24] C. Zhang, C.-Q.Jin, and A.-Y. Zhou. Clustering algorithm over uncertain data streams. Journal of Software, 21(9): 2173–2182, 2010.

[25] Ackermann, Lammersen, Mrtens, Raupach, Sohler, and Swierkot. Streamkm++: A clustering algorithm for data streams. In Proceedings of the 12th Workshop on Algorithm Engineering and Experiments, 2010.

[26] A. Bifet, S. Maniu, J. Qian, G. Tian, C. He, and W. Fan. Streamdm: Advanced data mining in spark streaming. IEEE International Conference on Data Mining Workshop (ICDMW), page 1608 1611, 2015.

[27] streamdm: Data mining for spark streaming. [Online]. http://huawei-noah.github.io/streamDM/.