Разработка алгоритма структурирования данных для оптимизации классификации на основе метода ближайшего соседа

А.В. Мухин[1], Р.А. Парингер[1]

[1]Самарский национальный исследовательский университет им. академика С.П. Королева, Московское шоссе 34А, Самара, Россия, 443086

Аннотация. В данной статье рассматривается проблема вычислительно сложного алгоритма классификации, используемого в "Науке о данных", и предлагается подход к снижению вычислительной сложности процедуры классификации за счёт оптимизации процедуры перебора кластеров. Таким образом полный перебор заменяется направленным. Так же в статье непосредственно описывается алгоритмы построения структуры, по которой будет происходить направленный перебор.

1. Введение

«Большие данные», зачастую можно представить в виде множества точек в пространстве, причём, расположение их зачастую хаотично. [1] Количество этих точек исчисляется миллионами, если не миллиардами. Отсюда возникает проблема, описание и идеи преодоления которой будут изложены в статье.

Проблему можно сформулировать так: “Вычислительно сложная классификация”, иначе говоря, “медленная классификация”. Для классификации объектов чаще всего применяют обычный перебор, где для каждого класса находится расстояние до объекта, а результатом будет тот класс, который имеет наименьшее расстояние. Действительно, ведь для того что бы определить к какому классу относится тот или иной объект, потребуется посчитать довольно много расстояний, но сделать такое в «больших данных» может быть крайне сложно. Необходимость оптимизации классификации – очевидна. Поэтому далее будут предложены идеи структурирования данных на основе метода ближайшего соседа.

Метод ближайшего соседа успешно применяется в работах [2-4], следовательно, предложенные в данной статье идеи могут быть использованы для проведения экспериментов с бо´льшим количеством данных, что даст возможность рассмотреть поднимающиеся проблемы глубже.

2. Описание эксперемента

Далее будут представлены следующие алгоритмы:

1. NNA (Nearest Neighbor Algorithm, алгоритм на основе метода ближайшего соседа)
2. DPA (Delaunay Projection Algorithm, алгоритм основанный на использовании Триангуляции Делоне)
3. NRA (Neighbor Relations Algorithm, алгоритм на основе связности соседних ЦК)
    - NRA with crossing check
    - NRA with recheck relations

- NRA without any check

«*Кластер* - группа объектов, которые расположены в многомерном пространстве переменных максимально близко друг к другу и при этом максимально удалены от объектов из других групп». [5]

«*Центр кластера (ЦК)* - наиболее типичный представитель данного кластера (его геометрический центр). По характеристикам центра кластера можно судить обо всем кластере» [5].

Немного подробнее об идее поиска ближайшего центра кластера (далее ЦК) для алгоритмов DPA и NRA. Суть идеи в том, чтобы построить между ЦК связи такие, что при переходе от ЦК, к некоторому соседнему ЦК, сосед оказывался ближе к определяемой точке, чем предыдущий ЦК. Таким образом при использовании такой идеи, количество требуемых для проверки кластеров сократиться в несколько раз.

Центры и связи будут храниться в структуре, напоминающей неориентированный, взвешенный граф. Где точки определяются координатами в евклидовом n-мерном пространстве, а вес ребра определяется евклидовым расстоянием между точками, которые соединяет ребро. Т.к. в таком графе вероятно буду петли и циклы, во всех алгоритмах (в частности рекурсивных) присутствуют соответствующие проверки, чтобы избежать некорректного поведения программы.

Такой алгоритм можно оптимизировать добавлением кэша и началом проверки с центра пространства (под центром подразумевается центр распределения ЦК)

Эксперимент проводился с использованием итеративного классификатора на основе микро-сервисов (Рисунок 1). Особенностью такой архитектуры является модульность, благодаря которой становится проще проводить тесты с различными алгоритмами и наборами данных.

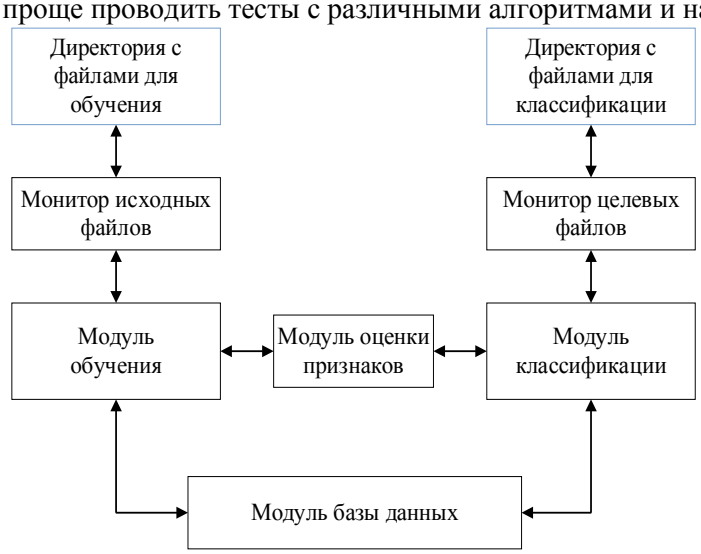

Рисунок 1. Архитектура приложения.

Эксперимент проводился на компьютере с фиксированной частотой центрального процессора, также, директории с файлами для обучения и классификации находились на ramdisk.

В эксперименте используются tiff изображения, признаками которого являются средние по каждой из компонент (RGB). Множество ЦК случайным образом разбито на три группы по 10 изображений. Множество целевых файлов представляет собой подборку из 100 различных пейзажей.

2.1 Алгоритм эксперимента:

1. Запуск таймера. Загружаем группу кластеров A в директорию с файлами для обучения. Структурируем (если требуется). Останавливаем таймер и запоминаем результат. Сброс таймера.

2. Запуск таймера. Загружаем целевые файлы в директорию с файлами для классификации и классифицируем их. Останавливаем таймер и запоминаем результат. Сброс таймера.
3. Запуск таймера. Загружаем группу кластеров B в директорию с файлами для обучения. Структурируем (если требуется). Обновляем классы у объектов, там, где это требуется. Останавливаем таймер и запоминаем результат. Сброс таймера.
4. Запуск таймера. Загружаем группу кластеров C в директорию с файлами для обучения. Структурируем (если требуется). Обновляем классы у объектов, там, где это требуется. Останавливаем таймер и запоминаем результат. Сброс таймера.
5. Запуск таймера. Удаляем группу кластеров D из директории с файлами для обучения. Структурируем (если требуется). Обновляем классы у объектов, там, где это требуется. Останавливаем таймер и запоминаем результат. Сброс таймера.

Группы A, B, C, D задаются при запуске программы.
Оценка точности производится по формуле и выполняется перед удалением группы кластеров D:

$$accuracy = \frac{correct}{total} \quad (1)$$

где correct – количество верно классифицированных объектов, total – общее количество объектов.

3. Алгоритм на основе метода ближайшего соседа

И так, преимущество NNA в том, что для поиска ближайшего ЦК не нужно хранить абсолютно никакой структуры с их описанием. Следующим его преимуществом является абсолютная точность. Далее о том, как работает NNA.

1) Добавление ЦК.

Новые точки добавляются во множество (имеется в виду стандартная для языков программирования структура). Таким образом обеспечивается хранение точек пространства.

2) Поиск ближайшего ЦК.

Поиск происходит путём подсчёта Евклидова расстояния по всем ЦК, находящимся в пространстве. Так находится ближайший центр кластера для точки.

3) Удаление ЦК.

Удаление ЦК тривиально. Он просто удаляется из множества всех ЦК.

Единственным и крупным недостатком является то, что при большом количестве ЦК становится трудоёмко проверять абсолютно все ЦК. Следовательно, он получается довольно медлителен. Далее можно увидеть таблицу, на которой видно, что в среднем, алгоритм работает 27.922 с.

Таблица 1. Время работы NNA с разными параметрами.

FirstOption	SecondOption	ThirdOption	FourthOption	TotalTime(с.)	Accuracy(%)
A	B	C	A	28.66	1.00
A	C	B	A	27.32	1.00
B	A	C	A	27.69	1.00
B	C	A	A	27.52	1.00
C	A	B	A	27.43	1.00
C	B	A	A	27.42	1.00

4. Алгоритм основанный на использовании Триангуляции Делоне

Алгоритм основанный на использовании Триангуляции Делоне в двумерном пространстве.
«*Триангуляция Делоне* - триангуляция для заданного множества точек S на плоскости, при которой для любого треугольника все точки из S за исключением точек, являющихся его вершинами, лежат вне окружности, описанной вокруг треугольника» [6].

Идея использовать данный алгоритм возникла из-за его свойств в двумерном пространстве. Другим подспорьем для реализации DPA является трудоёмкость описания динамической триангуляции Делоне для n-мерного пространство.

Для простоты объяснения, предположим, что в трёхмерном пространстве существует множество S_N из N центров кластера (далее просто ЦК) и проекции этих ЦК на плоскости образуют триангуляцию Делоне. Пусть, во множество S_N поместили новый ЦК – l. Тогда:

Множество соседних ЦК l определяется по следующему алгоритму:

Шаг 1. Новый ЦК l проецируется на каждую из плоскостей пространства. (Далее алгоритм рассматривается в плоскостях)

Шаг 2. Определяется местоположение точки на плоскости. Она может находится либо в одном из треугольников триангуляции, либо на ребре одного из треугольников, либо вне триангуляции. Каждый случай обрабатывается отдельно.

Шаг 3.1. *Точка оказалась в треугольнике*. В таком случае достаточно разбить треугольник, в котором находится точка, на три треугольника (рис 2). Переходим к шагу 4.

Шаг 3.2. *Точка оказалась на ребре треугольника*. Здесь же два случая. Первый из них – ребро на котором оказалась точка принадлежит лишь одному треугольнику. Тогда этот треугольник разбивается на два. Второй случай – ребро на котором лежит точка, принадлежит двум треугольникам. Тогда два треугольника разбиваются на четыре (рис 3.). Переходим к шагу 4.

Рисунок 2. Алгоритм добавления точки в треугольник.

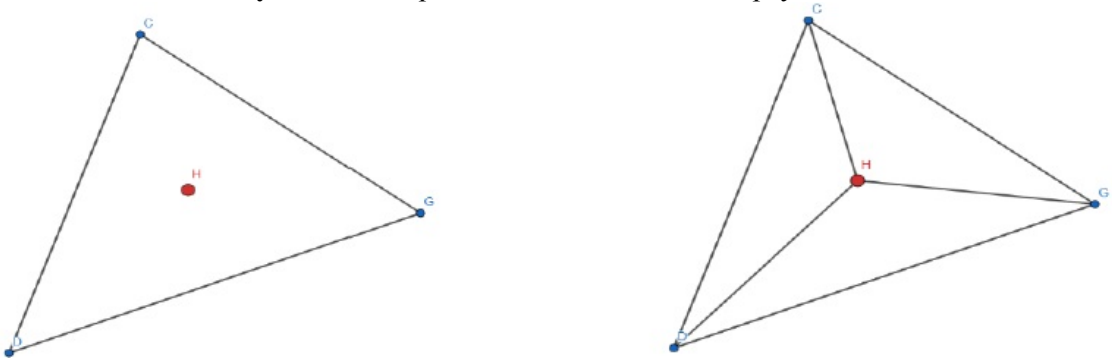

Рисунок 3. Добавление точки, лежащий на ребре триангуляции.

Шаг 3.3. *Точка оказалась вне триангуляции*. В таком случае, эта точка образует треугольники со всеми рёбрами, являющимися краевыми триангуляции и, если точка лежит по ту сторону ребра, с которой нет треугольника (рис 4.). Переходим к шагу 4.

Шаг 4. Проверка условия Делоне для всех образованных треугольников. Триангуляция станет триангуляцией Делоне за конечное число перестроений. (Доказательство этого факта можно найти здесь: [7]) Таким образом ЦК l будет корректно вставлен в каждую триангуляцию.

Шаг 5. Все связи, которые будет иметь точка в триангуляциях и будут связями точки в пространстве.

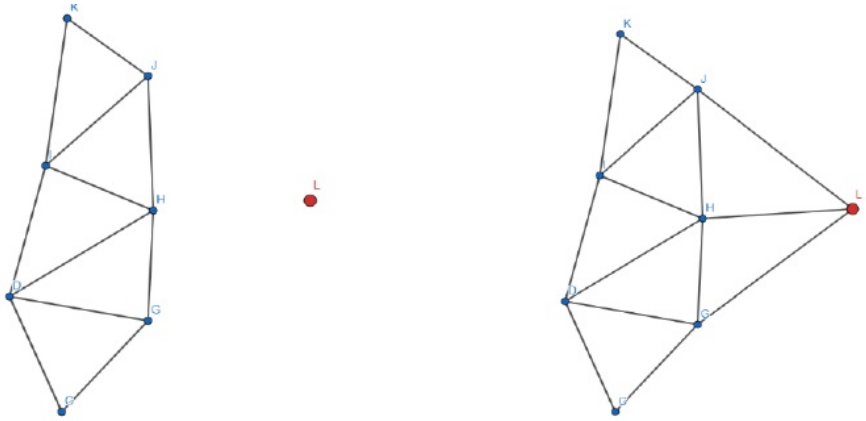

Рисунок 4. Добавление точки, лежащей вне триангуляции.

4.1. Проверка условия Делоне

Проверка условия Делоне для треугольника заключается в том, что, нужно определить, находятся ли вершины соседних треугольников (речь идёт о тех вершинах, которые не принадлежат проверяемому треугольнику), вне окружности, описанной вокруг проверяемого треугольника. Если условие нарушается, то необходим перестроить соседние треугольники. Т.к. два соседних треугольника образуют четырёхугольник с проведённой диагональю, то для восстановления условия Делоне необходимо будет переместить диагональ (Рис. 5). Красное ребро – ребро до перемещения, синее – после. Эту операцию обычно называют “Флипом”.

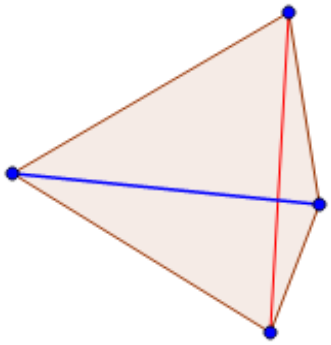

Рисунок 5. Флип. Красны – диагональ до флипа. Синий – диагональ после флипа.

4.2. Объединение результатов двумерной триангуляции

После добавления нового ЦК, каждая из 2D триангуляций возвращает соседей добавленного центра. Здесь стоит напомнить, что эти значения представляются координатами точек двумерного пространства, а исходная же точка принадлежит трёхмерному.

Сначала нужно объединить все полученные данные о соседях в их общее множество. Для корректной работы алгоритма требуется преобразование точек двумерного пространства в трёхмерное, путём поиска подходящей трёхмерной точки. (под подходящей – подразумевается та точка 3-ехмерного пространства, которая имеет те же координаты по тем же компонентам что и 2-ухмерная точка). После такого преобразования, получится сформировать список соседей для добавленного ЦК. Далее необходимо обновить списки соседей всех ЦК пространства.

4.3. При реализации DPA возникли следующие проблемы

Потеря информации. Так, например, для точек (255, 0, 0) и (254, 0, 0) в проекции на ось YOZ мы будем иметь одну точку (0, 0). В условиях работы программы, при передаче информации о связях обратно в пространство, мы потеряем информацию о том, с какой именно точкой мы работает (будет выбрана та точка 3-мерного пространства, которая первая подойдёт по координатам).

- Переизбыток связей (рис 6.).
- Не идеальная точность (в среднем = 0.91(3)) при большом числе связей.
- Медлительность (в среднем = 29.792 (аналогично классическому **NNA**) без учёта удаления) и сложность реализованного алгоритма.
- Неустойчивость алгоритма. Так при одних стартовых группах точность поднимается до 97%, при других опускается до 84%. Так же, точность очень сильно зависит от начальной точки, с которой начинается поиск ближайшего кластера. В таблице 2 можно увидеть точность, полученную алгоритмом при разном порядке загрузки групп кластеров.

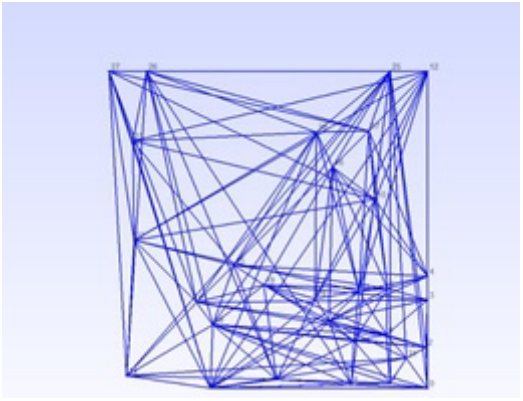

Рисунок 6. Структура, построенная DVA, для 30 точек.

Таблица 2. Точность алгоритма DVA для разных комбинаций кластеров.

FirstOption	SecondOption	ThirdOption	FourthOption	TotalTime(c.)	Accuracy(%)
A	B	C	A	31.51	0.97
A	C	B	A	30.27	0.94
B	A	C	A	29.47	0.94
B	C	A	A	27.41	0.94
C	A	B	A	30.89	0.85
C	B	A	A	27.75	0.84

5. Алгоритм на основе связности соседних ЦК

Данный алгоритм основан на прямом построении связей центров кластеров. Все точки и связи содержатся в структуре, представляющей из себя граф.

Для простоты объяснения, предположим, что в пространстве существует множество S_N из N центров кластера (далее просто ЦК) и они имеют правильные связи друг с другом (под правильным подразумевается то, что связи образованы этим же алгоритмом). Пусть, во множество S_N поместили новый ЦК – l. Пусть множество S_l – множество соседей l. Тогда

Множество соседних ЦК l определяется по следующему алгоритму:

Шаг 1. Для ЦК l вычисляется расстояние до всех ЦК. Переходим к шагу 2.

Шаг 2. Если множество S_l пусто, то в него добавляем ближайшую точку из S_N и переходим к шагу 1. Иначе из множества всех ЦК S_N выбирается ЦК $\acute{l}$ ближайшая к l. ЦК $\acute{l}$ исключается из множества S_N. Переходим к **шагу 3.**

Шаг 3. Если для любого ЦК, принадлежащего множеству S_l, расстояние до ЦК $\acute{l}$ больше, чем от l до $\acute{l}$, то $\acute{l}$ является соседнем ЦК и добавляется в множество S_l. Переходим к **шагу 4.**

Шаг 4. Если множество S_N является пустым, то множество S_l является искомым множеством соседних ЦК для l, если не является пустым, то переходим к **шагу 2.**

5.1. Реализация

На данный момент этот алгоритм имеет несколько реализаций. Друг от друга они отличаются лишь методом перепроверки связей и методом удаления.

Перепроверки связей:

1) *Без перепроверок.*
   Данная реализация не совершает никаких перепроверок связей после добавления ЦК в граф.
2) *Перепроверка соседства.*
   Данная проверка работает рекурсивно, однако, не редки случаи, когда эта рекурсия может стать бесконечной (из чего следует, что идеальную структуру, этим методом, создать невозможно). Смысл её в том, чтобы после добавления нового ЦК проверить, не помешало ли его добавление связям соседей, если так, то, нужно обновить связи у соседей, и т.д.
3) *Проверка пересечений.*
   Данная реализация, после добавления нового ЦК в граф проверяет наличие пересекающихся связей (рис 7.). Эта проверка не рекурсивная и затрагивает только соседей-соседей.

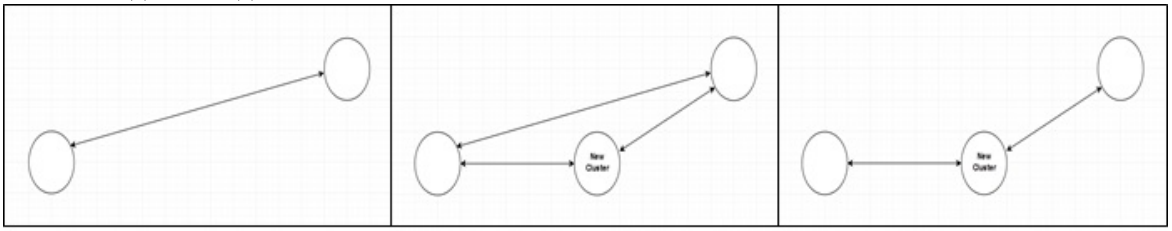

Рисунок 7. Пересекающиеся связи – и проверка пересечения.

Удаление ЦК

1) Классическое удаление.
   При данном варианте удаления, после удаления ЦК, у его соседей перепроверяются связи.
2) «Ломай и строй»
   При таком варианте удаления, помимо ЦК удаляются так же его соседи, затем, соседи вновь добавляются в граф.

Тестирование NRA проводилось с использованием всех возможных комбинаций этих алгоритмов.

Таблица 3. Итоговая таблица для алгоритма NRA.

Метод	Общее время	Точность
neighbour_relations_without_broke_and_build	10.08	0.79
neighbour_relations_crossing_broke_and_build	10.30	0.80
neighbour_relations_recheck_broke_and_build	10.10	0.79
neighbour_relations_without	10.08	0.79
neighbour_relations_crossing	10.31	0.80
neighbour_relations_recheck	10.10	0.79

Как можно увидеть из таблицы, различные методы не дают сильного преимущества в точности или времени. Все они находятся примерно в одном диапазоне. Стоить заметить, что построение структуры без каких-либо проверок порождает очень много лишних, шумных связей, а перепроверки позволяют устанавливать связи более точно и действенно. Так же, заметим, что NRA в три раза быстрее чем NNA. Количество ЦК в проведённых тестах всего лишь 30. Отсюда мы можем предположить, что алгоритмы, использующие связи соседства, намного быстрее классического подхода перебора всех ЦК.

6. Заключение

После проведение всех тестов, были получены следующие результаты (Таблица 4).

Таблица 4. Итоговые результаты эксперимента.

Алгоритм	Запуск	Первая группа	Загрузка файлов	Вторая группа	Третья группа	Удаление	Общее время	Точность
DPA	0.02	1.10	4.97	8.57	14.67	0.46	29.79	0.91
NNA	0.02	0.57	5.22	10.78	10.81	0.51	27.92	1
NRA	0.02	0.61	5.11	2.25	1.59	0.50	10.09	0.79
NRA_cross	0.02	0.62	5.14	2.12	1.93	0.48	10.32	0.80
NRA_recheck	0.02	0.61	5.12	2.25	1.59	0.50	10.10	0.79

Проанализировав результаты эксперимента, можем сделать несколько выводов.

1. Алгоритм NRA имеет прирост к скорости ~64%, и теряет в точности 21%. Так же следует учитывать, что такой результат был получен, при, всего лишь 30 ЦК.
2. Алгоритм DPA показал плохой результат, возможно его использование не целесообразно. Т.к. работает он дольше чем NNA, который мы хотим превзойти, и при этом теряет абсолютную точность.

7. Литература

[1] Визуализация данных [Электронный ресурс]. – Режим доступа: https://www.techrepublic.com/article/data-visualization-when-its-the-wrong-tool-for-the-job/.

[2] Smith, B.L. Comparison of parametric and nonparametric models for traffic flow forecasting / B.L. Smith, B.M. Wil-liams, R.K. Oswald // Transportation Research Part C: Emerging Technologies: Emerg. Technol. – 2002. – Vol. 10(4). – P. 303-321. DOI: 10.1016/S0968-090X(02)00009-8.

[3] Xia, D. A distributed spatial-temporal weighted model on MapReduce for short-term traffic flow forecasting / D. Xia, B. Wang, H. Li, Y. Li, Z. Zhang // Neurocomputing. – 2016. – Vol. 179. – P. 246-263. DOI: 10.1016/j.neucom.2015.12.013.

[4] Zheng, Z. Short-term traffic volume forecasting: a k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm / Z. Zheng, D. Su // Transportation Research Part C: Emerging Technologies. – 2014. – Vol. 43. – P. 143-157. – DOI: 10.1016/j.trc.2014.02.009.

[5] Кластерный анализ метод средних [Электронный ресурс]. – Режим доступа: http://cawi.fsocium.com/vortex/help/analyz/k_means.html.

[6] Триангуляция Делоне [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Delaunay_triangulation.

[7] Триангуляция Делоне. Определения, алгоритмизация, условия [Электронный ресурс]. – Режим доступа: https://neerc.ifmo.ru/wiki/index.php?title=%D0%A2%D1%80%D0%B8%D0%B0%D0%BD%D0%B3%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_%D0%94%D0%B5%D0%BB%D0%BE%D0%BD%D0%B5.

Development of data structuring algorithm to optimize the classification based on the method of the nearest neighbor.

A.V. Mukhin[1], R.A. Paringer[1]

[1]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086

Abstract. This article discusses the problem of the computationally complex classification algorithm used in data science, and proposes an approach to reducing the computational complexity of the classification procedure by optimizing the cluster enumeration procedure. Thus, the full search is replaced with a directional one. In addition, the article directly describes the structure construction algorithms that will be used for directional enumeration.