УДК:004.827

МОДИФИКАЦИЯ АЛГОРИТМА DBSCAN С ИСПОЛЬЗОВАНИЕМ ГИБРИДНЫХ ПОДХОДОВ К ОПРЕДЕЛЕНИЮ ГРАНИЦ КЛАСТЕРОВ ДЛЯ ОБРАБОТКИ ПОТОКОВЫХ ДАННЫХ

Митин Г.В., Панов А.В.

МИРЭА - Российский технологический университет, 119454, Россия, г. Москва, проспект Вернадского, 78, e-mail: grigory.mitin@mail.ru, insegnantenew@yandex.ru

В данной статье предлагается новый подход к решению задачи кластеризации с отсечением выбросов, неинформативных аномальных данных и прочего информационного шума для потоковых данных в пространстве признаков любой размерности и с памятью всех обработанных точек данных. Для реализации поставленной задачи была разработана оригинальная модификация алгоритма DBSCAN, использующая гибридный подход к поиску границ кластеров произвольной формы и определению, находится ли каждая из точек данных внутри или снаружи такой границы. При разработке были применены как технологии машинного обучения, так и математические методы, в частности метод вычисления выпуклой оболочки конечного набора точек в n-мерном пространстве Quickhull. Результирующий алгоритм состоит из нескольких блоков, активирующихся в зависимости от природы распределения данных полученных из входного потока. Применение разработанного алгоритма гарантирует создание замкнутой границы кластера произвольной формы. Использование механизма адаптивного разбиения на фреймы, позволяет проводить кластеризацию данных разной размерности и больших объемов, с памятью всех входящих точек. В результате авторам удалось создать модификацию алгоритма DBSCAN для потоковых данных эффективного по скорости выполнения и используемой памяти. Для иллюстрации прироста эффективности разработанной модификации алгоритма по сравнению с классическим вариантом DBSCAN проведена расчетная оценка производительности и требований к памяти. Правильность полученных оценок подтверждена экспериментально. Представленная модификация алгоритма DBSCAN для потоковых данных не только, позволяет получить общий прирост производительности при более низких требованиях к памяти по сравнению с классическим алгоритмом DBSCAN, но и имеет функциональные преимущества, связанные с возможностью эффективной работы с потоковыми данными при наличии информационного шума. Указанные преимущества делают представленную модификацию алгоритма DBSCAN полезной для решения сложных задач в системах обработки потоковых данных как, например, поиск корреляций и аномалий в статистических показателях распределенных систем сбора данных или для обнаружения устойчивых состояний моделей массового обслуживания, применяемых в логистике и на транспорте.

Ключевые слова: машинное обучение, обучение без учителя, гибридный алгоритм, кластеризация, алгоритм DBSCAN, потоковые данные, алгоритм вычисления выпуклой оболочки, плотность распределения данных, границы кластера, адаптивные фреймы данных, отсечение информационного шума, оценка производительности.

MODIFICATION OF DBSCAN ALGORITHM USING HYBRID METHODS FOR CLUSTERS BORDER DETECTION TO PROCESS STREAMING DATA

Mitin G.V., Panov A.V.

MIREA - Russian Technological University, 119454, Russia, Moscow, Vernadsky Avenue, 78, e-mail: grigory.mitin@mail.ru, insegnantenew@yandex.ru

This article proposes a new approach to solving the clustering problem with cutting off outliers, uninformative anomalous data and other information noise for streaming data in the feature space of any dimension and with the memory of all processed data points. To implement this task, an original modification of the DBSCAN algorithm was developed, using a hybrid approach to finding the boundaries of clusters of arbitrary shape and determining whether each of the data points is located inside or outside such a boundary. During the development, both machine learning technologies and mathematical methods were used, in particular, the method of calculating the convex hull of a finite set of points in the n-dimensional Quickhull space. The resulting algorithm consists of several blocks that are activated depending on the nature of the distribution of data received from the input stream. The application of the developed algorithm guarantees the creation of a closed cluster boundary of arbitrary shape. Using the adaptive frame splitting mechanism, it allows clustering of data

of different dimensions and large volumes, with the memory of all incoming points.As a result, the authors managed to create a modification of the DBSCAN algorithm for streaming data that is efficient in terms of execution speed and memory usage. To illustrate the efficiency, gain of the developed algorithm modification in comparison with the classic DBSCAN variant, a calculated assessment of performance and memory requirements was carried out. The correctness of the estimates obtained has been confirmed experimentally.
The presented modification of the DBSCAN algorithm for streaming data not only is able to get an overall performance gain with lower memory requirements compared to the classic DBSCAN algorithm, but also has functional advantages associated with the ability to work efficiently with streaming data in the presence of information noise. These advantages make the presented modification of the DBSCAN algorithm useful for solving complex problems in streaming data processing systems, such as searching for correlations and anomalies in statistical indicators of distributed data collection systems or for detecting stable states of queuing models used in logistics and transport.

Keywords: machine learning, unsupervised learning, hybrid algorithm, clustering, DBSCAN, streaming data, convex hull, density based, cluster borders, adaptive data frame, dealing with noise, performance check.

Введение

На сегодняшний день задача кластеризации является одноим из наиболее популярных применений методов машинного обучения с обучением без учителя. Существуют алгоритмы кластеризации, уже признанные «классическими» [1], способные эффективно решать множество прикладных задач. Однако разнообразие прикладных проблем, стоящих перед разработчиками, становится все больше, отчего растет необходимость в новых алгоритмах. Среди таких проблем стоит работа с потоковыми данными и большими объемами данных.

Существует большой пласт прикладных задач, связанных с обработкой потоковых данных, требующих поиск «неподвижных» областей в пространстве признаков, отвечающих значимым состояниям системы или уровням сигнала. Примером могут служить анализ данных сейсмической активности от массива датчиков для выявления характерных колебаний и фильтрации шумовых показаний [2], или обработка логов распределенной информационной системы для сбора статистики и поиска аномалий в запросах пользователей. Также, примерами могут служить задачи отсечения шумов в аудиосигнале, или задачи поиска устойчивых состояний сложной динамической системы, такой как системы массового обслуживания с большим влиянием внешних факторов – типа моделей обслуживания пассажиропотоков в терминалах аэропортов.

Алгоритмы кластеризации, разработанные для больших объемов данных и способные работать с потоковыми данными, существуют [3 - 6]. Классическим примером является алгоритм кластеризации BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [7]. Однако алгоритм кластеризации BIRCH имеет ряд серьезных ограничений: неспособность выделять кластеры произвольной формы. Данный алгоритм выделяет только кластеры сферической формы, выделенные кластеры не имеют четкого радиуса, а только среднее расстояние от элементов кластера до его центра. Это связано с тем, что задачей алгоритма кластеризации BIRCH является скорее исследование плотности распределения точек данных в выборке или потоке, а не разделение входных данных на четкие кластеры.

В условиях наличия базы рабочих алгоритмов и действенных методов, естественным шагом является модификация уже существующих алгоритмов кластеризации для расширения границ их применимости в современных реалиях. Это позволяет использовать проверенные временем подходы для решения новых, более сложных задач, что в ряде случаев дает более качественные и детерминированные результаты, чем создание совершенно нового алгоритма «с нуля». Рассмотрим задачу поиска кластеров произвольной формы в условиях сильной зашумленности в данных. Быстрые, но примитивные алгоритмы, наподобие алгоритма K-средних, не способны справиться с этой задачей и зачастую требуют дополнительной информации, например, о числе кластеров. Более сложные алгоритмы иерархической кластеризации, такие как OPTICS (Ordering Points To Identify the Clustering Structure) [8], уже находят кластеры произвольной формы, но требуют значительного объема вычислений. Наиболее перспективным алгоритмом кластеризации, способным выделять кластеры произвольной формы и вошедшим в число «классических» является DBSCAN – алгоритм пространственной кластеризации на основании плотности распределения данных. Этот алгоритм обладает сравнительно небольшой вычислительной сложностью, по сравнению с другими алгоритмами кластеризации, способными выделять кластера произвольной формы [9, 10]. Представленная в данной статье модификация алгоритма DBSCAN имеет целью расширить применимость алгоритма.

В настоящее время существуют алгоритмы на основе плотности распределения данных, реализующие кластеризацию потоковых данных, такие как PreDeConStream [11], DenStream и DenStream2 [12]. Эти алгоритмы разбивают входные данные на набор микро-кластеров сферической формы, характеризующиеся положением центра, радиусом и весом, который зависит от числа элементов микро-кластера. Впоследствии

микро-кластеры объединяются в макро-кластеры, используя какой-либо алгоритм кластеризации, например, DBSCAN. При этом сами элементы микро-кластеров не сохраняются, что экономит память, но идет в ущерб точности кластеризации. Хотя подобный подход и позволяет оперировать кластерами произвольной формы, занимающими значительный объем в пространстве признаков, он не позволяет установить форму границы кластера на уровне точек данных, а только на уровне микро-кластеров, что сужает область применения, как в случае любых алгоритмов с потерей данных, что, например, делает их полностью неприменимыми для кластеризации пространств признаков малого объема.

Существуют также методы кластеризации больших объемов данных, опирающиеся на технологию параллельных вычислений. К таковым относится, например, алгоритм Stream-DBSCAN [14]. Работа указанных алгоритмов сводится к тому, чтобы разделить данные на некоторый набор партиций, например, при помощи алгоритма k-means, а затем обрабатывать партиции параллельно, при помощи более сложных алгоритмов кластеризации, таких как DBSCAN, при этом не опираясь на результаты предыдущих вычислений. Однако распараллеливание вычислений возможно не всегда, и при решении многих задач требуется память обработанных точек данных.

Не стоит забывать и об уже существующих модификациях DBSCAN для работы с потоковыми данными, одной из которых является Incremental DBSCAN. Incremental DBSCAN вносит функции «добавления» и «удаления» точек данных, тем самым организуя возможность обновления выборки, и вводит понятие «зависимых» точек данных, которые могут поменять принадлежность к кластеру при добавлении или удалении выделенной точки данных [16]. При добавлении или удалении очередной точки данных, статус «зависимых» от нее точек обновляется при помощи классического DBSCAN, не затрагивая остальную выборку. Incremental DBSCAN позволяет значительно снизить нагрузку на систему по сравнению с постоянным обновлением всех точек данных, однако требует частого перерасчета выборки, что существенно замедляет его работу. Преимущества Incremental DBSCAN реализуются в полной мере лишь при работе с кластерами, перемещающимися со временем, что не всегда соответствует решаемой задаче.

Другой известной модификацией является ST-DBSCAN, нацеленный на поиск кластеров, распределенных как в пространстве, так и во времени [17]. Такой поиск подразумевает использование двух мер близости точек данных – близость по расстоянию в пространстве признаков, и близость по прошедшему времени. Помимо этого, алгоритм ST-DBSCAN вводит понятие фактора плотности (density factor) для лучшего отделения информационного шума, и параметр порогового значения по плотности, предотвращающий объединение близко расположенных кластеров в один. Однако алгоритм ST-DBSCAN не уменьшает вычислительную сложность по сравнению с классическим алгоритмом DBSCAN, что является серьезным ограничением при работе с потоком данных.

Еще одной значимой модификацией DBSCAN является RT-DBSCAN [18], активно использующий параллельные вычисления и предполагающий разделение пространства признаков на области, внутри которых алгоритм параллельно проводит кластеризацию. Разделение происходит динамически, на основании расположения уже размеченных точек данных, то есть опирается на результаты предыдущих вычислений. Однако, распараллеливание вычислений возможно не всегда, а при обработке одним процессом выигрыш по быстродействию RT-DBSCAN перед классическим DBSCAN сходит на нет.

Классический алгоритм DBSCAN и особенности его применения

Классический алгоритм DBSCAN

Алгоритм DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [20] предназначен для группирования информационных объектов на основании их плотности распределения в пространстве признаков. Подразумевается, что каждый объект описывается типовым набором признаков, выраженных численно. Совокупность таких признаков принято рассматривать как некоторое пространство признаков, в котором информационные объекты располагаются относительно друг друга, наподобие точек в многомерной системе координат. В дальнейшем мы будем называть объекты, подлежащие кластеризации, точками или точками данных.

Одним из ключевых понятий данного алгоритма является расстояние между точками. В задаче кластеризации, расстояние является мерой «родства» или «подобия», определяющей, насколько точки данных похожи друг на друга. Способы вычисления расстояния могут разниться от задачи к задаче, но как правило, используется Евклидова метрика расстояния.

Классическая реализация алгоритма DBSCAN (рис. 1) имеет два параметра: радиус окрестности точек данных (обозначаемый ***EPS***) и минимальное число соседей (обозначенный ***MinPts***). При проведении кластеризации, для каждой точки данных составляется *Список EPS-соседей* и проводится следующий ряд проверок:

1. Если число соседей, находящихся в радиусе окрестности точки, то есть входящих в *Список EPS-соседей,* больше или равно ***MinPts***, точка помечается как корневая и получает метку кластера в соответствии с логикой, показанной на рис.1. Таким образом расширяются уже существующие кластеры и закладывается основа для следующего кластера. При этом фактическое число кластеров определяется во время исполнения алгоритма и задание числа кластеров на этапе настройки алгоритма не требуется. Данное свойство позволяет использовать алгоритм DBSCAN в условиях отсутствия предварительной информации об организации данных.
2. Если число соседей точки в *Списке EPS-соседей*, меньше ***MinPts***, но среди соседей есть корневая точка, искомая точка помечается как граничная и в случае конфликта между корневыми точками разных кластеров, граничная точка будет принадлежать тому кластеру, который «захватит» ее первым. Такие точки обозначают границу кластера, то есть область, за пределами которой, плотность данных слишком мала, чтобы дать точный ответ о принадлежности к какому-либо конкретному кластеру. В некоторых реализациях DBSCAN для снижения вычислительной нагрузки не производится поиск граничных точек, и такие точки помечаются как шумовые. Однако, такой подход является тупиковым для модификации алгоритма под потоковые данные.
3. Если число соседей точки в *Списке EPS-соседей*, меньше ***MinPts*** и среди соседей нет корневых точек, точка помечается как шумовая. Однако, если дальнейший анализ выявит среди соседей шумовой точки корневую точку, эта шумовая точка будет помечена заново, как граничная.

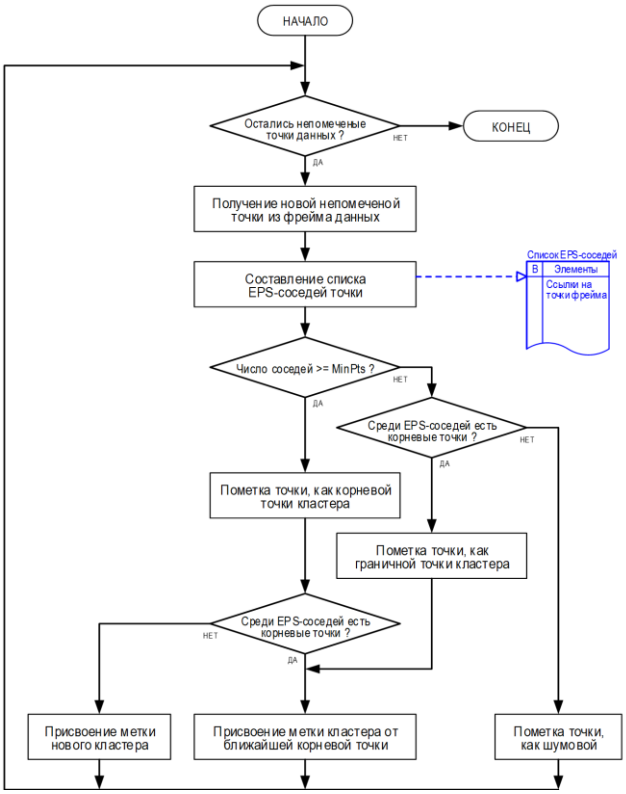

Рисунок 1. Классический алгоритм DBSCAN

Так продолжается до тех пор, пока не будут исследованы все точки данных. В зависимости от особенностей конкретной реализации, алгоритм может быть, как однопроходным, так и многопроходным.

Этот алгоритм в явном виде не опирается на математические законы и не требует сведений о числе кластеров перед началом работы. С его помощью можно решать широкий круг прикладных задач, таких как поиск аномалий и выбросов, фильтрации шумов в аудио-видео системах, и вплоть до поиска формализуемых правил группировки данных в многомерных данных геоинформационных систем (city traffic control, social dynamic awareness, cargo logistics).

Проблемы применения классического алгоритма DBSCAN

Несмотря на множество достоинств, описанный выше классический алгоритм DBSCAN не предназначен для обработки потока данных или объемных данных, предполагающих страничную организацию, либо какой-то другой способ обработки «по частям». Алгоритм разрабатывался для кластеризации точек данных, распределенных по заранее не определенному пространству признаков, но не предполагалось, что сами данные

будут обновляться, например, дополняясь новыми точками в процессе работы. Кластеризация при помощи DBSCAN требует держать в памяти все элементы выборки, что невозможно в системах, работающих с условно бесконечными потоковыми данными. Также стоит принять во внимание, что вычислительная сложность классического DBSCAN равна $O(n^2)$ где n – число элементов выборки.

Ниже приводится комплексное решение означенных проблем использования алгоритма кластеризации DBSCAN для работы с потоковыми данными и пути улучшения его эффективности при работе с большими объемами данных.

Подход к модификации классического алгоритма DBSCAN

Решением проблемы неэффективного использования памяти классическим алгоритмом DBSCAN при работе с потоковыми данными является такая модификация алгоритма, при которой каждый кластер будет сохранять только минимальную совокупность данных, необходимую и достаточную для проверки вхождения очередной пришедшей точки данных в этот кластер. Примером подобной совокупности является описание границы кластера, поддерживающее возможность его обновления: тогда вопрос о вхождении каждой новой точки данных в кластер сводится к вопросу о нахождении точки со внутренней или внешней стороны границы.

Подобный подход потенциально дает возможность работать с неограниченным потоком данных, увеличивая скорость работы алгоритма за счет снижения объема обрабатываемых данных, в идеале приближая вычислительную сложность алгоритма к линейной сложности. В частности, это один из основных принципов, на которые опирается алгоритм кластеризации BIRCH, упомянутый во введении к этой статье. [9]

Детально описанные в следующих разделах модификации используют для обработки данных алгоритм DBSCAN и его производные, относящиеся к алгоритмам **машинного обучения**, а также **специализированные математические методы**, такие как алгоритм вычисления выпуклой оболочки (Quickhull). Таким образом, означенные модификации следует считать **гибридными алгоритмами**.

Модификация, обозаченная далее MOD1 является развитием алгоритма DBSCAN и активно использует особенности данного алгоритма кластеризации. В частности, MOD1 использует те же принципы при добавлении новых точек данных в существующие кластеры и обновлении границ (см. рис.1, рис.5-7). Однако, MOD1 имеет ряд специфических ограничений, связанных с распределением данных, о которых подрбоно рассказано ниже. Для смягчения этих ограничений, гибридный DBSCAN был дополнен модификацией, далее обозначенной MOD2. MOD2 применяется только в случаях упомянутых ограничений, так как он является существенно менее эффективным с точки зрения объема вычислений и точности кластеризации. MOD2 опирается на математический метод вычисления выпуклой оболочки (Quickhull), но сохраняет часть ключевых принципов работы алгоритма машинного обучения DBSCAN.

Совокупность этих модификаций делает гибридный DBSCAN способным решать широкий круг ранее недоступных ему задач: от обработки непрерывных сигналов, до поиска устойчивых состояний динамических систем и работы с большими объемами многомерных данных геоинформационных систем.

Реализация модифицированного алгоритма DBSCAN

Ниже приведены модификации алгортма DBSCAN, призванные обеспечить возможность работы с потоковыми данными. Работа каждой модификации сводится к выявлению границы кластера и распознаванию, находятся ли точки «внутри» или «снаружи» этой границы. Модификации идут по тому же пути, что и класический DBSCAN, и используют процедуры поиска соседей и составления временного списка соседей, манипуляции с которыми позволят упростить ряд действий, связанных с обновлением границ кластера.

Модификация, далее обозначенная MOD1, наиболее эффективная из представленных модификаций, и позволяет сохранить возможность работы с кластерами произвольной формы. Для построения полной границы кластера MOD1 использует только существующие корневые и граничные элементы кластера. Однако, при наличии разрывов в расположении граничных элементов, эта модификация не может быть применена, так как кластер с незамкнутой границей признается бесконечно большим или бесконечно малым, что является ошибкой кластеризации. Соответственно, необходимо ввести процедуру проверки замкнутости границы кластера.

Кроме того, стоит проверить, является ли один кластер частью другого, так как пересечение границ означает что одна из них определена некорректно. Один кластер не может быть частью другого, за исключением граничных элементов, которые кластеры могут «перетягивать» друг у друга. Для признания границы непригодной достаточно доказать, что, согласно алгоритму проверки вхождения точки в кластер, хотябы один элемент чужого кластера находится «внутри» этой границы.

Модификация, далее обозначенная MOD2, гарантирует генерацию искусственной замкнутой границы кластера независимо от расположения корневых или граничных элементов, и удовлетворяет потребность в сохранении минимального числа точек данных при определении границы кластера, при этом сохраняя возможность работы с кластерами различных форм. MOD2 использует только корневые элементы исходного

кластера, однако, граница, которую строит эта модификация, всегда будет выпуклой, что снижает точность кластеризации в случае кластеров произвольной формы.

Каждая модификация имеет блок проверки вхождения новой точки в соответствующий кластер, который и проводит кластеризацию данных; блок обновления границы кластера, который необходим для поддержания актуальной формы границы кластера при обработке новых входных данных; блок нахождения границы кластера из набора элементов кластера.

Различия между модификациями предписывают следующий подход: при нахождении множества точек кластера с помощью алгоритма DBSCAN, по умолчанию нужно построить границу кластера по алгоритму, описанному в MOD1, и пометить его как подлежащий обработке по соответствующему алгоритму MOD1, затем граница проходит проверку на замкнутость и если она оказывается не замкнута, то пересчитывается согласно алгоритму, описанному в MOD2 и сам кластер, помечается для обработки MOD2.

Модификация MOD1 лучше сохраняет сильные стороны классического алгоритма DBSCAN, а именно, работу с кластерами произвольной формы, поэтому ее использование в анализе является предпочтительным. В то же время, MOD2 создает границу при любых входных данных и с меньшим числом элементов, но при этом теряется точность кластеризации, так как выпуклая граница, сгенерированная MOD2 может не отвечать реальной форме кластера.

Общий вид гибридного DBSCAN с использованием всех модификаций показан на рис.2. Блоки этого алгоритма детально описаны и проиллюстрированы в нижеследующих разделах. Блоки с двойными границами обозначают алгоритмы, взятые из сторонних источников, а не разработанными в рамках данной статьи.

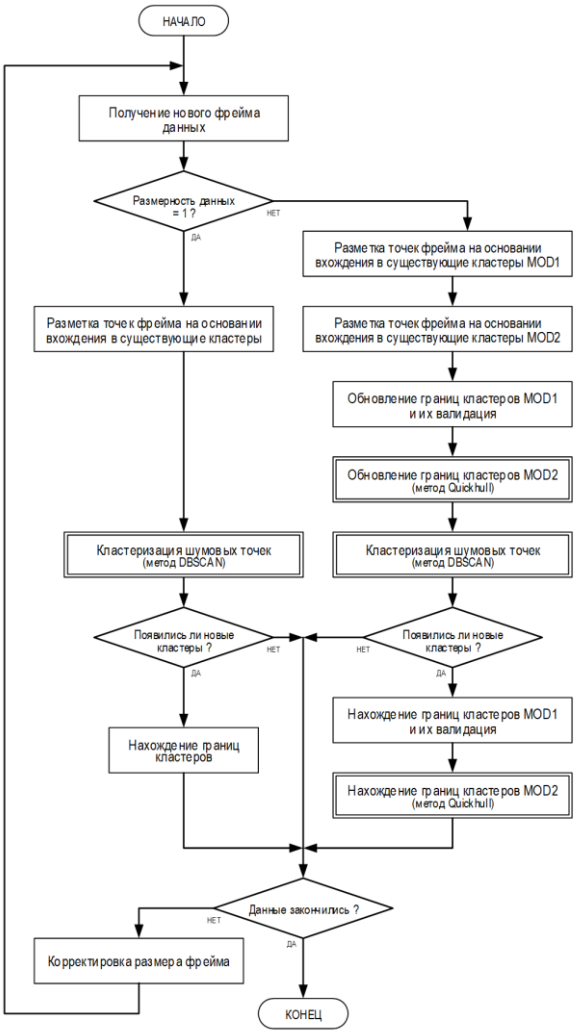

Рисунок 2. Общий алгоритм гибридного DBSCAN

Работа с потоковыми данными означает, что в любой момент времени нет конечной выборки данных. Поэтому, необходимо ввести понятие *фрейма данных*, разбивающего бесконечный поток на наборы точек

данных, ограниченные по какому-либо признаку, связанному с особенностями передачи данных – по фиксированному временному отрезку, либо по фиксированному числу точек. Размер фрейма модет быть адаптивным, то есть подстраиваемым, в зависимости от каких-либо условий.

Адаптивный размер фрейма позволяет также уравнять скорость работы модификаций в условиях значительной неравномерности информационного потока по плотности и информативности входных данных. Предполагается, что фреймы меньших размеров обрабатываются быстрее, в то время как во фреймах больших размеров выше вероятность встретить информативную последовательность данных. Как уже было сказано, вычислительная сложность алгороитма DBSCAN в худшем случае составляет $O(n^2)$, где n – число элементов выборки. Таким образом, чем меньше точек обрабатывается за один фрейм, тем быстрее будет идти обработка в целом. Размер фрейма должен быть ограничен максимальным и минимальным числом точек. Минимальное число точек должно быть больше ***MinPts***, иначе DBSCAN не сможет распознать ни одного кластера.

Одна из ключевых особенностей работы с фреймами, в отличие от полной выборки данных, заключается в том, что точки из отдельно взятых фреймов могут не составлять кластеров, то есть определить число и примерную форму границ кластеров будет невозможно. Поребуется итеративный подход, включающий в себя проверку новых данных на наличие в них кластеров. В предложенных модификациях эта проблема решена следующим образом: вначале точки данных из нового фрейма проходят проерку на вхождение в каждый из уже существующих кластеров, затем точки, не вошедшие ни в один кластер, проходят кластеризацию методом DBSCAN. Для найденных кластеров вычисляется граница, «лишние» элементы кластеров удаляются, а точки, по итогам кластеризации не вошедшие ни в один кластер, помечаются как шумовые.

Заметим, что при обработке первого фрейма кластеров еще не существует, поэтому работа начинается с первичной кластеризации и определения границ первого набора кластеров.

Размерность данных 1D

Отдельно стоит рассмотреть случай кластеризации данных в одномерном пространстве признаков. Подобная задача встречается при анализе простых временных рядов – например, для предсказания динамики процесса или отсечения шума при получении сигнала от датчика.

Граница кластера для одномерного случая, представленная *Списком Элементов Границы 1D*, являет собой два пороговых значения – максимальную и минимальную координаты, между которыми располагаются элементы кластера.

Значительная часть реальных задач так или иначе связана с обработкой одномерных данных – так, большинство датчиков передают информацию в формате одномерного сигнала. Двумерных по своей природе данных очень мало: это могут быть геоданные, либо графическое изображение. Данные размерности более 3, как правило, являются комплексными данными от систем реального времени, например, наборами одномерных показаний с датчиков, скомпонованными по времени. В большинстве случаев их можно разбить на комплес одномерных данных для анализа. Однако, возможность обработки многомерных комплексных данных может оказаться крайне эффективна в определенных случаях, связанных с особенностями какой-либо конкретной прикладной системы.

Работа с одномерными кластерами разделяется на два этапа: инициализация границы и обработка фрейма, совмещенная с обновлением границ. Инициализация устанавливает базовые границы на основании имеющихся сведений о кластере, обработка уточняет границы кластера и выясняет принадлежность к тому или иному кластеру.

Алгоритм обработки фрейма одномерных данных изображен на рис.3. Он представляет собой перебор точек фрейма с последовательной проверкой вхождения точки фрейма в интервал между максимальным и минимальным элементами *Списка Элементов Границы 1D* – то есть вхождения в кластер, и проверкой близости точки фрейма к границам – то есть обновления границ кластера.

Алгоритм нахождения границы кластера в одномерном пространстве признаков показан на рис.4. Он сводится к поиску минимального и максимального элементов кластера с последующим сохранением в *Список Элементов Границы 1D*. Как нетрудно заметить, вычислительная сложность каждого из этапов составляет $O(n)$, где n – число точек, подлежащих обработке: в первом случае – число элементов кластера, во втором – число точек фрейма.

MOD1: на основе граничных элементов

Рассмотрим модификацию MOD1. Как было сказано выше, можно выделить три логических блока: блок проверки вхождения точки данных в кластер, блок обновления границы кластера и блок нахождения границы кластера. Действия, описанные в эти блоках, выполняются для каждого кластера отдельно.

Работа блока проверки вхождения точек фрейма в кластер, изображена на рис.5. Ключевые элементы данных – корневые и граничные точки кластера хранятся в *Списке Элементов Границы MOD1*.

Работа блока предполагает перебор всех точек фрейма с проверкой нахождения каждой точки с внешней или внутренней стороны границы. Далее находится расстояние до корневых и граничных точек из *Списка Элементов Границы MOD1* чтобы отделить новые корневые точки, лежащие внутри кластера от новых корневых точек, лежащих на границе кластера. Это позволяет также отделить новые граничные точки от шумовых точек.

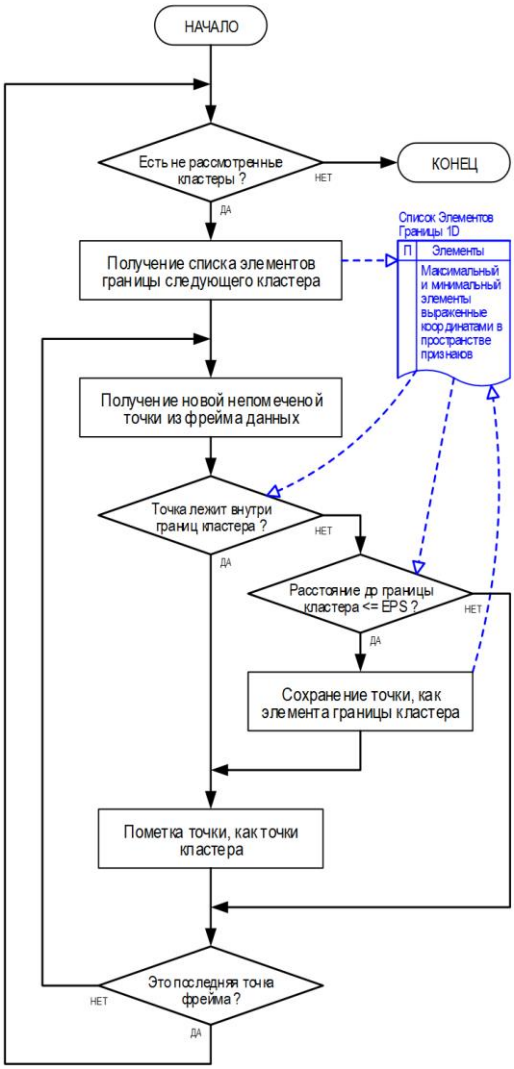

Рисунок 3. Разметка точек фрейма

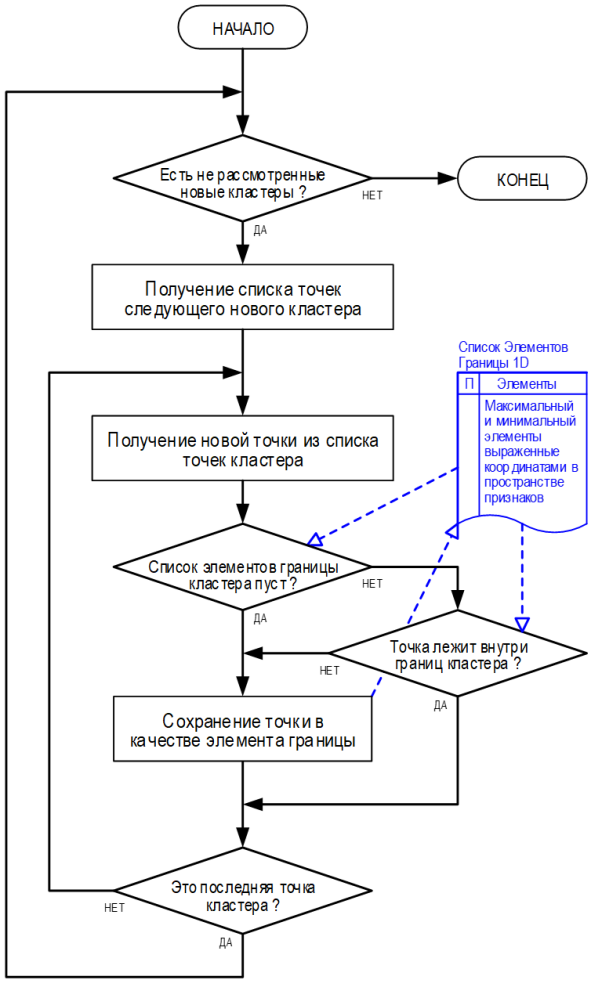

Рисунок 4. Нахождение границ одномерных кластеров на основании вхождения в кластеры

Граничные точки находятся в местах соприкосновения плотно упакованных кластеров с пространством, где плотность данных слишком низка. Это позволяет с высокой долей уверенности утверждать, что они находятся на границе кластера. В то же время, корневые точки кластера в большинстве своем находятся внутри кластера, где плотность данных достаточно высока, а не на границе, но те из них, которые имеют среди своих соседей граничные точки, позволяют добавлять в кластер новые граничные точки, тем самым обновляя границу кластера по мере поступления новых данных.

Как указывалось, выше, модификации алгоритма способны работать с данными разной размерности (***dim***). В связи с этим возникает задача определения ***dim***-мерной точки внтури ***dim***-мерной замкнутой фигуры произвольной формы. Обычно для каждого отдельного ***dim***-мерного случая требуется создание своего решения. Для упрощения алгоритма здесь применен нетривиальный подход к проверке нахождения точки внутри или снаружи границы кластера не зависящий от размерности пространства признаков и использующий специфику алгоритма DBSCAN.

Основная идея данного подхода состоит в том, что элементы границы кластера составляют как бы многослойную границу. То есть, можно явно различить внешний и внутренний слои такой границы. Проверка

нахождения точки внутри произвольной замкнутой фигуры предполагает нахождение ближайшего к ней элемента из *Списка Элементов Границы MOD1* и определение, относится ли он к корневым либо граничным. В первом случае искомая точка считается лежащий внутри границы, во втором – снаружи. Такой подход не привязан к конкретной размерности данных, а только к метрике расстояния. Также подход позволяет работать с фигурой произвольной формы любой размерности, заданной простым набором точек, то есть таблично, а не аналитически.

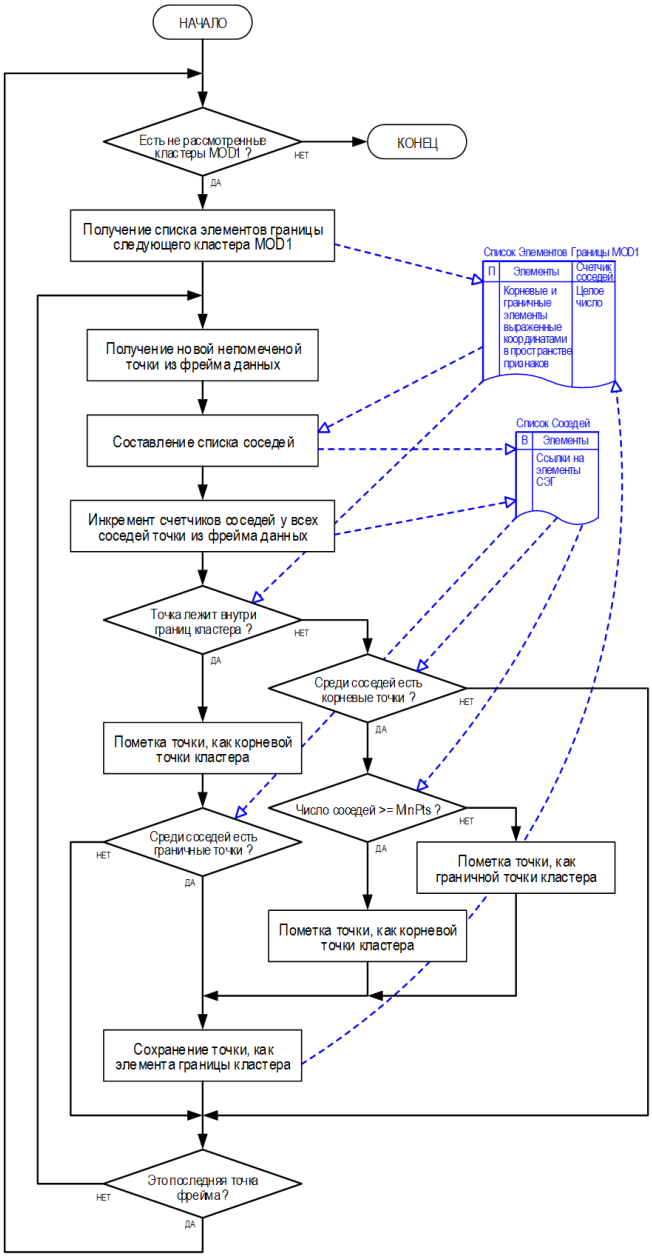

Рисунок 5. Разметка точек фрейма на основании вхождения в существующие кластеры MOD1

В результате работы блока проверки вхождения точек фрейма в кластер (см. рис.5), *Список Элементов Границы MOD1* дополняется новыми корневыми и граничными точками, опираясь на границу кластера, актуальную на момент получения фрейма данных. Однако, при этом не осуществляется реклассификация элементов списка после изменения границы. Этим занимается следующий блок алгоритма – блок обновления границы кластера, представленный на рис.6.

Обновление границы для MOD1 заключается в реклассификации граничных точек, для которых число элементов *Списка Соседей* достигло или превысило ***MinPts*** с переводом их в разряд корневых точек. А также, в удалении из *Списка Элементов Границы MOD1* корневых точек, у которых в *Списке Соседей* больше нет граничных точек, так как все соседние граничные точки были переведены в корневые. Последним этапом

следует валидация, которая ищет нарушения целостности границы кластера после ее обновления.

Следующий блок алгоритма – блок нахождения границы кластера, приведенный на рис.7, получает на вход точки данных, прошедшие кластеризацию алгоритмом DBSCAN в виде *Списка Элементов Кластера*. Задачей этого блока является занесение новых точек в *Список Элементов Границы MOD1*, где каждая точка представлена набором координат в пространстве признаков и счетчиком соседей точки. Сюда попадают точки следующих типов: граничные точки кластеров, с указанием числа *EPS-соседей* каждой граничной точки в виде единственного целого числа и корневые точки кластеров, имеющие в своей окрестности граничные точки.

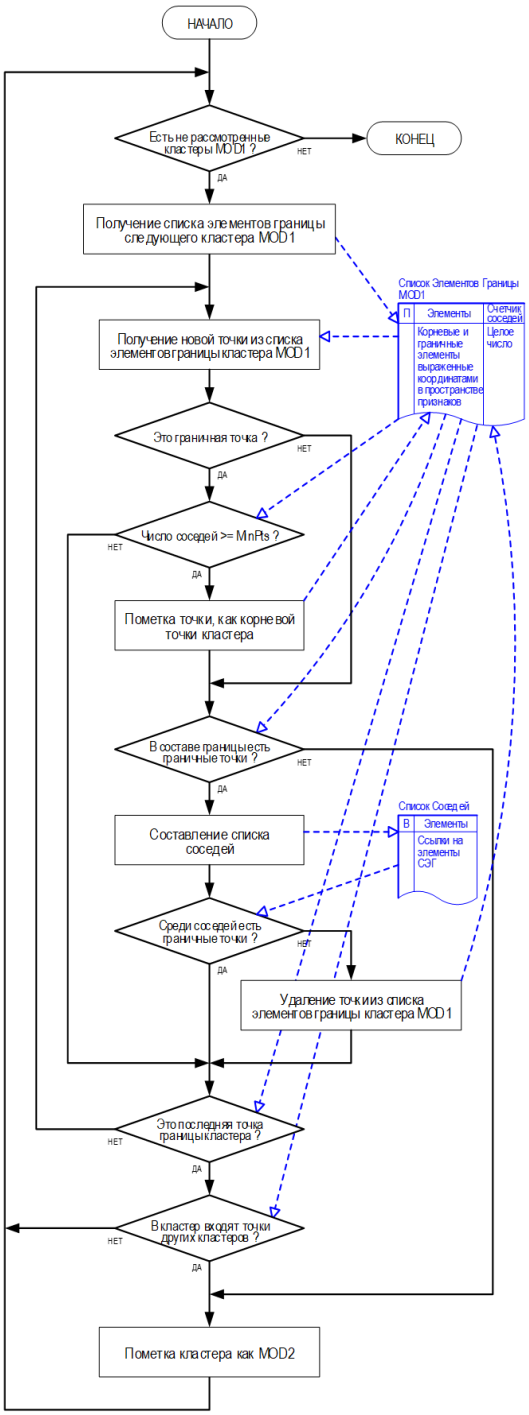

Рисунок 6. Обновление границ кластера MOD1

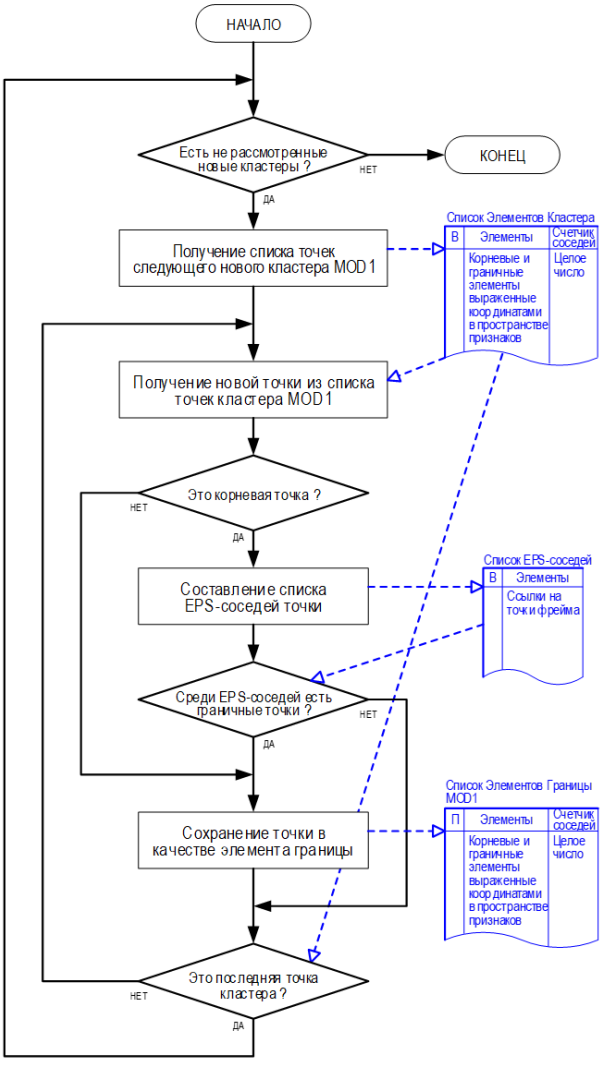

Рисунок 7. Нахождение границы кластера MOD1

Очевидно, что общее количество элементов кластера больше или равно количеству элементов границы. Экономия в числе точек данных, которые необходимо держать в памяти, по сравнению с классическим алгоритмом DBSCAN, тем выше, чем больше элементов кластера не включается в состав границы. То есть, чем

ближе форма кластера к ***dim***-мерному аналогу сферы, где ***dim*** – размерность пространства признаков, и чем более объемной является область в пространстве признаков, которую занимает кластер, тем меньше данных сохраняется для описания границы.

Работоспособность модификации MOD1 в целом, зависит от истинности следующих предположений:

1. Вокруг каждого кластера располагается достаточное количество граничных точек для формирования полноценной границы кластера. Невыполнение этого условия делает модификацию MOD1 неработоспособной. Примерами невыполнения этого условия являются ситуации, когда кластер содержит только корневые точки, или когда расположение граничных точек не позволяет описать замкнутую фигуру, повторяющую форму кластера.
2. Новые данные поступают таким образом, что предположение 1 никогда не нарушается. Невыполнение этого условия также делает модификацию MOD1 неработоспособной. Примером является такое поступление данных, при котором граничные элементы кластера превращаются в корневые элементы, среди соседей которых нет других граничных элементов.

Невыполнение перечисленных выше требований в определенных выделенных случаях делает модификацию MOD1 неработоспособной. Для решения подобных проблем есть несколько подходов. Например, можно воспользоваться методами генерации искусственных точек выборки, чтобы заполнить границу кластера искусственными граничными значениями посредством добавления точек через равные промежутки, превышающие ***EPS***. Такие искусственные значения следует дополнительно пометить как служебные, чтобы впоследствии исключить из процесса анализа. Однако при этом необходимо помнить, что генерация искусственных значений не всегда является допустимой мерой. Также необходимо иметь ввиду, что, в зависимости от принципов работы, генерация искусственных точек данных может приводить к ошибкам кластеризации: например, может быть сгенерирован искусственный кластер, описывающий особенности, не харатерные для реальных данных.

Также можно воспользоваться алгоритмами аппроксимации полигональной кривой, такими как алгоритм Рамьера-Дугласа-Пекера, но необходимо помнить, что подобные меры могут привести к снижению точности кластеризации, ввиду того что задача, которую решают подобные алгоритмы, не коррелирует с задачами кластерного анализа. Например, алгоритмы аппроксимации полигональной кривой могут «выбрасывать» точки, принадлежащие границе кластера, из кластера, так как в логике алгоритмов аппроксимации не прописана проверка на вхождение в фигуру.

Несмотря на упомянутые ограничения, модификация MOD1 способна работать в условиях, когда в набор данных постоянно добавляются новые точки, и поддерживает работу с кластерами произвольной формы, как и классический DBSCAN. При возникновении перечисленных выше ограничений, предлагается решение, представленное модификацией MOD2, описанное ниже, как наиболее оптимальное.

MOD2: для выпуклого кластера без учета граничных элементов

Модификация MOD2 представлена на рис.8. Она не способна работать с кластерами любой формы и разработана как дополнение к MOD1 для устранения ограничений, описанных выше.

Если известно, что для текущего набора данных предположение о том, что граничные элементы составляют полноценную границу кластера, не выполняется, для нахождения границы кластера следует использовать метод, работающий только с корневыми элементами кластера. Для этого можно использовать алгоритмы построения выпуклого корпуса (convex hull), например, алгоритм Quickhull [21] для многомерного случая.

В модификации MOD2 граница кластера представлена совокупностью вершин, сторон и граней выпуклой фигуры размерности ***dim***. Здесь под стороной понимается ориентированная гиперплоскость, то есть поверхность размерности ***dim-1***, расстояние до которой может быть, как положительным, так и отрицательным. Последнее означает, что точка находится с внутренней стороны гиперплоскости. Под гранью понимается место пересечения двух сторон, то есть поверхность размерности ***dim-2***. Алгоритм Quickhull работает с данными размерности 2 и выше [21].

При поступлении нового фрейма данных, проверяется нахождение каждой точки фрейма внутри выпуклой фигуры путем вычисления расстояния от точки до каждой стороны из *Списка Элементов Границы MOD2*. Если точка находится внутри фигуры, но не является частью границы, то есть расстояние от нее до любой из сторон меньше либо равно нулю, точка помечается как корневой элемент кластера, но не сохраняется в *Список Элементов Границы MOD2*. Если расстояние от точки до одной из сторон границы кластера больше нуля, но меньше или равно ***EPS***, точка является частью границы и включается в список внешних точек соответствующей стороны *Списка Элементов Границы MOD2*.

Описанный алгоритм добавления новых точек данных в *Список Элементов Границы MOD2* опирается на следующие предположения:

1. Граница кластера описывается выпуклой фигурой с достаточной точностью. Невыполнение этого условия может привести к включению в кластер посторонних (шумовых) точек данных, находящихся близко от границ фигуры – например, в тех местах, где реальная граница кластера «окружает» шумовые данные.

2. Плотность кластера такова что соседство точки со стороной границы кластера означает, что точка также является корневой. Невыполнение этого условия ведет к включению в границу кластера посторонних объектов.

Построение и обновление выпуклого контура кластера, с использованием принципов алгоритма Quickhull (см. рис.8) выглядят следующим образом:

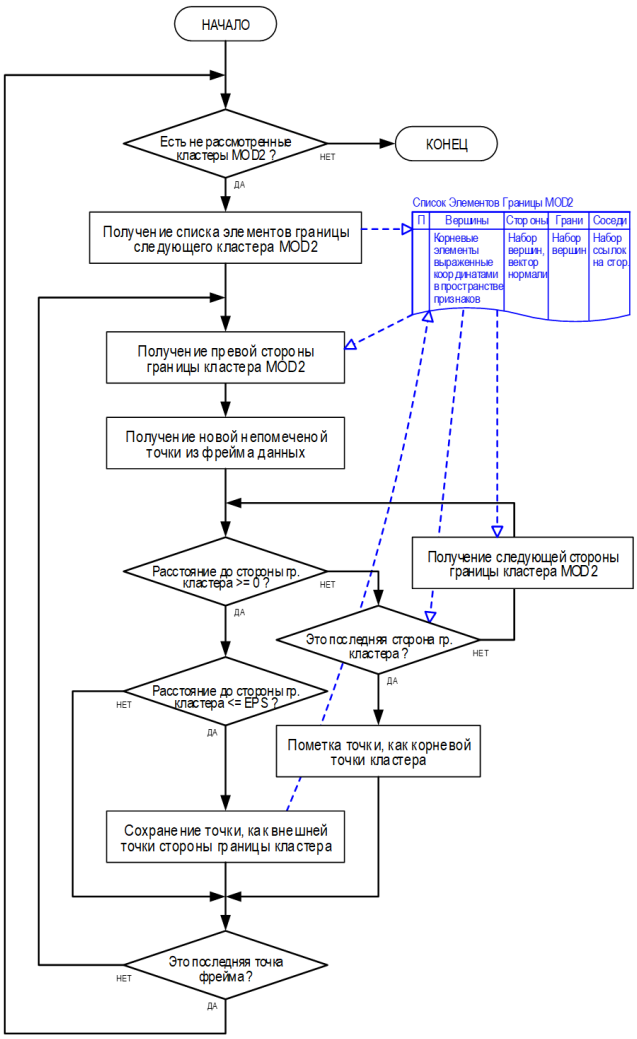

Рисунок 8. Разметка точек фрейма на основании вхождения в существующие кластеры MOD2

1. Если выпуклая фигура, описыающая границу кластера не задана, то есть в *Списке Элеменнтов Границы MOD2* нет информации о сторонах такой фигуры, тогда необходимо по ***dim+1*** наиболее удаленным друг от друга экстремальным корневым точкам кластера построить симплекс – ***dim***-мерную фигуру с минимальным числом вершин. По итогам этой операции по каждым ***dim*** из ***dim+1*** выделенных точек строится сторона – плоскость, ориентированная так, что геометрический центр кластера находится со внутренней стороны. То есть, расстояние до него будет выражено отрицательным числом. Также, для каждой стороны должен быть определен список соседних сторон, то есть тех, с которыми есть общие грани. Эта информация заносится в *Список Элементов Границы MOD2.*

2. Для каждой построенной стороны определяется набор внешних точек – точек кластера, лежащих «снаружи» соответствующей стороны. То есть, расстояние от которых до ориентированной плоскости будет выражаться положительным числом. При этом точки, входящие в список внешних точек одной стороны не рассматриваются для всех остальных сторон. Точки, не признанные внешними ни для одной из сторон, считаются корневыми точками кластера, не входящими в границу, и далее не рассматриваются.

3. Для каждой стороны с не пустым списком внешних точек находится самая удаленная от нее внешняя

точка ***P*** и соседние стороны, для которых точка ***P*** также является внешней. Стороны, для которых точка ***P*** является внешней, добавляются в *Список на удаление*. Затем находятся грани, внешние по отношению к сторонам из списка на удаление, и заносятся во временный *Список внешних граней*. От каждой такой грани до точки ***P*** строится новая сторона. Далее стороны из *Списка на удаление* удаляются, а для каждой новой стороны составляется список соседних сторон и находятся внешние точки. Алгоритм завершается, когда список внешних точек для всех сторон кластера будет пуст.

Таким образом, имея набор корневых точек кластера, можно найти среди них точки, описывающие выпуклую границу кластера. Фактически, подход предлагает построение выпуклого многомерного многогранника, который и представляет собой границу кластера с достаточной степенью приближения. Этим можно существенно уменьшить объем данных, хранимый в памяти, для корректной кластеризации.

Модификация MOD1 предполагает, что элементы границы кластера должны располагаться по всему периметру кластера наподобие замкнутой кривой, в то время как MOD2 предполагает выбирать элементы границы кластера так, чтобы они обозначали лишь вершины выпуклого многоугольника.

Обновление границы кластера (см. рис.9), по большей части, повторяет построение базовой границы кластера. Разница заключается в том, что при обновлении границы расчет ведется лишь для элементов, входящих в границу кластера, а не для всех элементов кластера. А также в том, что базовый набор сторон уже построен, а внешние точки известны, поэтому алгоритм начинает работу с обработки сторон, у которых есть внешние точки. По итогам обновления, вершины и стороны фигуры, находящиеся с внутренней стороны границы, будут удалены как избыточные.

Можно сказать, что инструментарий, на котором построен MOD2 более универсален, чем инструментарий MOD1. Так, например, алгоритм проверки вхождения точки внутрь фигуры, описывающей границу кластера, можно использовать без изменений в задачах, связанных с многомерными геометрическими фигурами. В то же время аналогичный алгоритм для MOD1 можно использовать лишь тогда, когда граница фигуры имеет два ярко выраженных слоя: внутренний и внешний, однако при выполнении этого специфического условия, модификация MOD1 может работать с невыпуклыми кластерами, что важнее с точки зрения качества кластеризации.

Расчет вычислительной сложности

Далее приведены ***O***-оценки временной сложности для каждой из модификаций.

Анализ вычислительной сложности для одномерного потока данных привел к следующим оценкам:

Вычислительная сложность *инициализации границы* кластера составляет

$$O(cpn), \quad (1)$$

где ***cpn*** – число элементов нового кластера.

Вычислительная сложность *обработки точек одного фрейма* составляет

$$O(fp), \quad (2)$$

где ***fp*** – число точек данных во фрейме.

Опираясь на общий алгоритм гибридного DBSCAN (см. рис. 2) и с учетом (1) и (2), получаем итоговую вычислительную сложность:

$$O(fp + (fp\text{-}fpo)^2 + cpn), \quad (3)$$

где ***fp*** – число точек данных во фрейме, ***fpo*** – число элементов фрейма, принадлежащих к кластерам, определенным в предыдущих фреймах, ***cpn*** – число элеметов фрейма, образующих новые кластеры, появившиеся после поступления фрейма на обработку. По мере увеличения числа элементов, вошедших в один из ранее существовавших кластеров (число ***fpo***), вычислительная сложность обработки каждого нового фрейма будет снижаться, стремясь к линейной.

Как видно из алгоритма, изображенного на рис.2, обработка одномерных потоков является вырожденным случаем и на порядок проще (легче), чем многомерные модификации. Поэтому, интерес с точки зрения детальной оценки вычислительной сложности и расхода памяти, представляет обработка многомерных потоков данных. Далее рассмотрим подробнее MOD1 и MOD2 для ***dim >= 2***.

Анализ вычислительной сложности для многомерного потока данных в MOD1 привел к следующему:

Вычислительная сложность *проверки нахождения одной точки внутри границ кластеров* составляет

$$O(bpt), \quad (4)$$

где ***bpt*** – суммарное число элементов границ всех кластеров.

Вычислительная сложность *обработки точек всего фрейма для всех существующих кластеров* составляет

$$O(bpt*fp), \quad (5)$$

где ***fp*** – число точек данных во фрейме.

Фактически, большую часть вычислительной сложности этой операции составляет проверка нахождения точки данных внутри кластера. Подразумевается, что эта опреация будет выполняться часто, соответсвенно скорость ее исполнения оказывает серьезное влияние на скорость анализа в целом.

Вычислительная сложность операции *обновления кластера* составляет

$$O(bpt^2 + bpt*MinPts), \quad (6)$$

что можно округлить до

$$O(bpt^2) \quad (7)$$

Эта операция считается вычислительно сложной. В целях ускорения анализа можно регулировать частоту обновления кластера, однако это может привести к пометке весомых элементов кластера как шумовых элементов.

Вычислительная сложность операции *отыскания границы кластера*, полученной при проходе алгоритма DBSCAN, для каждого кластера составляет

$$O(cpn^2), \quad (8)$$

где ***cpn*** – число элементов нового кластера.

Таким образом, опираясь на общий алгоритм гибридного DBSCAN (см. рис. 2) и с учетом вышесказанного (4-8), итоговая *вычислительная сложность обработки фрейма* данных оценивается как

$$O(bpt^2), \quad (9)$$

где ***bpt*** – число элементов, включенных в границы всех кластеров, определенных до получения текущего фрейма.

Анализ вычислительной сложности MOD2 привел к следующим оценкам:

Вычислительная сложность *проверки нахождения одной точки внутри границ кластеров* составляет

$$O(n), \quad (10)$$

где ***n*** – суммарное число сторон фигур, описывающих границы всех кластеров.

Алгоритм *генерации границы кластера* практически полностью повторяет алгоритм поиска выпуклой оболочки Quickhull. Согласно первоисточнику [21], вычислительная сложность такого алгоритма соствляет

$$O(cpn*f/r), \quad (11)$$

где ***cpn*** – количество элементов нового кластера, ***r*** – число вершин итоговой фигуры, описывающей границу, ***f*** – максимально число сторон, которые могут пройти через ***r*** точек, которая определяется как

$$f = \frac{r^{\dim/2}}{(\dim/2)!}, \quad (12)$$

где ***dim*** – размерность пространства признаков. Таким образом итоговая вычислительная сложность в худшем случае получится равной

$$O\left(\frac{r^{(\dim/2)-1}}{(\dim/2)!}\right) \quad (13)$$

Эта операция выполняется при появлении каждого нового кластера, что серьезно влияет на работу MOD2.

Обновление границы кластера использует урезанную версиию алгоритма Quickhull.

Вычислительная сложность в этом случае составляет

$$O\left(\frac{cvn^{(\dim/2)-1}}{(\dim/2)!}\right), \quad (14)$$

где ***cvn*** – число вершин, добавившихся к существующим вершинам кластера после обновления. Фактически, вычислительная сложность этого этапа зависит от числа внешних точек, обнаруженных при обновлении кластера и, в идеале, при отсутствии таковых, стремится к нулю.

Вычислительная сложность операции *добавления элементов* составляет

$$O(n*fp), \quad (15)$$

где ***n*** – суммарное число сторон выпуклых многомерных фигур, ***fp*** – число точек фрейма.

Таким образом, *итоговая вычислительная сложность* обработки следующего фрейма в худшем случае оценивается как

$$O\left(\frac{cvn^{(\dim/2)-1}}{(\dim/2)!} + \frac{r^{(\dim/2)-1}}{(\dim/2)!}\right), \quad (16)$$

где ***cvn*** – число вершин, добавившихся к существующим вершинам кластера вследствии обновления, ***dim*** – размерность пространсва признаков, ***cpn*** – число элементов фрейма, включенных в новые кластеры, ***r*** – число

вершин фигуры, описывающей границу кластера, определенного после получения фрейма данных.

Проследим динамику вычислительной сложности каждой модификации. Для этого воспользуемся искуственно сгенерированной выборкой данных, для которой гарантируется создание замкнутого контура при использовании MOD1. Точки подаются на вход каждой модификации одинаковыми фреймами, каждый из которых содержит 100 точек.

В настройках алгоритма DBSCAN установлены значения ***EPS = 10*** и ***MinPts = 3***. Пространство признаков ограничено таким образом, что в нем может быть не более 3000 точек на расстоянии ***EPS*** друг от друга.

В связи с тем, что на вход алгоритма обработки поступает поток данных, по прошествии некоторого времени в этой модели обслуживания начинают действовать законы больших чисел. В частности, закон нормального распределения независимых равновероятных событий, таких как попадание новых точек в уже имеющиеся области пространства признаков.

Распределение данных по новым и старым кластерам в тестовой выборке подчиняется нормальному закону. То есть, число элементов, образующих новые кластеры, описывается убывающей функцией, стремящейся к нулю по мере заполнения пространства признаков, в то же время число элементов, поступающих в уже существующие кластеры, описывается возрастающей функцией, входящей в насыщение по мере заполнения пространства признаков.

Распределение точек данных внутри ***dim***-мерного кластера также описывается нормальным распределением Гаусса на ***dim***-мерной поверхности. При этом вероятность попадания в ***Список Элементов Границы MOD1*** любого кластера для точки из фрейма расчитывается по формуле $\boldsymbol{0.9972^{dim} - 0.9544^{dim}}$ по аналогии с законом трех сигм для одномерной случайной величины [22].

Формы кластеров подобраны так, что в среднем около 50% точек ***Списка Элементов Границы MOD1*** являются вершинами выпуклых многогранников, описывающих границы кластеров, входящих в ***Список Элементов Границы MOD2***. Согласно [21], число сторон таких многогранников можно расчитать по формуле (12). Графики на рис.10 иллюстрируют вычислительную сложность обработки потоковых данных при разных показателях размерности. На каждом графике по оси абсцисс отложено количество обработанных фреймов, а на оси ординат – суммарное количество операций над точкой размерности ***dim***, необходимых для обработки общего объема данных поступивших на вход. Кроме модификаций MOD1 и MOD2 на графиках также показаны результаты работы классического DBSCAN, настроенного на сохранение точек, принадлежащих всем обнаруженным кластерам.

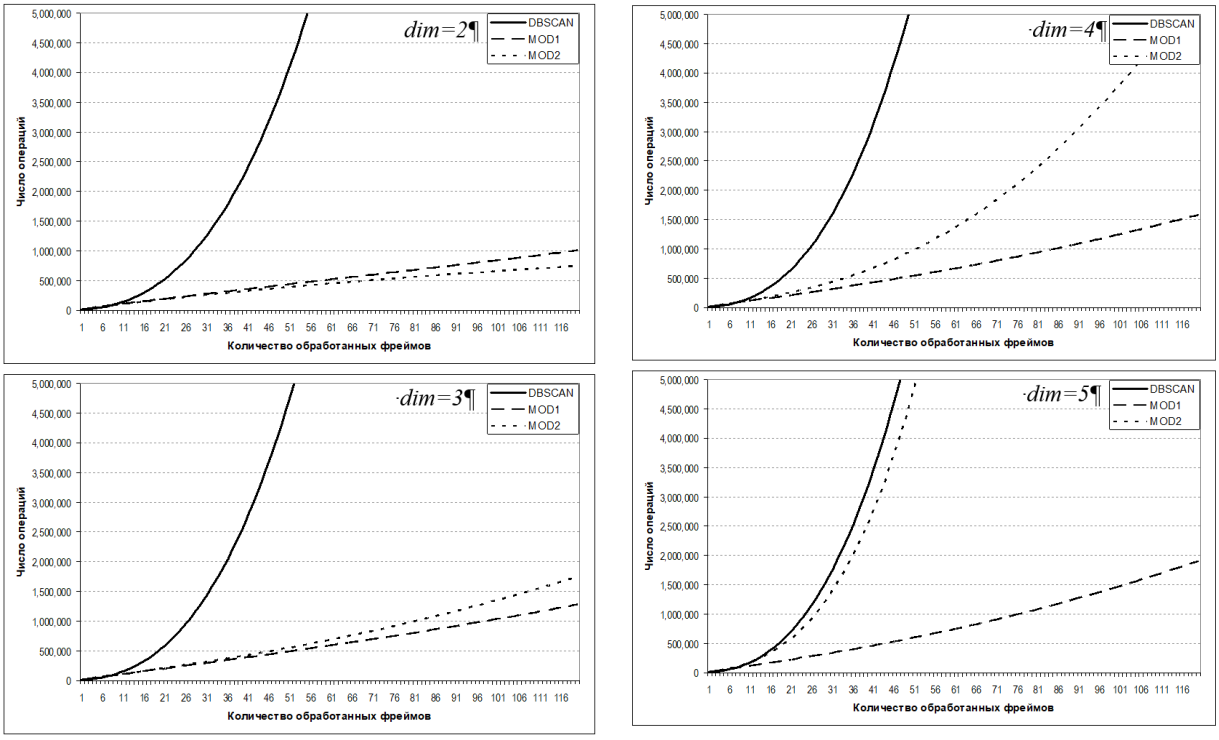

Рисунок 10. Сравнение динамики вычислительной сложности обработки данных для разных модификаций при размерности данных от 2 до 5

Стоит заметить, что вычислительная сложность MOD1 менее зависима от размерности пространства признаков, чем вычислительная сложность MOD2 (см. рис.10): при ***dim = 2*** вычислительная сложность MOD2 растет медленнее, чем MOD1, но при ***dim = 3*** и ***dim = 4*** вычислительная сложность MOD2 растет быстрее, а начиная с ***dim = 5*** превышает таковую даже для классического DBSCAN. Это связано с тем, что число сторон выпуклого многоугольника с повышением размерности пространства признаков растет быстрее, чем число точек на границах кластеров.

Экспериментальное подтверждение оценки вычислительной сложности

Для проверки вышеизложенных математических оценок вычислительной сложности был проведен ряд экспериментов с реальными данными из набора «Electrical Fault detection and classification» [23] существующего в открытом доступе в рамках международной платформы соревновенований по исследованию данных и машинному обучению «Kaggle». Означенный набор данных содержит 12 000 замеров напряжения и силы тока в трехфазной сети для каждой фазы, в режимах нормальной работы и короткого замыкания одной или нескольких фаз.

Для проведения эксперимента на данных размерности 2 использовались замеры силы тока для фаз «A» и «B», для эксперимента на данных размерности 3 использовались замеры силы тока для фаз «A», «B» и «C», при этом предобработка данных не проводилась. Подтверждение характера зависимостей на данных размерностью 2 и 3 считаем достаточным основанием, чтобы принять верными все выкладки предыдущего раздела статьи, так как они универсальны с точки зрения размерности данных.

Как и в предыдущем эксперименте данные подавались на вход каждой модификации (MOD1 и MOD2) одинаковыми фреймами по 100 точек, и при таких же настройках алгоритма DBSCAN. Графики на рис.11 иллюстрируют изменения времени обработки потоковых данных при разной размерности данных. На каждом графике по оси абсцисс отложено количество обработанных фреймов, а по оси ординат – суммарное время, затраченное алгоритмом на их обработку. При сравнении графиков на рис.10 и 11 видно, что характеры отображаемых зависимостей совпадают. Это позволяет говорить о том, что математическая оценка вычислительной сложности, приведенная в предыдущем разделе статьи, была верной.

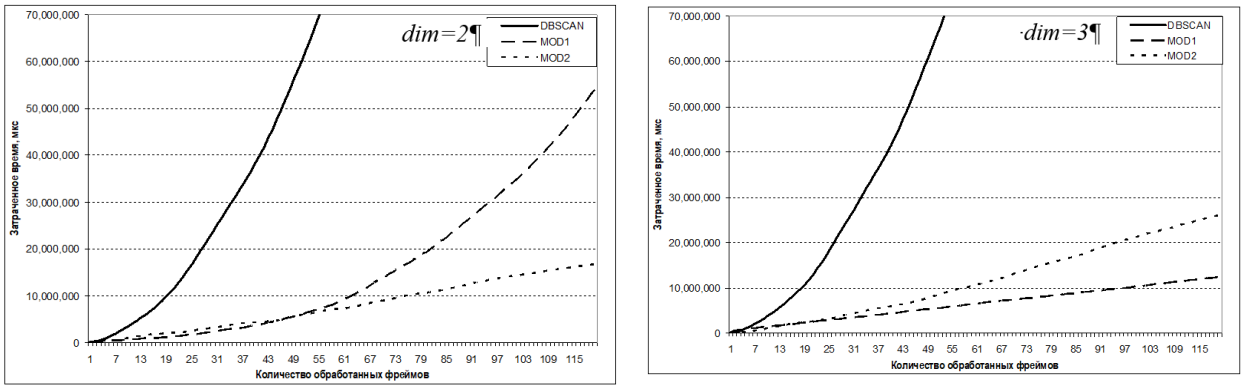

Рисунок 11. Сравнение времени обработки данных для разных модификаций на наборе реальных данных при размерности данных 2 и 3

Разница в масштабах графиков на рис.10 и рис.11, принятых для оси ординат при том же числе измерений по оси абсцисс объясняется тем, что расчет вычислительной сложности (рис.10) производился в количестве операций над точкой данных размерности ***dim***, тогда как в реальном эксперименте замерялось время обработки в микросекундах (рис.11). Таким образом, конкретная авторская реализация алгоритма, работающая в 64-битной среде Windows 10 на домашнем персональном компьютере (ПК) со средними характеристиками, производит одну операцию над многомерной точкой за несколько микросекунд. Точное соотношение между временем и количеством операций серьезно зависит от вычислительной мощности конкретного ПК.

Эксперимент также показал, что размер фрейма в 100 точек достаточно мал, чтобы на графиках еще можно было заметить флуктуации, вызванные неравномерным распределением поступающих на вход реальных значений. Кроме того, результаты, показанные расчётной оценкой вычислительной сложности, являются теоретическим пределом быстродействия. Любая реализация алгоритма на контретном языке программирования, работающая под управлением конкретной операционной системы на конкретной аппаратной платформе, будет стремиться к рассчетной оценке в зависимости от оптимальности реализации и условий ее работы. Но даже худшие из приведенных результатов показывают выигрыш в быстродействии модифицированного алгоритма относительно классического DBSCAN от 3-ех до 10-и раз.

Сравнение точности кластеризации

Для проверки точности кластеризации был проведен эксперимент на данных размерности 3, взятых из 5-и тестовых наборов, относящихся к различным отраслям и с заведомо несовместимыми плотностями распределения информации для изучения поведения алгоритмов при разном распределении данных по пространству признаков.

В эксперименте были использованы следующие источники данных:

Набор данных «Electrical Fault detection and classification» [23]. Набор содержит замеры силы тока трехфазной сети в нормальном режиме работы и котороткого замыкания, которые можно считать информационным шумом (короткое замыкание) и кластерами (нормальный режим работы).

Набор данных «IMS Bearing Dataset» [24]. Он содержит три этапа экспериментов по проверке износоустойчивости подшипников с замерами вибрации. Каждый эксперимент проводился на разных наборах подшипников, поэтому распределение вибрации существенно отличается.

Набор данных «Individual Household Power Consumption» [25]. В нем содержится статистика расхода электроэнергии одного частного домохозяйства в течении нескольких лет. Потребление энергии в тои или ином помещении зависит от времени, что создает несколько перескекающихся кластеров.

Набор данных «Online shoppers intention» [26]. В нем содержится статистика посещения пользователями некоторого сайта страниц с разным содержанием.

Набор данных «Postures» [27]. В нем содержатся данные о положении в пространстве ключевых точек на специальных перчатках для отслеживания жестов руками.

Чтобы создать равные условия, из каждого набора данных было взято по 10'000 элементов, и задействованы три измерительных канала (пространство признаков в каждом случае – трехмерное). Других мероприятий по предобработке данных не проводилось.

Точность оценивалась при помощи метрик Rand Index и Adjusted Rand Index. Эти метрики позволяют сравнивать ответы двух разных алгоритмов кластеризации на одном и том же наборе данных [28]. Метрики опираются на подсчет пар объектов, которые оба классификатора отнесли к одному классу, пар объектов, которые оба классификатора отнесли к разным классам, и пар объектов, которые один классификатор отнес к разным кластерам, а другой – к одному кластеру. Таким образом метрики не опираются на конкретные обозначения кластеров, а только на логику кластеризации.

Ответы Rand Index (RI) находятся в диапазоне [0; 1] где 1 означает полное соответствие ответов кластеризации, а 0 – полное несоответствие. Adjusted Rand Index (ARI) вводит понятие случайного совпадения. Данная метрика дает оценку в диапазоне [-1; 1], где 1 – полное соответствие ответов кластеризации, 0 – совпадение ответов, которое можно считать случайным, -1 – полное несоответствие.

В качестве эталонного алгоритма взят классический DBSCAN. Результаты представлены в Таблице 1.

Как показал детальный анализ, при определенном распределении данных гибридный DBSCAN разбивает на несколько кластеров данные, которые классический DBSCAN относит к одному кластеру. Также, можно отметить, что гибридный DBSCAN чувствителен к вырожденным случаям распределения данных по кластерам. Так, например, если фактическое распределение данных по пространству признаков меньше чем размерность самого пространства признаков, то модификация MOD2 испытывает трудности с построением выпуклой границы кластера.

Для модификации MOD1, слабым местом являются кластеры, сгруппированные так плотно, что не имеют граничных элементов. Тем не менее гибридный DBSCAN правильно идентифицируют выбросы даже в экстремальных случаях, что позволяет говорить о высокой эффективности алгоритма как минимум для интеллектуальной фильтрации шума и других подобных задач.

Расчет требуемой памяти

Перейдем к расчету памяти, требуемой для поддержания корректной работы описанных алгоритмов. Здесь мы будем оперировать только математическими оценками, так как реальные объемы используемой памяти зависят от модели данных, особенностей механизмов управления памятью конкретной операционной системы и множества других факторов.

Поэтому, они не могут быть эффективно подтверждены в результате простого эксперимента. В то же время, законы распределения и соотношения требований по памяти между различными модификациями останутся теми же вне зависимости от деталей конкретной реализациии.

Резюмируя вышесказанное, под используемой памятью будем подразумевать количество точек размерности ***dim***, которое сохраняет та или иная модификация, а не объем памяти в килобайтах.

Таблица 1. Оценка точности кластеризации.

Датасет	Режим работы	Эталонный алгоритм	RI	ARI
Electric Fault Prediction_3d	MOD1	DBSCAN	0.517 *	n/a **
	MOD2		0.938	0.910
	HybridDBSCAN с автоматическим переключением MOD1 и MOD2		0.934	0.865
Individual Household Power Consumption_3d	MOD1	DBSCAN	0.929	0.654
	MOD2		0.818	0.635
	HybridDBSCAN с автоматическим переключением MOD1 и MOD2		0.956	0.911
IMS Bearing Dataset_1_3d	MOD1	DBSCAN	0.745 *	n/a **
	MOD2		0.522 ***	n/a **
	HybridDBSCAN с автоматическим переключением MOD1 и MOD2		0.752 *	n/a **
IMS Bearing Dataset_2_3d	MOD1	DBSCAN	0.716 *	n/a **
	MOD2		0.536 ***	n/a **
	HybridDBSCAN с автоматическим переключением MOD1 и MOD2		0.718 *	n/a **
IMS Bearing Dataset_3_3d	MOD1	DBSCAN	0.630 *	n/a **
	MOD2		0.493 ***	n/a **
	HybridDBSCAN с автоматическим переключением MOD1 и MOD2		0.886	0.533
Online shoppers intention_3d	MOD1	DBSCAN	0.807	0.614
	MOD2		1.000	1.000
	HybridDBSCAN с автоматическим переключением MOD1 и MOD2		0.837	0.675
Postures_3d	MOD1	DBSCAN	0.927	0.716
	MOD2		0.970	0.904
	HybridDBSCAN с автоматическим переключением MOD1 и MOD2		0.973 *	0.905

* – точность снижена из-за разделения эталонных кластеров в вырожденных случаях, но форма сохраняется.
** – не удалось рассчитать в связи с особенностями данных, подобно указанному выше (*).
*** – точность снижена из-за неравномерности распределения данных в пространстве признаков.

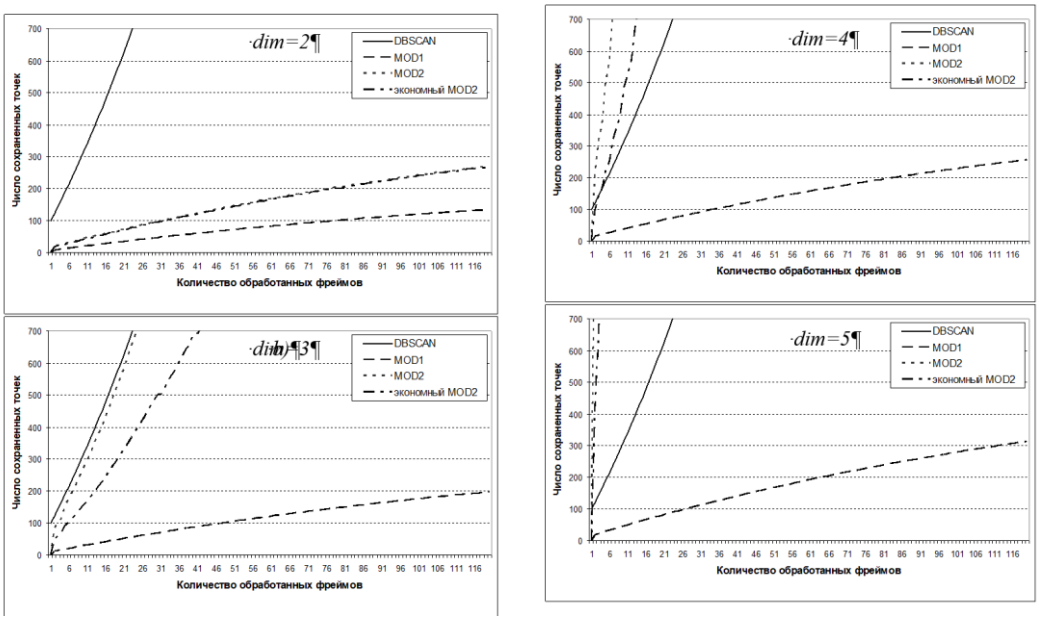

Рисунок 12. Сравнение требуемого объема памяти для разной модификации при размерности данных от 2 до 5

В случае MOD1 основной кластер представляет собой некоторую ***dim***-мерную фигуру, элементы границы составят ***dim***-мерную «ленту» по поверхности этой фигуры, с шириной равной радиусу соседства (***EPS***). Проводя аналогию с задачами многомерной геометрии, допустим, что число точек кластера или его границы можно выразить через объем соответствующей фигуры. Тогда, если объем ***dim***-мерной фигуры, представляющей кластер, вычисляется взятием ***dim***-кратного интеграла от некоторой функции, то объем фигуры, представляющей его границу, вычисляется взятием ***dim***-кратного интеграла от производной той же самой функции, умноженной на ***EPS***.

Таким образом, отношение элементов кластера к элементам границы оценивается как:

$$\frac{cin + cout}{cout}, \quad (17)$$

где ***cin*** – число точек, находящихся внутри кластера, а ***cout*** – число точек, находящихся на границе кластера.

В случае MOD2 требуемый объем памяти пропорционален числу вершин и сторон описанной кластером выпуклой фигуры согласно выражению (12). Из числа всех элементов кластера для работы MOD2 необходим набор точек, соответствующий вершинам выпуклого многогранника, и информация о сторонах многогранника. Она включает в себя набор вершин, соответствующей каждой стороне, набор граней, список ссылок на соседние стороны и вектора нормали к гиперплоскости, то есть аналог еще одной точки данных, а также грани, каждая из которых представляет набор вершин, через которые проходит эта грань. Таким образом, эквивалентное число хранимых точек данных оценивается как

$$cvt + n*(dim+1) + crt*(dim-1), \quad (18)$$

где ***cvt*** – общее число вершин всех кластеров, ***n*** – суммарное число сторон выпуклых многомерных фигур,

crt – общее число граней всех кластеров.

Наилучшей формой кластера является симплекс, у которого

$$cvt = dim+1,\ n = dim+1,\ crt = dim*(dim+1), \quad (19)$$

и каждая сторона должна иметь ***dim*** вершин.

Графики на рис.12 иллюстрируют зависимость требуемой памяти для классического DBSCAN и модификаций MOD1, MOD2 от потока входных данных разной размерности. По оси ординат отложено число точек размерности ***dim***, которые необходимо держать в памяти для сохранения границы всех найденных к этому моменту кластеров, а по оси абсцисс – объем данных для обработки, выраженный числом фреймов.

Как можно видеть, размер памяти, требуемый MOD2 сильнее, зависит от размерности пространства признаков, чем MOD1: если при ***dim = 2*** для сохранения границ кластеров число точек, которое требуемое MOD2, сопоставимо с таковым для MOD1, то начиная с ***dim = 3*** требования MOD2 растут экспоненциально быстрее таковых для MOD1, а начиная с ***dim = 4***, превышают таковые для классического DBSCAN. Причина такого роста кроется в экспоненциальном росте числа сторон ***dim***-мерного многогранника при увеличении размерности входных данных. Важным фактором является то, что каждая сторона многогранника сохраняет список вершин, принадлежащих этой стороне, то есть повторно записывает не менее ***dim+1*** точек данных, включая нормаль к поверхности.

При этом, замена координат вершин ссылками на вершины не решает проблему (см. рис.12, «экономный MOD2»): экономия памяти для MOD2 по сравнению с DBSCAN сохраняется до ***dim = 3*** включительно, но при ***dim >= 4*** требования по памяти для MOD2 растут быстрее.

Таким образом, MOD1 является более эффективной модификацией с точки зрения затрат памяти. Модификация MOD2 может играть лишь вспомогательную роль в пространствах признаков малых размерностей и слабо применима в пространствах признаков больших размерностей.

Заключение

В рамках данной работы рассмотрены общие принципы работы алгоритма кластеризации DBSCAN, обозначены границы его применимости, связанные с особенностями организации данных, в частности, проблемы при обработке потоков данных. Рассмотрены пути решения означенных проблем.

Предложенная комплексная модификацикация включает в себя часть, предназначенную для работы с одномерными потоковыми данными, которая использует их особенности, что привело к низкой вычислительной сложности (близкая к линейной) и экономному расходу памяти. Другие части позволяют приспособить этот алгоритм для работы с потоками многомерных данных с соответствующими им интеллектуальными процедурами проверки нахождения точки данных внутри границы кластера, сочетающими в себе как математические подходы, так и методы машинного обучения.

Блок MOD1 опирается на исходные принципы работы алгоритма DBSCAN и позволяет оперировать кластерами произвольной формы, однако требует наличия достаточного количества граничных точек данных и расположения этих точек по всей границе кластера. К положительным сторонам данной модификации стоит

отнести высокую скорость выполнения процедуры проверки вхождения точки данных в кластер, работающей с кластерами произвольной формы и любой размерности. Главной проблемой данной модификации является возможность потери кластеров в результате поступления новых точек данных не образующих замкнутый контур в пространстве признаков.

Блок MOD2 позволяет игнорировать граничные точки и опираться лишь на корневые точки кластера, следовательно, он будет работать независимо от организации входных данных. Однако эта модификация позволяет оперировать лишь кластерами выпуклой формы и обладает большей вычислительной сложностью. К положительным сторонам данной модификации стоит отнести универсальность процедуры проверки вхождения точки данных в выпуклую фигуру, работающий в многомерном пространстве признаков. Главными проблемами этой модификации являются неэффективность работы в пространствах признаков высоких размерностей, а также большую сложность процедуры определения границ кластера, что приводит к росту ресурсоемкости всего процесса.

Резюмируя все вышесказанное хотелось бы отметить, что предложенная комплексная модификация алгоритма DBSCAN позволяет с большой точностью выделять в потоковых данных кластера произвольной формы с памятью всех элементов. Этой функции не было обнаружено у других модификаций DBSCAN для работы с потоковыми данными. Такое расширение функциональности потребовало существенного роста сложности алгоритма, по сравнению с другими модификациями, что должно увеличить требования к вычислительным ресурсам конечной прикладной системы. Однако, как было описано выше, синергия между методами машинного обучения и математическими методами позволила одновременно с расширением функциональности достигнуть выигрыша по используемой памяти и быстродействию по сравнению с классическим DBSCAN.

Список литературы

1. Jasmine Irani, Nitin Pise, Madhura Phatak, Clustering Techniques and the Similarity Measures used in Clustering: A Survey // International Journal of Computer Applications - 2016 – V.134 – No.7 – pp. 9-14
2. Geoffrey Charles Fox, John B. Rundle, Andrea Donnellan, Bo Feng, Earthquake Nowcasting with Deep Learning / Research Gate, Available at https://www.researchgate.net/publication/357170781 (дата обращения: 19.12.2023)
3. Jonathan A. Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, Andre C. P. L. F. De Carvalho, Joao Gama, Data Stream Clustering: A Survey / ACM Computing Surveys, Vol. 46, No. 1, Article 13, Publication date: October 2013
4. Umesh Kokate, Arvind Deshpande, Parikshit Mahalle, Pramod Patil, Data Stream Clustering Techniques, Applications, and Models: Comparative Analysis and Discussion // Big Data and Cognitive Computing, 2018, 2:32; doi:10.3390/bdcc2040032
5. Xin Wang, Zhengru Wang, Zhenyu Wu, Shuhao Zhang, Xuanhua Shi, Li Lu, Data Stream Clustering: An In-depth Empirical Study // ACM Manag. Data., Vol. 1, No. 2, Article 162. Publication date: June 2023.
6. N.E. Ivari, M. Wachowicz, P.A.H. Williams, T. Agnew, Discovering clustering patterns from e-counter data streams // ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume X-4/W3-2022 7th International Conference on Smart Data and Smart Cities (SDSC), 19–21 October 2022, Sydney, Australia
7. Zhang T., Ramakrishnan R., Livny M, BIRCH: an efficient data clustering method for very large databases, // ACM SIGMOD international conference on Management of data - SIGMOD '96. pp. 103–114.
8. Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, Jörg Sander. OPTICS: Ordering Points To Identify the Clustering Structure // ACM SIGMOD international conference on Management of data. — ACM Press, 1999. — С. 49–60.
9. Демидова Л.А., Митин Г.В., Сравнительный анализ современных методов машинного обучения в контексте специфики их требований к обучающей выборке // Актуальные проблемы прикладной математики, информатики и механики: сборник трудов Международной научной конференции, 13-15 декабря 2021 г. Воронеж, 2022. - С. 1547-1556
10. Панов А.В., Митин Г.В. Сравнительный анализ современных методов машинного обучения в контексте специфики типов их выходных значений // Актуальные проблемы прикладной математики, информатики и механики: сборник трудов Международной научной конференции, 12-14 декабря 2022 г. Воронеж, 2023. - С. 1426-1435
11. Hassani M, Spaus P, Gaber MM, Seidl T., Density-Based Projected Clustering of Data Streams, 2012 (pp. 311-324). Springer Berlin Heidelberg

12. Tang Jiaowei, An Algorithm of streaming clustering / Uppsala universitet Teknisk- naturvetenskaplig fakultet UTH-enheten, 2011

13. David Tam, Reza Azimi, Michael Stumm, Thread Clustering: Sharing-Aware Scheduling on SMP-CMP-SMT Multiprocessors // Proc. European Conf. on Computer Systems (EUROSYS 07), March 2007, pp. 47-58.

14. Mu C, Hou Y, Zhao J, Wei S, Wu Y, Stream-DBSCAN: A Streaming Distributed Clustering Model for Water Quality Monitoring / Applied Sciences. 2023 Apr 26;13(9):5408.

15. Yiqiu Wang, Yan Gu, Julian Shun, // Theoretically-Efficient and Practical Parallel DBSCAN // Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, 2020

16. Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer, Xiaowei Xu, Incremental Clustering for Mining in a Data Warehousing Environment // VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, 1998, -p 323-333.

17. Birant Derya, Alp Kut, ST-DBSCAN: An algorithm for clustering spatial-temporal data // Data & Knowledge Engineering 60, 2007, p. 208-221.

18. Yikai Gong, Richard O. Sinnott, Paul Rimba, RT-DBSCAN: Real-Time Parallel Clustering of Spatio-Temporal Data Using Spark-Streaming // Computational Science – ICCS, 2018 -pp.524-539

19. Hamza Mustafa, Eleazar Leal, Le Gruenwald, An Experimental Comparison of GPU Techniques for DBSCAN Clustering // IEEE International Conference on Big Data, 2019

20. Martin Ester, Hans-Peter Kriegel, Jorg Sander, Xiaowei Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise // KDD-96 Proceedings. 1996, -p 226-231.

21. C Bradford Barber, David P. Dobkin, Hannu Huhdanpaa. The Quickhull Algorithm for Convex Hulls / ACM Transactions of Mathematical Software, 9 Jan 1995.

22. Вентцель Е.С. Теория вероятностей - издание третье, исправленное. М: Изд. "Наука". 1968. - 575с.

23. Electrical Fault detection and classification / Kaggle, Available at https://www.kaggle.com/datasets/esathyaprakash/electrical-fault-detection-and-classification (дата обращения: 19.12.2023)

24. IMS Bearing Dataset / Kaggle, Available at https://www.kaggle.com/datasets/vinayak123tyagi/bearing-dataset (дата обращения: 19.12.2023)

25. Individual household electric power consumption / UC Irvine Machine Learning Repository, Available at https://archive.ics.uci.edu/dataset/235/individual+household+electric+power+consumption (дата обращения: 19.12.2023)

26. Online shoppers intention / UC Irvine Machine Learning Repository, Available at https://archive.ics.uci.edu/dataset/468/online+shoppers+purchasing+intention+dataset (дата обращения: 19.12.2023)

27. Motion capture hand postures / UC Irvine Machine Learning Repository, Available at https://archive.ics.uci.edu/dataset/405/motion+capture+hand+postures (дата обращения: 19.12.2023)

28. Lawrence Hubert, Phipps Arabie, Comparing Partitions // Journal of classification 2:193-218 -1985 (дата обращения: 19.12.2023)

References

1. Jasmine Irani, Nitin Pise, Madhura Phatak, Clustering Techniques and the Similarity Measures used in Clustering: A Survey // International Journal of Computer Applications - 2016 – V.134 – No.7 – pp. 9-14

2. Geoffrey Charles Fox, John B. Rundle, Andrea Donnellan, Bo Feng, Earthquake Nowcasting with Deep Learning / Research Gate, Available at https://www.researchgate.net/publication/357170781 (date of application: 12/19/2023)

3. Jonathan A. Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, Andre C. P. L. F. De Carvalho, Joao Gama, Data Stream Clustering: A Survey / ACM Computing Surveys, Vol. 46, No. 1, Article 13, Publication date: October 2013

4. Umesh Kokate, Arvind Deshpande, Parikshit Mahalle, Pramod Patil, Data Stream Clustering Techniques, Applications, and Models: Comparative Analysis and Discussion // Big Data and Cognitive Computing, 2018, 2:32; doi:10.3390/bdcc2040032

5. Xin Wang, Zhengru Wang, Zhenyu Wu, Shuhao Zhang, Xuanhua Shi, Li Lu, Data Stream Clustering: An In-depth Empirical Study // ACM Manag. Data., Vol. 1, No. 2, Article 162. Publication date: June 2023.

6. N.E. Ivari, M. Wachowicz, P.A.H. Williams, T. Agnew, Discovering clustering patterns from e-counter data streams // ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume X-4/W3-2022 7th International Conference on Smart Data and Smart Cities (SDSC), 19–21 October 2022, Sydney, Australia

7. Zhang T., Ramakrishnan R., Livny M, BIRCH: an efficient data clustering method for very large databases, // ACM SIGMOD international conference on Management of data - SIGMOD '96. pp. 103–114.

8. Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, Jörg Sander. OPTICS: Ordering Points To Identify the Clustering Structure // ACM SIGMOD international conference on Management of data. - ACM Press, 1999. — pp. 49-60.

9. Demidova L.A., Mitin G.V., Comparative analysis of modern machine learning methods in the context of the specifics of their requirements for a training sample // Actual problems of applied mathematics, computer science and mechanics: proceedings of the International Scientific Conference, December 13-15, 2021 Voronezh, 2022. - pp. 1547-1556

10. Panov A.V., Mitin G.V. Comparative analysis of modern machine learning methods in the context of the specifics of the types of their output values // Actual problems of applied mathematics, computer science and mechanics: proceedings of the International Scientific Conference, December 12-14, 2022 Voronezh, 2023. - pp. 1426-1435

11. Hassani M, Spaus P, Gaber MM, Seidl T., Density-Based Projected Clustering of Data Streams, 2012 (pp. 311-324). Springer Berlin Heidelberg

12. Tang Jiaowei, An Algorithm of streaming clustering / Uppsala universitet Teknisk- naturvetenskaplig fakultet UTH-enheten, 2011

13. David Tam, Reza Azimi, Michael Stumm, Thread Clustering: Sharing-Aware Scheduling on SMP-CMP-SMT Multiprocessors // Proc. European Conf. on Computer Systems (EUROSYS 07), March 2007, pp. 47-58.

14. Mu C, Hou Y, Zhao J, Wei S, Wu Y, Stream-DBSCAN: A Streaming Distributed Clustering Model for Water Quality Monitoring / Applied Sciences. 2023 Apr 26;13(9):5408.

15. Yiqiu Wang, Yan Gu, Julian Shun, // Theoretically-Efficient and Practical Parallel DBSCAN // Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, 2020

16. Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer, Xiaowei Xu, Incremental Clustering for Mining in a Data Warehousing Environment // VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, 1998, -p 323-333.

17. Birant Derya, Alp Kut, ST-DBSCAN: An algorithm for clustering spatial-temporal data // Data & Knowledge Engineering 60, 2007, p. 208-221.

18. Yikai Gong, Richard O. Sinnott, Paul Rimba, RT-DBSCAN: Real-Time Parallel Clustering of Spatio-Temporal Data Using Spark-Streaming // Computational Science – ICCS, 2018 -pp.524-539

19. Hamza Mustafa, Eleazar Leal, Le Gruenwald, An Experimental Comparison of GPU Techniques for DBSCAN Clustering // IEEE International Conference on Big Data, 2019

20. Martin Ester, Hans-Peter Kriegel, Jorg Sander, Xiaowei Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise // KDD-96 Proceedings. 1996, -p 226-231.

21. C Bradford Barber, David P. Dobkin, Hannu Huhdanpaa. The Quickhull Algorithm for Convex Hulls / ACM Transactions of Mathematical Software, 9 Jan 1995.

22. Wentzel E.S. Probability Theory - third edition, revised. Moscow: Nauka Publishing House. The main edition of the physical and mathematical literature.1968. - 575c.

23. Electrical Fault detection and classification / Kaggle, Available at https://www.kaggle.com/datasets/esathyaprakash/electrical-fault-detection-and-classification (accessed: 12/19/2023)

24. IMS Bearing Dataset / Kaggle, Available at https://www.kaggle.com/datasets/vinayak123tyagi/bearing-dataset (date of application: 12/19/2023)

25. Individual household electric power consumption / UC Irvine Machine Learning Repository, Available at https://archive.ics.uci.edu/dataset/235/individual+household+electric+power+consumption (accessed: 12/19/2023)

26. Online shopper's intention / UC Irvine Machine Learning Repository, Available at https://archive.ics.uci.edu/dataset/468/online+shoppers+purchasing+intention+dataset (date of application: 12/19/2023)

27. Motion capture hand postures / UC Irvine Machine Learning Repository, Available at https://archive.ics.uci.edu/dataset/405/motion+capture+hand+postures (accessed: 12/19/2023)

28. Lawrence Hubert, Phipps Arabie, Comparing Partitions // Journal of classification 2:193-218 -1985 (accessed: 12/19/2023)