федеральное государственное бюджетное образовательное учреждение
высшего образования
«**Тольяттинский государственный университет**»

Институт **математики, физики и информационных технологий**
Кафедра «Прикладная математика и информатика»

01.04.02 ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему Математическая модель алгоритма k-means: исследование кластеризации и практическая реализация

Студент С. С. Шипилов _____

Научный
руководитель к.т.н., доцент кафедры, В. С. Климов _____

Руководитель магистерской
программы д.ф.-м.н., доцент, С.В. Талалов _____
«_____»_____20_____г.

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
«_____»_____20_____г.

Тольятти 2017

Оглавление

Введение

Существующие методы машинного обучения в зависимости от степени автоматизации поиска решений поставленной задачи, по мнению доктор наук Yoshua Bengio можно разделить на 4 типа: rule-based systems (экспертные системы, основанные на базе знаний); classic machine learning (классические алгоритмы машинного обучения); representation learning (обучение представлениям); deep learning (глубокое обучение) (рисунок 1.1).

Экспертные системы, основанные на базе знаний, обладают наименьшей степень автоматизации получения решения поставленной задачи. В этом случае качество работы такой системы зависит только от мастерства программиста, инженера по знаниям и эксперта (источника экспертных знаний). Алгоритмы глубоко машинного обучения наоборот обладают высокой степенью автоматизации при работе с данными. Такие алгоритмы самостоятельно анализируют исходные данные, выделяют существенные признаки данных, вырабатывают стратегию решения поставленной задачи. Классические алгоритмы машинного обучения и обучение представлениям обладают средней степенью автоматизации при работе с данными.

В настоящее время проводятся исследования по повышению степени автоматизации алгоритмов машинного обучения. Например, на начальном этапе своего развития нейронные сети нейронные сети с малым количеством слоев относились к классическим алгоритмам машинного обучения. Однако с увеличением количества вычислительных слоев и совершенствованием способов их обучения стали появляться конфигурации нейронных сетей, относящихся к глубокому обучению.

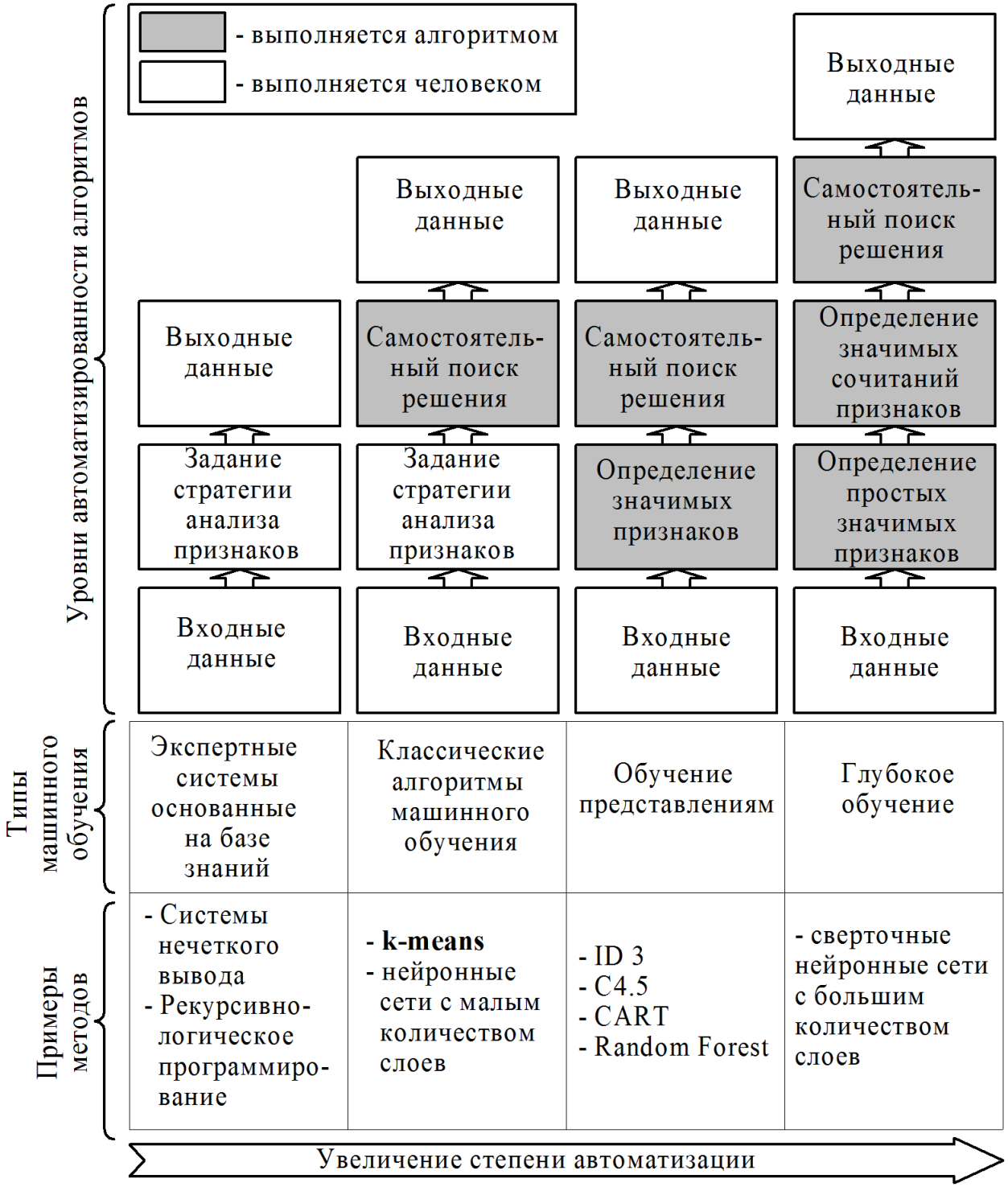


Рисунок 1.1 – Анализ алгоритмов машинного обучения

В машинном обучении существует алгоритм, предназначенный для кластеризации данных – k-means. Он автоматически анализирует исходные данные и подбирает оптимальную кластерную структуру. Одной из главных его проблем является возможность попадания алгоритма в локальное решение (вместо глобального). Это связано с тем, что начальное расположение кластеров (которое, впоследствии выполнения алгоритма, уточняется) выбирается случайным образом.

Гипотезой исследования является предположение, что степень автоматизации, точность и скорость работы алгоритма k-means можно повысить путем разработки метода по расчету начального расположению центров кластеров.

Исследованиями в области совершенствования алгоритмов машинного обучения занимаются такие ученые как: Langley P., Carbonell J., Ding S., Zong W., Zhu C., Chen Z., Blum A., Bing L., Günnemann S., Langley P., Abdolrazzaghi M., Carbonell J., Shen Q., Blocki M.M., Celebe H.M., Wuy H., Liotte D., Mucherin B., Mirkins M., Tryone C.B., Li X., Alom M.Z., Mirza B., Ertuğrul Ö.F., Stein G., Li B., Perlich C., Chang K.C., Dietterich T.G., Carbonell J., Tamura H., Langlei M., Wiens J. E., Zhou ZH., Hemmi H. L., Webb G.I., Zhang D., Nayak P.K. и др.

Объектом исследования являются кластеризация данных, предметом исследования – разработки метода по расчету начального расположению центров кластеров.

Цель исследования – повышение степени автоматизации алгоритма k-means путем разработки метода по расчету начального расположению центров кластеров.

Цель достигается путем решения следующих задач:

1. Проведение анализа состояния вопроса по теме исследования

2. Разработка метода по определению начальных положений центров кластеров (при анализе данных с помощью алгоритма k-means).

3. Разработка программной реализация данного метода для проверки его эффективности на практике.

4. Тестирование предложенного метода на практике. Формулирование выводов по результатам вычислительных экспериментов.

Научная новизна исследования – доказано что скорость и точность результатов кластеризации данных с помощью алгоритма k-means может быть увеличена путем обоснованного задания начального положения центров кластеров. Это также приведет к снижению вероятности схождения алгоритма к локальному решению.

Практическая значимость работы заключается в разработке методики выбора начального положения центров кластеров, что позволяет увеличить точность и скорость кластеризации данных с использованием алгоритма k-means.

В первой главе настоящего исследования рассматриваются проблемы развития алгоритмов машинного обучения. Также рассмотрены проблемы кластерного анализа с помощью алгоритма k-means.

Во второй главе рассматривается формальное описание задачи кластеризации данных. Описан математический аппарат EM-алгоритмов кластеризации Затем дается описание метода по определению оптимального начального положения центров кластеров при кластеризации данных с использованием алгоритма k-means. Далее даются результаты вычислительных экспериментов по применению предложенного метода в сравнении с классической реализацией алгоритма.

В третьей главе дается описание разработанного программного обеспечения. Затем приведен алгоритм работы с, созданным в рамках данного исследования, приложением. И описывается реализация модуля вычислительных экспериментов.

На защиту выносятся:

1. Методика по расчету начального положения центров кластеров.
2. Результаты апробации предложенной методики при

кластеризации данных.

По результатам проведенных исследований опубликовано 5 статей в сборниках, входящих в РИНЦ.

Результаты исследований доложены на таких конференциях как:

- XXXI студенческой международной научно-практической конференции, г. Москва.
- Научно-практической конференции "Студенческие дни науки в ТГУ", г. Тольятти
- III научно-практической всероссийской конференции (школе-семинаре) молодых ученых «Прикладная математика и информатика: современные исследования в области естественных и технических наук».

1 Анализ состояния вопроса

1.1 Проблемы развития алгоритмов машинного обучения

Понятию машинное обучение (Machine Learning) существует множество определений. Наиболее часто используемое определение говорит, что это множество методов, особенностями которых является решение задач не напрямую, а обучение решению текущей задачи на примере множества сходных задач. В разработке методов машинного обучения используются средства математической статистики, различные численные методы, а также методы оптимизации и вероятностные подходы.

В настоящее время с усложнение методов машинного обучения стали наборы алгоритмов относящихся к классу глубокое обучение (Deep Learning).

Основным отличие данного класса алгоритмов от классических методов машинного обучения является, то, что их работа основана на изучении множества уровней представлений (множество уровней абстракций). Т.е. алгоритмы глубоко обучения, позволяют определять объекты сложной конфигурации на основе данных различных уровней абстракции.

На основании выше сказанного доктор наук Yoshua Bengio в своём учебном курсе под названием Deep Learning: Theoretical Motivations предлагает следующую классификацию алгоритмов машинного обучения (рисунок 1.2).

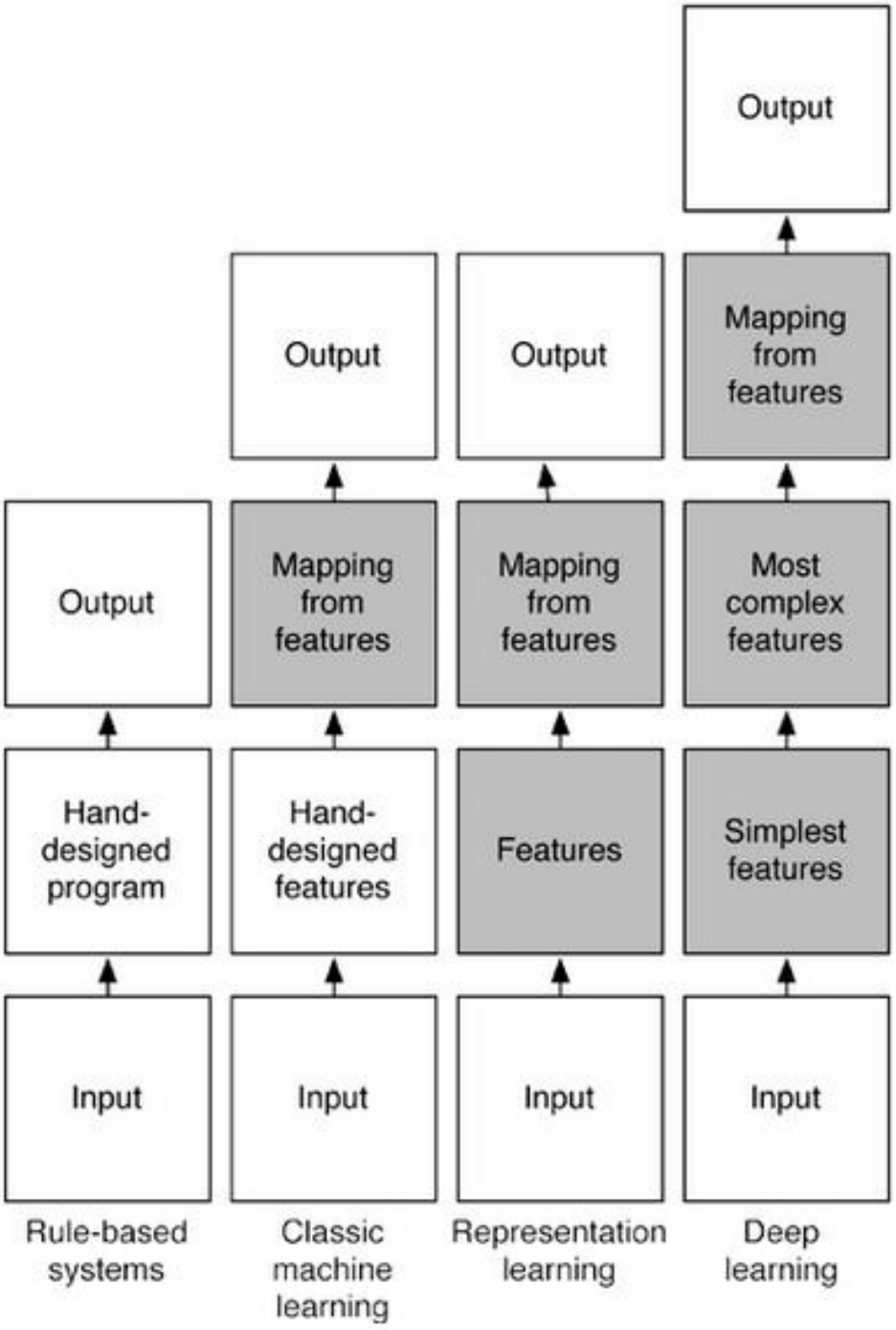


Рисунок 1.2 – Классификация алгоритмов машинного обучения, предложенная Yoshua Bengio

На рисунке 1.2 показано взаимодействие уровней абстракций для четырех типов множеств методов машинного обучения:

- rule-based systems (экспертные системы основанные на базе знаний);
- classic machine learning (классические алгоритмы машинного обучения);
- representation learning (обучение представлениям);
- deep learning (глубокое обучение).

Необходимо отметить, что на схеме (рисунок 1.2) серым цветом выделены блоки, способные (без вмешательства человека) обучаться по имеющимся данным. А белым цветом выделены блоки, которые человеку необходимо программировать и анализировать вручную.

Рассмотрим особенности экспертных систем (rule-based systems)

Экспертные системы, основанные на базе знаний, представляют из себя компьютерную программу. Причем, то как будет функционировать экспертная систем определяется человеком вручную. Поэтому, для создания экспертной системы ну жен эксперт из рассматриваемой области, который будет представлять из себя источник знаний и программист, который будет формализовывать эти знания в виде программного кода. Для того, чтобы сгенерировать ответ на поставленный вопрос, экспертная система использует факты, хранящиеся в базе знаний с учетом правил логических правил.

Рассмотри особенности классического машинного обучения (classic machine learning).

При построение программ с использование классических алгоритмов машинного обучения наиболее важные указания вводятся человеком вручную. Но при этом программа самостоятельно обучается на основе входных признаков определять требуемое решение. Данный класс машинного обучения может решать задачи распознавания простых объектов. При создании интеллектуальных систем, основанных на использовании

данного типа машинного обучения, наиболее трудоемкими являются задачи подбора обучающей выборки и определение наиболее удачного сочетания параметров используемого метода.

Рассмотрим особенности обучения представлениям (Representation Learning).

Если сравнивать классические методы машинного обучения и обучениям представлениям, то последние являются более автоматизированными, т.к. избавляют от необходимости формализации знаний экспертов. Поиск значимых закономерностей в этом случае осуществляется автоматически на основании предъявленной человеком обучающей выборке.

Рассмотрим особенности глубокого обучения (Deep Learning).

Методы глубоко обучения отличаются от других методов, что входные данные анализируются на нескольких уровнях абстракции. На самом нижнем уровне в результате анализа данных выделяются простые признаки, на следующем уровне производится анализ простых признаков с целью выделения более сложных признаков и т.д. до самого верхнего уровня абстракции.

Ярким представителем методов глубокого обучения являются сверхточные нейронные сети, которые применяются для поиска объектов изображениях. Такая нейронная сеть является многослойной. Каждый слой, в данном случае, отвечает за свой уровень абстракции. Слои, близкие ко входам нейронной сети отвечают за поиск простых признаков присутствия объекта на изображении, слои близкие к выходам нейронной сети отвечают за анализ простых признаков с целью формирования вывода о присутсвии искомого объекта на изображении.

Исследователи искусственного интеллекта отмечают три основных компонента необходимые для его создания:

1. Наличие огромных объемов данных.
2. Использование гибких моделей для их анализа.

3. Наличие априорных знаний.

Таким образом, для создания систем искусственного интеллекта требуется большое количество знаний. Для формализации знаний из имеющихся данных на раннем этапе развития искусственного интеллекта применялся человеческий труд. Сейчас с этой задачей успешно справляются алгоритмы машинного обучения. Знания необходимы, чтобы система искусственного интеллекта могла принимать верные решения в любых ситуациях.

Одних данных для создания интеллектуальных систем недостаточно. Имеющиеся данные еще необходимо правильно использовать, для того, что принимать верные решения. Для этого необходимы гибкие модели анализа данных. Помимо анализа, модели выполняют еще и задачу хранения имеющихся знаний.

Наличие априорных знаний позволяет частично решить проблему большого размера избыточных знаний. Для этого системе искусственного интеллекта сообщаются только те априорные знания, которые необходимы для решения возложенных на нее задач.

Непараметрические методы умеют работать с большими объемами данных, обдают достаточно гибкими моделями, однако их использование требует обеспечение процедуры сглаживания данных.

Для того, чтобы искусственный интеллект начал понимать окружающую обстановку в него необходимо вложить знания об мире. Основная проблема заключается в то, что существующие методы машинного обучения не позволяют хранить в себе такой объем знаний.

Тем не менее, все алгоритмы машинного обучения объединяет общая особенность – обучаемость. Это означает, что на основе данных частных случаев алгоритмы машинного обучения самостоятельно могут получать знания.

Также алгоритмы машинного обучения обладают способностью общения. Это означает, что они способы определять, какой результат является наиболее вероятным при данном наборе входных сигналов. Например, с точки зрения кластеризация – это предсказывание центра масс кластеров.

Одной из проблем практического применения алгоритмов машинного обучения является большая размерность вектора входных сигналов, к которая приводит к повышению сложности функций, необходимых для обобщения данных обучающей выборки. При количестве элементов вектора входных сигналов, равном двум – количество вариантов конфигураций обещающих функций значительно. Если количество элементов вектора входных сигналов больше двух, то даже просто посчитать количество возможных конфигураций обещающих функций затруднительно.

Другой проблемой алгоритмов машинного обучения является понимание данных. Так классические алгоритмы машинного обучения манипулирует данными, как обычными числами, не задумываясь о природе их появления. В этом отношении алгоритмы глубокого обучения делают шаг навстречу к пониманию программой природы данных.

В теории искусственного интеллекта в понятие непараметрических моделей вкладывается иной смысл, нежели в математическом моделировании. Здесь алгоритм машинного обучения называется непараметрическим, если сложность функций, которые он способен изучить, растет с ростом объема обучающей выборки (т.е. размерность вектора входных сигналов не является фиксированной).

С учетом данных содержащихся в обучающей выборке возможен выбор семейства функций, которые будет более или менее точно обещать имеющейся данные. Например при обучение линейного классификатора SVM увеличение размерности обучающей выборки не приведет к изменению модели. А при использовании алгоритмов построения деревьев принятия

решений, наоборот – увеличение обучающей выборки приводит к усложнению правил классификации данных.

Таким образом, понятие непараметрический обозначает отсутствие фиксированного набора параметров описывающих обучающую выборку. Т.е. в зависимости от характера данных мы сами можем выбирать, какие параметры будут использоваться при обучении.

Проблема многомерности связана с большим количеством вариантов конфигураций используемой модели. Чем больше измерений у входных данных – тем больше вариантов конфигураций. Причем количество возможных конфигураций увеличивается от размерности данных по экспоненте.

Стандартные подходы в непараметрической статистике основаны на использовании интерполяции. Стоит отметить, что такой подход хорошо работает на малом количестве данных. Однако при большом количестве данных такой подход теряет свою эффективность, так как в конечный результат могут попасть не все примеры обучающей выборки.

Для локального обобщения, необходимы примеры представлений для всех релевантных исходов. Очевидно, что с использование локального усреднения в большинстве случаев получаемые результат общения данных обучающей выборки неудовлетворителен.

С точки зрения математики число измерений – количество различных вариаций изучаемых функций. Тогда понятие равномерности будет относиться к количеству впадин и выпуклостей, представленных на кривой.

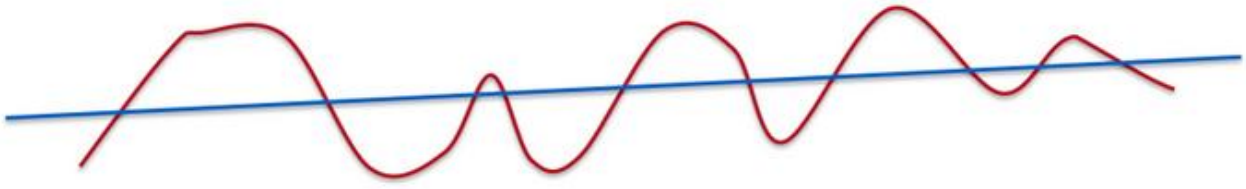


Рисунок 1.3 – Графическая интерпретация сглаживания

Можно представить, что линия – предельный случай «гладкости кривой». Кривая с несколькими выпуклостями и впадинами по-прежнему гладкая, но уже в меньшей степени (рисунок 1.3).

Используемые в моделях машинного обучения функции не являются гладкими. При реализации систем компьютерного зрения или обработки натурального языка, функции используемые в моделях машинного обучения имеют очень сложный вид.

Многие непараметрические статистические методы полагаются на гауссову функцию влияния, усредняющую значения в определенной небольшой области. Однако ядерным машинам, использующим распределение Гаусса, требуется как минимум k примеров для изучения функции, имеющей 2k пересечений оси x. С ростом числа измерений количество выпуклостей и впадин увеличивается по экспоненте. Можно получить очень негладкую функцию даже в одном измерении.

Если подходить к проблеме с точки зрения геометрии, то мы должны поместить вероятностную меру туда, где структура наиболее правдоподобна. В случае эмпирической функции распределения мера попадает на тренировочные примеры. Рассмотрим иллюстрацию выше, где представлены двумерные данные. Если считать график гладким, то вероятностная мера оказывается равномерно распределенной между примерами.

Округлые фигуры изображают Гауссовы ядра для каждого примера. К такому подходу прибегают многие непараметрические статистические

методы. В случае двух измерений все выглядит достаточно просто и реализуемо, однако с увеличением числа измерений размеры кругов (шаров) увеличиваются настолько, что перекрывают собой все пространство или же, наоборот, оставляют пустые места там, где должна находиться максимальная вероятность. Поэтому не стоит надеяться на гладкость функции и нужно использовать более эффективные методы, обладающие своей структурой.

Такой структурой может являться одномерное многообразие, где собрана вероятностная мера. Если мы сможем определить представление вероятности, то решим наши проблемы. Представление может быть меньшей размерности или располагаться вдоль других осей в том же измерении. Мы берем сложную нелинейную функцию и встраиваем её в Евклидово пространство, изменяя представление. Так проще делать предсказания, проводить интерполяцию и осуществлять оценку плотности распределения.

Гладкость была главным требованием в большинстве непараметрических методов, но довольно очевидно, что с её помощью мы не можем побороть проблему размерности. Мы хотим, чтобы гибкость семейства функций росла при увеличении количества данных. В нейронных сетях мы изменяем количество скрытых элементов в зависимости от объема данных.

Наши модели машинного обучения должны стать композиционными. Натуральные языки используют композиционность, чтобы представлять более сложные идеи. В глубоком обучении используются: распределенные представления; глубокая архитектура.

Предположим, что данные поступают к нам не все сразу, а партиями. Партии могут поступать или параллельно, или последовательно. Параллельное поступление партий являет собой распределенное представление (обучение представлениям). Последовательное поступление данных похоже на обучение представлениям, но с несколькими уровнями. Композиционность дает нам возможность эффективного описания мира вокруг нас.

Среди методов, не использующих распределенные представления, можно выделить кластеризаци., метод N-грамм, метод ближайших соседей, метод радиальных базисных функций и опорных векторов, а также метод дерева решений. На вход таких алгоритмов подается пространство, а на выходе получаются области. На выходе некоторых алгоритмов получаются строгие разделения, а на выходе других нестрогие, позволяющие гладкую интерполяцию между близлежащими областями. Каждая область обладает своим набором параметров.

Результат, который сообщает область, и её местоположение регулируются данными. С количеством областей связано понятие сложности. С точки зрения теории обучения, обобщение зависит от отношения числа необходимых примеров и сложности. Богатая функция требует большего количества областей и данных. Существует линейная зависимость между числом различимых областей и числом параметров. Также существует линейная зависимость между числом различимых областей и числом тренировочных примеров (рисунок 1.4).

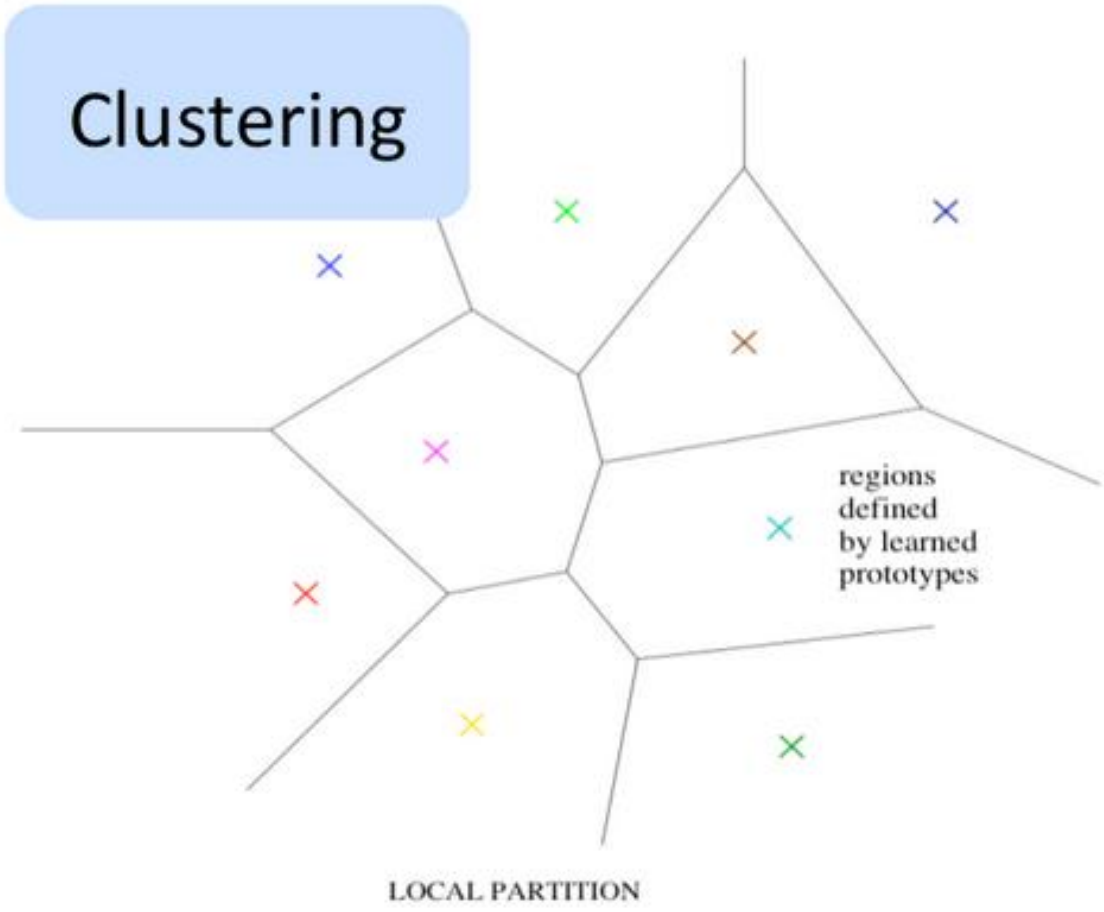


Рисунок 1.4 – Локальная кластеризация

Рассмотрим другой вариант. Возможно представить экспоненциальное количество областей линейным набором параметров, используя распределенные представления. Достоинством распределенного представления заключается в том, что с его помощью можно изучить очень сложную функцию (с множеством выпуклостей и впадин), имея в распоряжении небольшое количество примеров (рисунок 1.5).

В нераспределенных представлениях количество параметров находится в линейной зависимости от числа областей. Отсюда следует, что количество областей может расти экспоненциально числу параметров и примеров. В распределенных представлениях отдельные признаки имеют значение, не зависящее от других признаков. Различные корреляции имеют право на существование, но большинство признаков изучаются вне

зависимости друг от друга. Нам не обязательно знать все конфигурации, чтобы принять правильное решение. Несовместные признаки создают большой комбинаторный набор различных конфигураций. Все преимущества видны при использовании всего одного слоя – количество примеров может быть очень маленьким.

Однако такое преимущество не наблюдается на практике – оно большое, да – но не экспоненциальное. Если представления хорошие, то они разворачивают многообразие в новой плоской системе координат. Нейронные сети преуспели в изучении представлений, затрагивающих семантические аспекты. Обобщение образовывается на основании таких представлений. Если пример находится в пространстве, где нет никаких данных, то непараметрическая система не сможет сказать о нем ничего. Используя распределенные представления, мы можем делать выводы о том, что никогда не видели. Это суть обобщения.

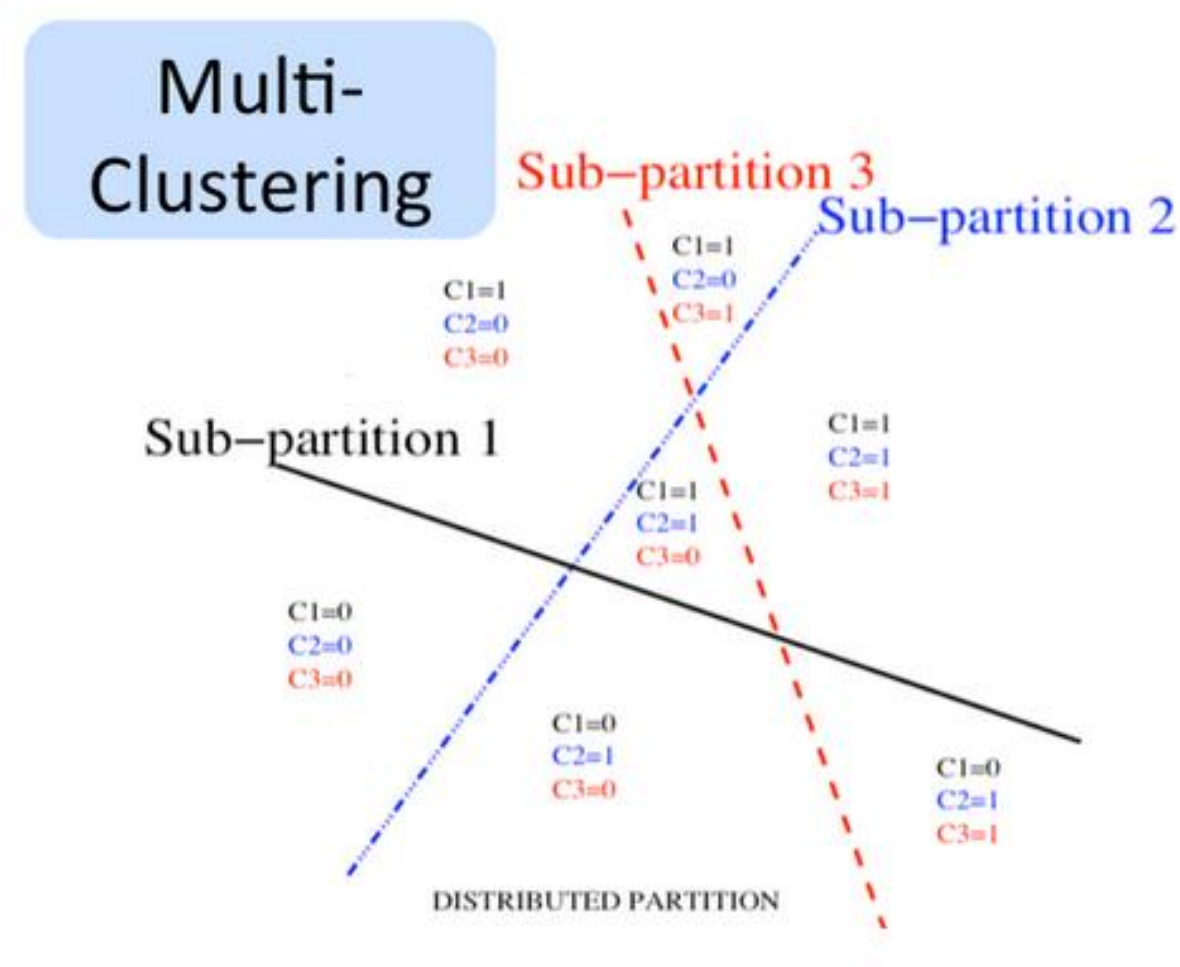


Рисунок 1.5 – Распределенная кластеризация

Рассмотрим классический символьный искусственный интеллект и обучение представлениям. Распределенные представления лежат в основе коннекционизма или коннеционистского подхода, зародившегося в 1980 году. Классический подход основан на понятии символов. В символьной обработке таких вещей, как язык, логика, или правила, каждая концепция ассоциирована с определенной сущностью – символом. Символ или существует, или нет.

Нет ничего, что бы определяло связи между ними. Обратимся к примеру с собакой и кошкой. В символьном искусственном интеллекте – это два разных символа, не имеющие никакой взаимосвязи между собой. В распределенном представлении они имеют схожие особенности, к примеру, они являются питомцами, имеют 4 лапы и так далее. Можно расценивать эти

концепции как шаблоны признаков или шаблоны активизации нейронов в мозгу.

С помощью распределенных представлений удалось добиться очень интересных результатов в обработке естественных языков.

Исследование глубоких нейронных сетей не проводилось раньше, поскольку исследователи считали, что в них нет нужды. Плоская нейронная сеть с одним слоем скрытых элементов способна представить любую функцию с заданной степенью точности.

Это свойство называется универсальной аппроксимацией. Однако в этом случае мы не знаем, сколько скрытых элементов нам понадобится. Глубокая нейронная сеть позволяет значительно уменьшить количество скрытых элементов – снизить стоимость представления функции. Если мы пытаемся изучить глубокую функцию (имеющую множество уровней композиции), то нейронной сети понадобится большее число слоев.

Глубина не является необходимостью – не имея глубоких сетей, мы по-прежнему можем получить семейство функций с достаточной гибкостью. Более глубокие сети не обладают большей емкостью. Глубже – не означает, что мы можем представить больше функций. Использовать глубокую нейронную сеть стоит тогда, когда изучаемая нами функция обладает определенными характеристиками, ставшими результатом композиции нескольких операций.

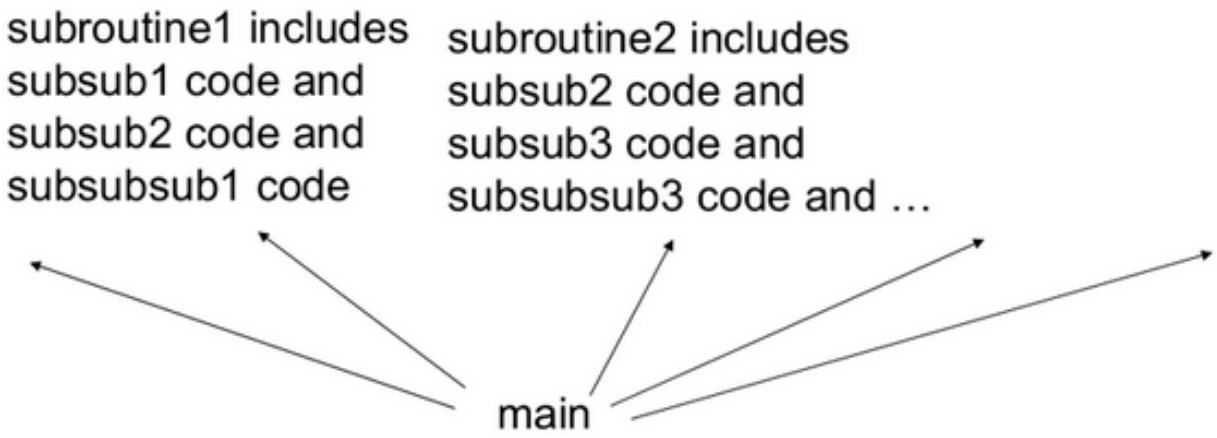


Рисунок 1.6 – Архитектура обычной компьютерной программы

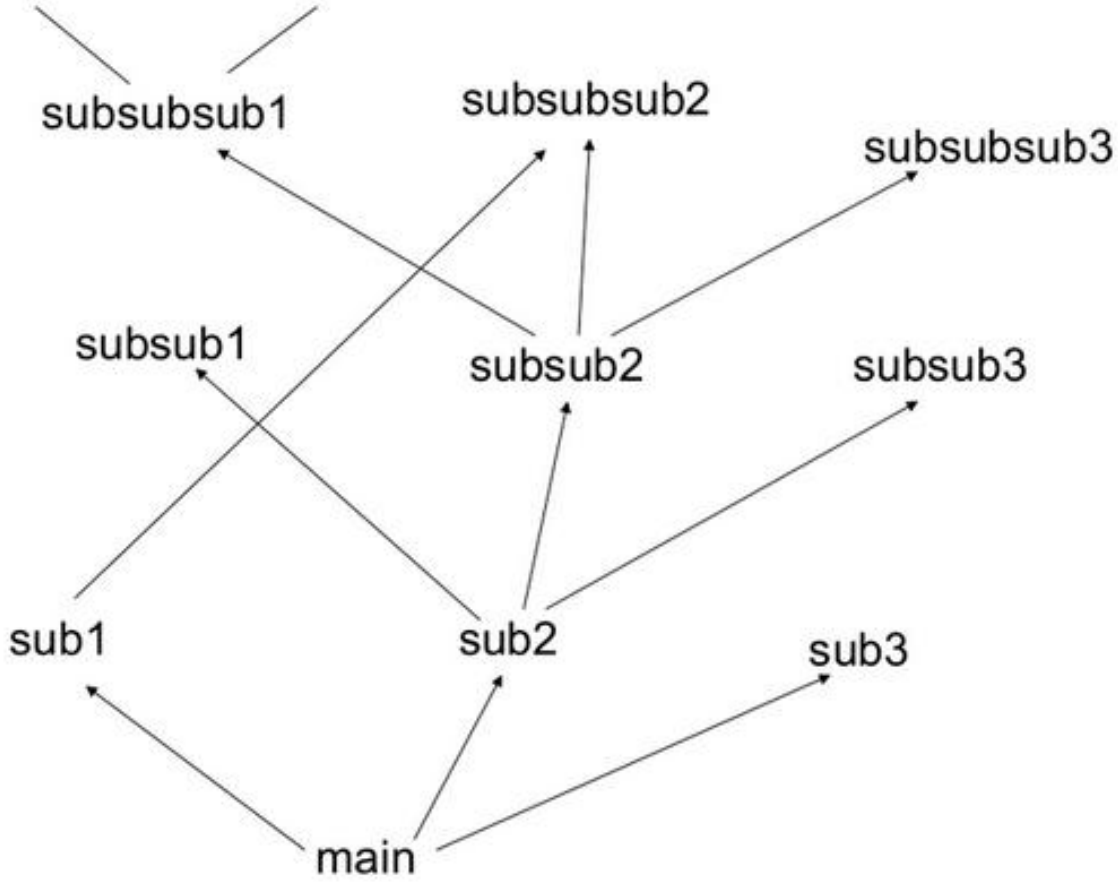


Рисунок 1.7 – Архитектура компьютерной программы для анализа данных на разных уровнях абстракции.

При написании компьютерных программ мы не располагаем все строки кода друг за другом – обычно мы используем подпрограммы. Скрытые элементы выступают в качестве подпрограммы для большой программы – финального слоя. Можно считать, что результат вычислений каждой строки программы меняет состояние машины, одновременно передавая свой вывод другой строке. Каждой строке на вход подается состояние машины, а на выходе строки появляется уже другое состояние (рисунки 1.6, 1.7).

Это похоже на машину Тьюринга. Количество шагов, выполняемых машиной Тьюринга, зависит от глубины вычислений. В принципе, существует возможность представить любую функцию за два шага (таблица поиска), однако это не всегда эффективно. Ядерный метод опорных векторов или плоская нейронная сеть могут рассматриваться как таблица поиска.

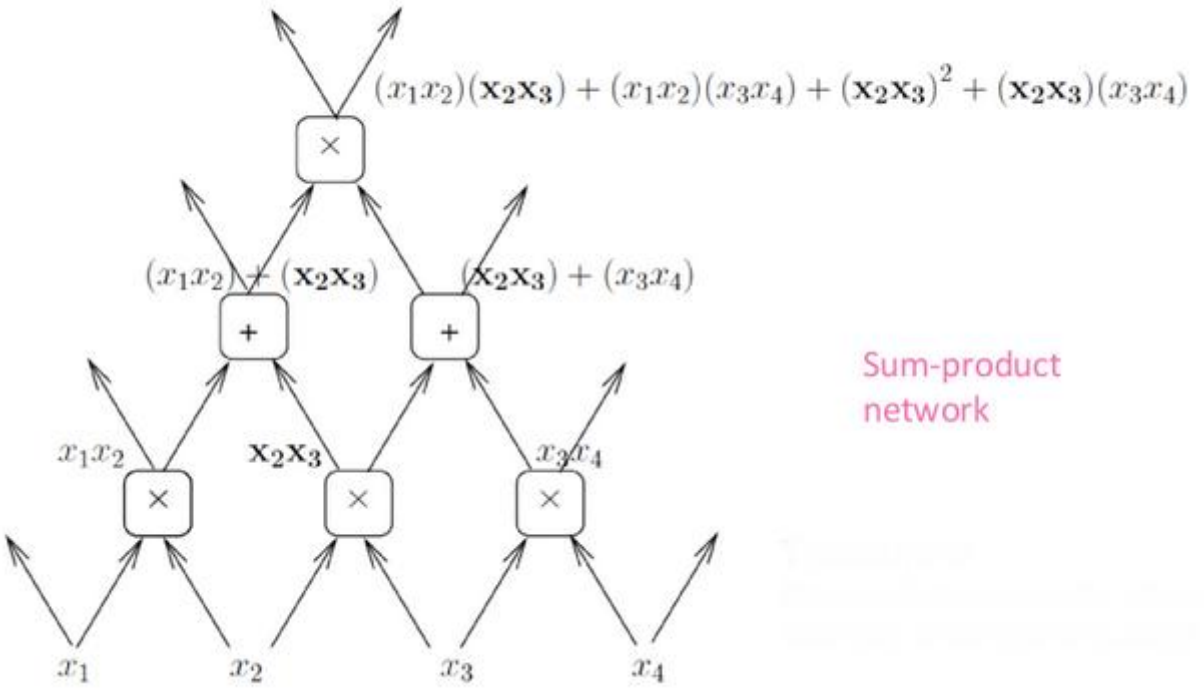


Рисунок 1.8 – Реализация сети основанная на разделяемых компонентах данных

Полиномы часто представляются как суммы произведений (рисунок 1.8). Еще одним способом представления является граф вычислений, где каждая вершина производит сложение или умножение. Подобным образом мы можем представить и глубокие вычисления – так количество вычислений уменьшится, поскольку мы сможем повторно использовать некоторые операции.

Однако глубокие сети с вентильными нейронными модулями гораздо дороже элементов плоских сетей, поскольку они способны разделять пространство на большее количество линейных областей (с условиями).

Иллюзия выпуклости

Одной из причин отвержения нейронных сетей в 90-х годах стала невыпуклость проблемы оптимизации. С конца 80-х и начала 90-х годов мы знаем, что в нейронных сетях имеется экспоненциальное количество локальных минимумов. Это знание, наряду с успехом ядерных машин в 90-х годах, сыграло свою роль и сильно снизило интерес многих исследователей нейронных сетей.

Они считали, что раз оптимизация является невыпуклой, то нет никаких гарантий нахождения оптимального решения. Более того, сеть могла зациклиться на плохих, неоптимальных решениях. Исследователи изменили свое мнение совсем недавно. Появились теоретические и эмпирические доказательства того, что проблема невыпуклости – вовсе не проблема. Это поменяло все наше представление об оптимизации в нейронных сетях.

Давайте рассмотрим проблему оптимизации на малом и большом количестве измерений. В малых измерениях существует множество локальных минимумов. Однако в высоких измерениях локальные минимумы не являются критическими точками (точками интереса). Когда мы оптимизируем нейронные сети или любую другую функцию нескольких переменных, для большинства траекторий критические точки (точки, где производная равняется нулю или близка к нему) являются седловыми. Седловые точки являются неустойчивыми.

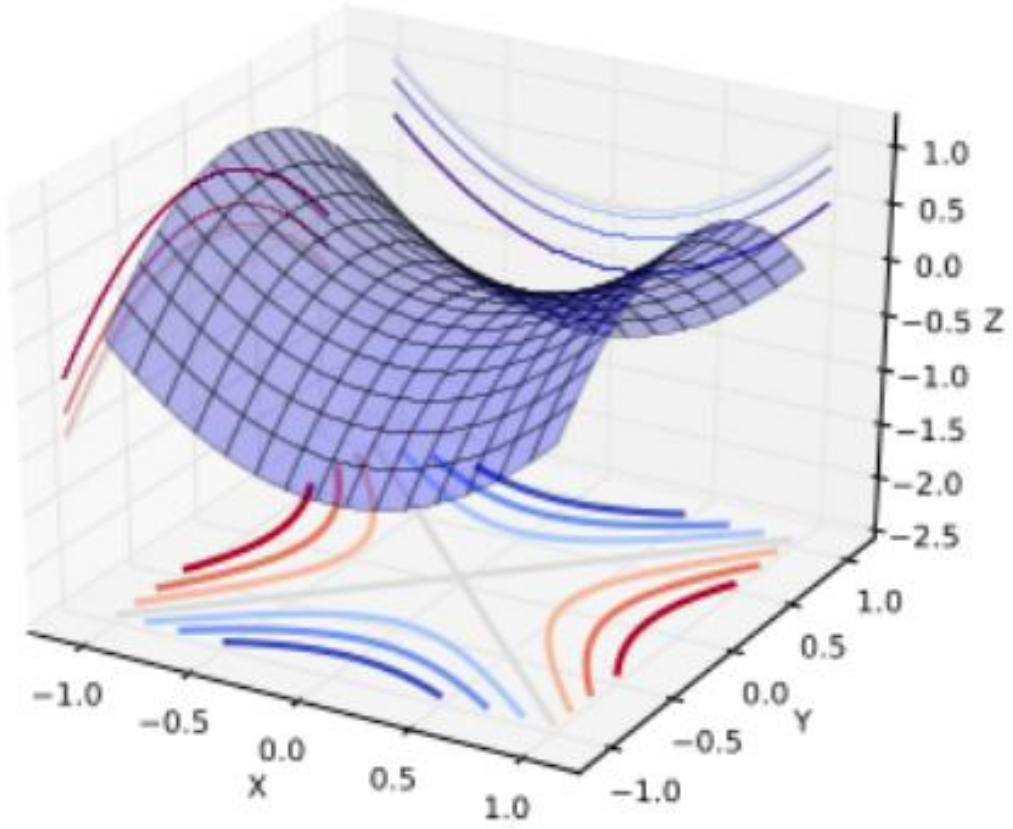


Рисунок 1.9 – Седлова точка (0;0)

На изображении выше как раз приведена седловая точка (рисунок 1.9). Если точка является локальным или глобальным минимумом, то при движении от неё во всех направлениях будет наблюдаться увеличение значений функции (при движении от локального максимума – уменьшение). При наличии фактора случайности в задании функций или при независимом выборе направления движения, крайне маловероятно, что функция будет возрастать во всех направлениях в других точках, кроме глобального минимума.

Интуитивно понятно, что если мы обнаружили минимум, который близок к глобальному, то функция будет возрастать во всех направлениях – ниже этой точки опуститься нельзя. Статистическая физика и матричная теория предполагают, что для некоторых семейств функций (достаточно

большого их количества), наблюдается концентрация вероятности между индексом критических точек и целевой функцией.

Индекс – это коэффициент направления, определяющий, в каком направлении происходит уменьшение значения функции. Если индекс равен нулю – это локальный минимум, а если индекс равен единице – это локальный максимум. Если индекс равен числу между нулем и единицей, то это седловая точка. Таким образом, локальный минимум – это частный случай седловой точки, индекс которой равняется нулю. Чаще всего встречаются именно седловые точки. Эмпирические результаты показывают, что, действительно, между индексом и целевой функцией существует сильная связь.

Это лишь эмпирическая оценка, и нет доказательств того, что результаты подходят для оптимизации нейронных сетей, однако подобное поведение соответствует теории. На практике видно, что стохастический градиентный спуск всегда будет «покидать» поверхности, где нет локального минимума.

Люди способны делать выводы на основании очень небольшого количества примеров. Дети обычно учатся делать что-то новое на небольшом количестве примеров. Иногда даже на основании одного, что статистически невозможно. Единственное объяснение – ребенок использует какие-то знания, полученные им до этого. Априорные знания могут быть использованы для построения представлений, благодаря которым, уже в новом пространстве, появляется возможность сделать вывод на основании всего одного примера. Человек в большей степени опирается на априорные знания.

Частичное обучение – это нечто среднее между обучением с учителем и без учителя. В обучении с учителем мы используем только маркированные примеры. В частичном обучении мы дополнительно используем немаркированные примеры. На изображении ниже показано, как частичное

обучение может найти лучшее разделение, используя неразмеченные примеры (рисунок 1.10).

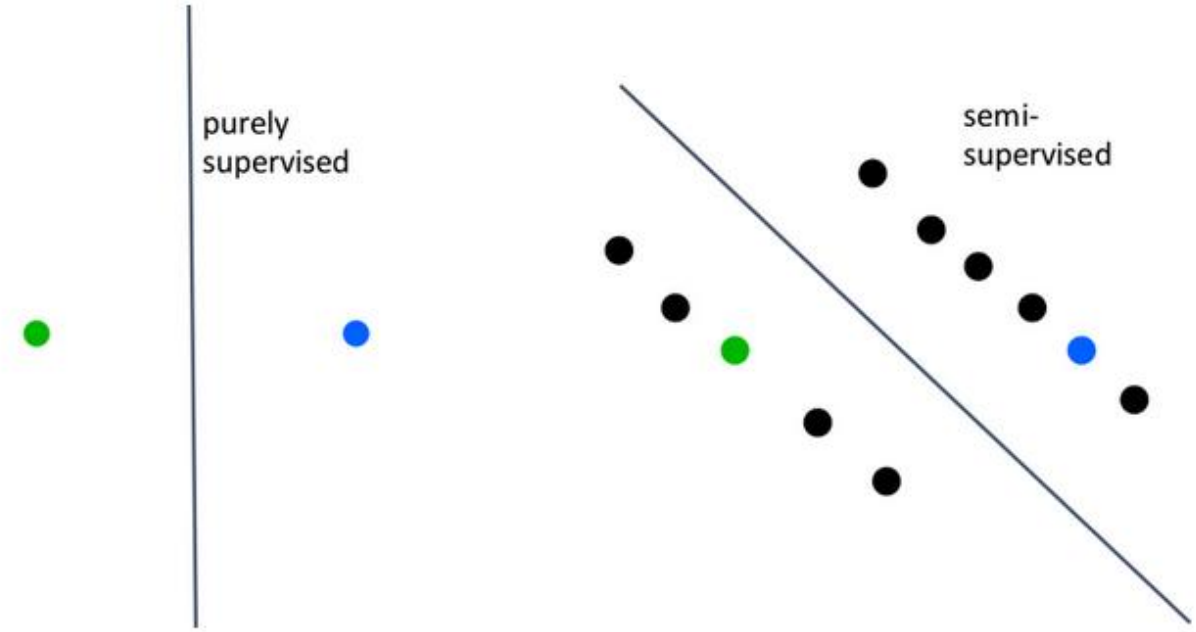


Рисунок 1.10 – Использование немаркированных объектов обучающей выборки

Приспособление к решению нового типа задач – это очень важный момент в разработке искусственного интеллекта. Априорные знания – это связывающие знания. Глубокие архитектуры учат промежуточные представления, которые могут быть разделены между заданиями. Хорошие представления, увеличивающие коэффициент вариации, применимы для решения множества задач, поскольку каждое задание учитывает поднабор признаков (рисунок 1.11).

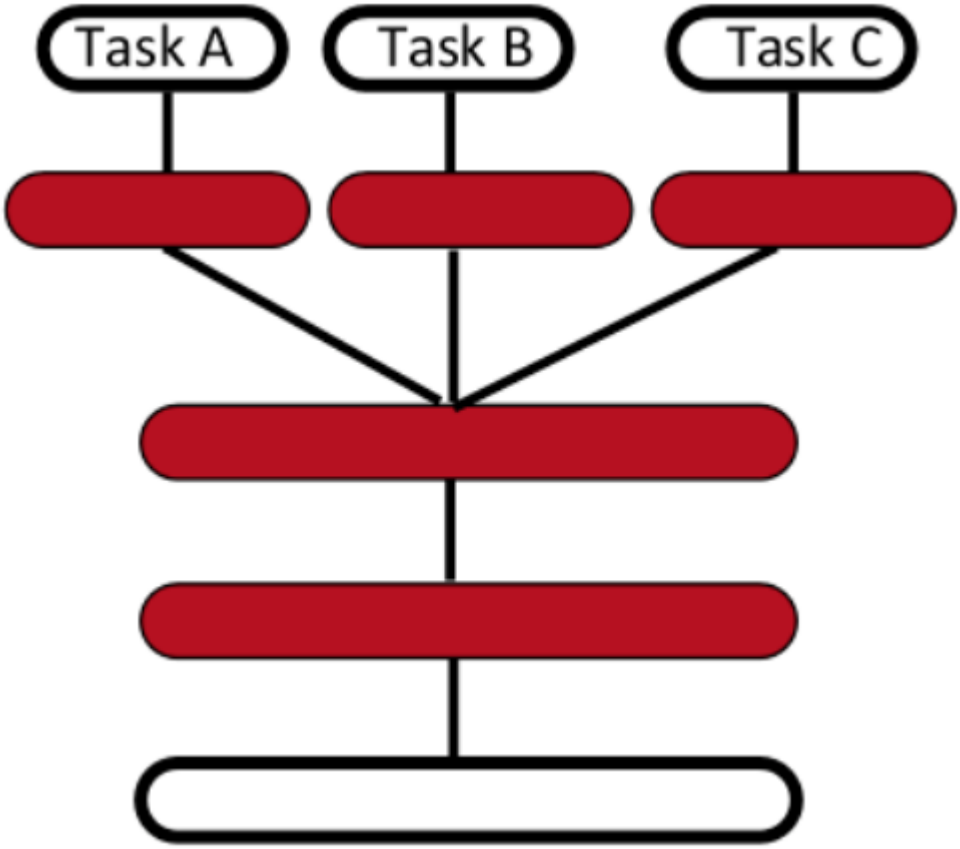


Рисунок 1.11 – Схема многозадачного обучения

Следующая схема иллюстрирует многозадачное обучение с различными входными данными (рисунок 1.12)

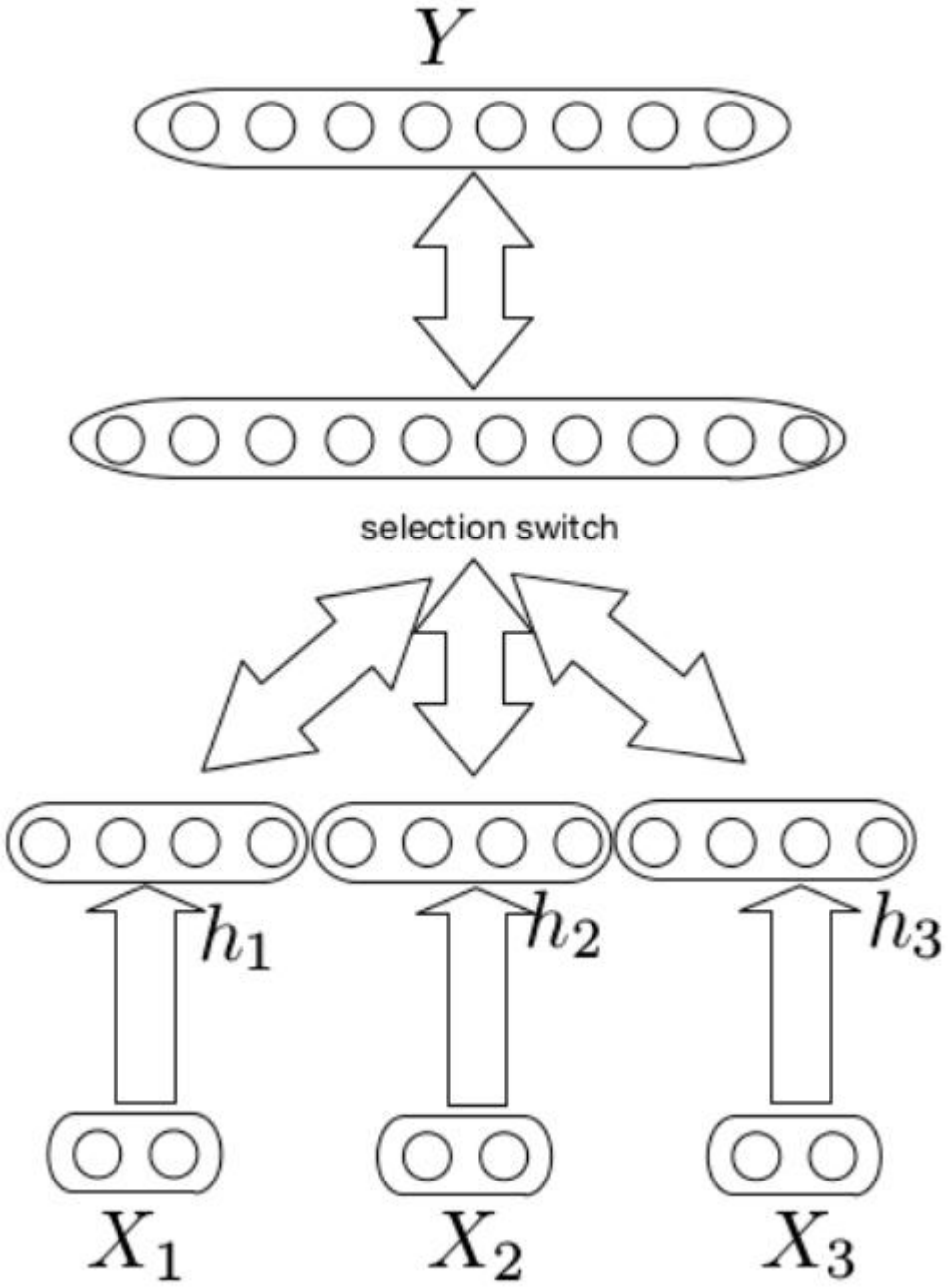


Рисунок 1.12 – Многозадачное обучение с различными входными данными

Глубокое обучение позволяет изучать большие уровни абстракции, увеличивающие коэффициент вариации, что упрощает процесс обобщения (рисунок 1.13).

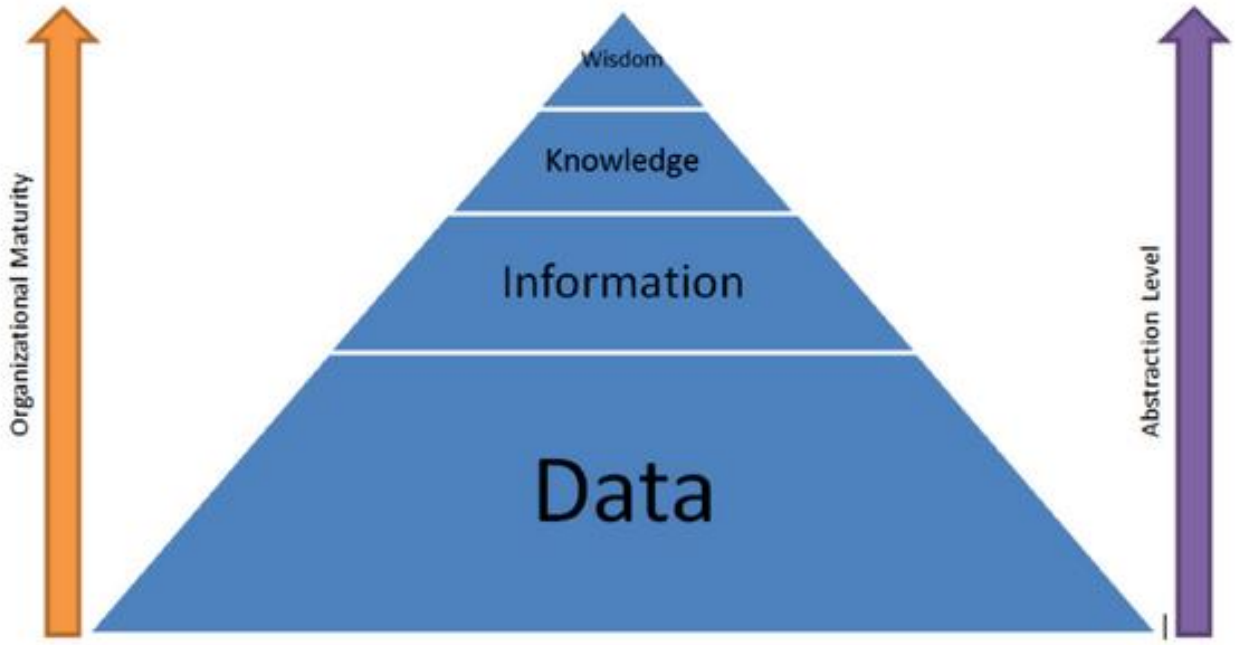


Рисунок 1.13 – Различные уровни абстракции данных

На основе приведенного обзора можно сделать следующие выводы:

- распределенное представление и глубокая композиция позволяют значительно улучшить способности к обобщению;
- распределенное представление и глубокая композиция выдают нелокальное обобщение;
- необходимо пользоваться другими методами, такими как частичное обучение и многозадачное обучение, способными лучше обобщать глубокие распределенные представления.

1.2 Проблемы кластерного анализа

Кластеризация – это процедура сбора и анализа информации об обучающей выборке данных с целью упорядочивания объектов выборки в однородные группы. В машинном обучении зада кластеризации решается алгоритмами работающих по принципу «Обучению без учителя» Unsupervised learning. В этом случае для каждого прецедента (объекта обучающей выборки) задаётся только ситуация (набор входных параметров). При этом решается задача оптимального распределения объектов по кратерам.

Существует большое количество различных алгоритмов кластеризации, всех их принято делить на следующие группы: неиерархические и иерархические алгоритмы.

Самым известным неиерархическим алгоритмом кластеризации является – k-means (k-средних). Он был разработан в 1954 году Ллойдом С., Штейнгаузом Г.

Кластеризация данных применяется при решении обширного круга задач возникающих при проектировании интеллектуальных систем. Например в системах анализа видео ряда, кластеризация позволяет решать задачи обнаружения и локализации двужущихся объектов на изображениях (рисунок 1.14).

При этом объектами подвергающихся кластеризации выступают пиксели изображения, изменивших свое значение при сравнеии с предудущими кадрами. Данные объекты описываются двумя измерениями – координата по оси X и координата по оси Y.

Для выполнения кластаризации по алгоритму k-means необходимо определить:

- Обучающую выборку, объекты из которой будут использованы при кластеризации. Для всех объектов должны быть известны все значения параметров, пропуски не допускаются.

- Тип метрики, используемой для оценки расстояний. Известно множество разных метрик, для оценки расстояния между объектами. Сюда можно отнести метрики Махаланобиса, Евклида, Манхэттена и др. Какая из метрик сможет обеспечить наилучший результат кластеризации, зависит от данных обучающей выборки.
- Количество кластеров.

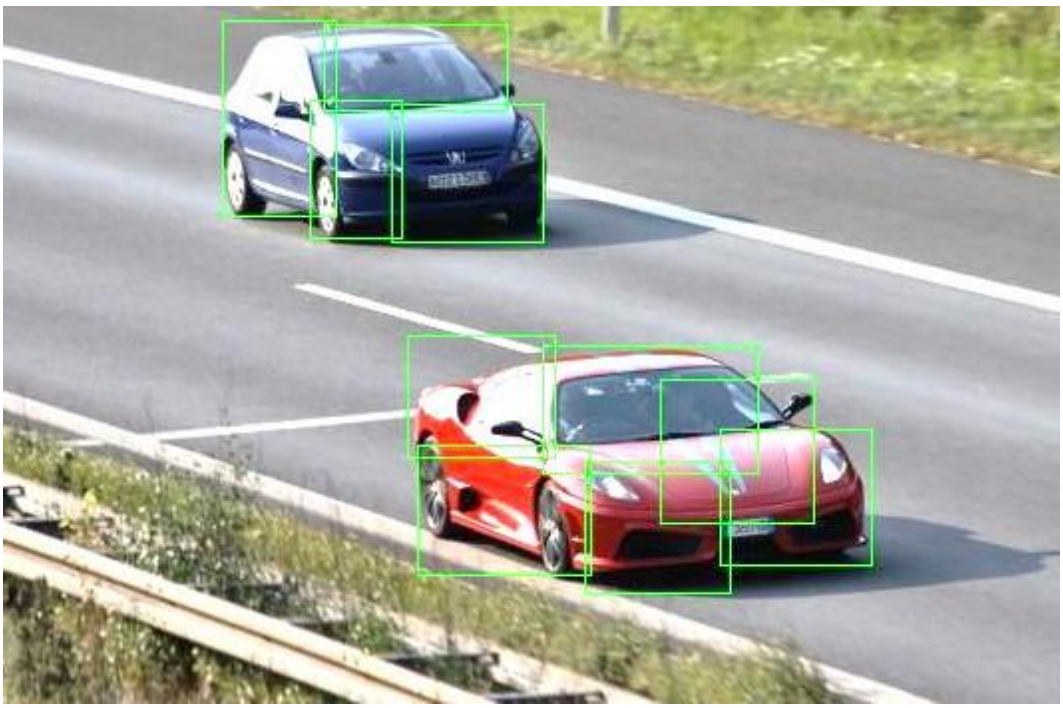


Рисунок 1.14 – Локализация движущихся объектов с применением алгоритма кластеризации количество кластеров k=10

Работа алгоритма k-means заключается в последовательном выполнении следующих этапов:

Этап №1. Из обучающей выборки случайно выбираются несколько элементов, по координатам которых располагают предполагаемые центы кластеров. Выбираемое количество элементов равно количеству кластеров.

Этап №2. Для того, чтобы определить принадлежность обектов обучающей выборки к кластерам производится расчет расстояния от каждого объекта до каждого центра кластера. К какому центру кластера объект ближе к той группе объектов он и относится.

Для оценки расстояний могут применяться различные метрики:

- Метрика Евклида d_E, для оценки расстояния между векторами *x* и *y* вычисляется по 1.1 :

$$d_E(x,y) = \sqrt{\sum_i (x_i - y_i)^2}\,, \qquad 1.1$$

где *i* – размерность векторов.

- Метрика Манхэттена d_M для оценки расстояния между векторами *x* и *y* вычисляется по 1.2 :

$$d_M(x,y) = \sum_i |x_i - y_i| \qquad 1.2$$

где *i* – размерность векторов.

Этап №3. После распределения объектов по кластерам становиться возможным расчёт центра масс кластеров, как среднее значении по каждому из параметров объектов попавших в данный кластер.

С целью минимизации ошибки кластеризации *V* (суммы квадратов ошибок) этапы 2 и 3 повторяются до тех пор пока распределение объектов по кластерам не перестанет меняться.

Ошибка кластеризации рассчитывается следующим образом (1.3):

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \qquad 1.3$$

где $k -$ число кластеров; $i = 1...k$; $j -$ количество объектов; $S_i -$ полученные кластеры; $\mu_i -$ центры масс векторов $x_j \in S_i$.

Результатом выполнения кластеризация с использованием алгоритма k-means разделение обучающей выборки на *k* групп (кластеры).

В процессе поиска оптимальной кластерной структуры на каждой последующей итерации проводится уточнение центров кластеров. Это приводит к тому, что изменяется и принадлежность объектов к имеющимся кластерам.

К становиться понятно, поиск оптимальной кластерной структуры является много итерационным процессом. На каждой итерации уменьшается ошибка кластеризации *V*. На последней итерации принадлежность объектов обучающей выборки к своим кластерам не изменяется и процесс кластеризации останавливается.

У алгоритма k-means существует много различных вариаций полученных комбинированием различных интеллектуальных технологий – нечеткой логики, нейронных сетей и др.

К преимуществам алгоритма k-means отнести следующее:

- возможность применения алгоритма к описываемых лубым сочетанием числовых и категориальных признаков;

- простой математический аппарат алгоритма.

Одной из проблем алгоритма является вероятность схождения процесса поиска оптимальной кластерной структуры к неоптимальному решению.

Для демонстрации этой проблемы была сгенерирована обучающая выборка, состоящая из 77 элементов. При этом каждый объект описывается двумя числовыми признаками – X и Y

Таблица 1.1 – Обучающая выборка

N	*X*	*Y*	*N*	*X*	*Y*	*N*	*X*	*Y*	*N*	*X*	*Y*
1	0,124	0,939	21	0,113	0,392	41	0,497	0,229	61	0,872	0,910
2	0,144	0,932	22	0,171	0,392	42	0,489	0,188	62	0,960	0,869
3	0,028	0,893	23	0,180	0,360	43	0,479	0,186	63	0,879	0,863
4	0,040	0,884	24	0,208	0,349	44	0,419	0,195	64	0,829	0,889
5	0,055	0,893	25	0,234	0,325	45	0,473	0,163	65	0,838	0,869
6	0,075	0,887	26	0,197	0,300	46	0,445	0,130	66	0,871	0,839
7	0,042	0,828	27	0,197	0,286	47	0,428	0,073	67	0,894	0,839
8	0,098	0,842	28	0,260	0,267	48	0,447	0,064	68	0,889	0,835
9	0,170	0,877	29	0,234	0,249	49	0,516	0,046	69	0,929	0,799
10	0,223	0,844	30	0,238	0,235	50	0,605	0,072	70	0,902	0,783
11	0,189	0,812	31	0,197	0,228	51	0,592	0,100	71	0,859	0,766
12	0,060	0,765	32	0,180	0,192	52	0,572	0,093	72	0,924	0,754
13	0,182	0,765	33	0,136	0,283	53	0,559	0,096	73	0,952	0,745
14	0,094	0,754	34	0,113	0,360	54	0,536	0,144	74	0,891	0,743
15	0,156	0,735	35	0,088	0,374	55	0,562	0,171	75	0,902	0,723
16	0,167	0,717	36	0,076	0,383	56	0,616	0,174	76	0,853	0,747
17	0,061	0,690	37	0,074	0,367	57	0,585	0,148	77	0,882	0,705
18	0,087	0,696	38	0,068	0,354	58	0,582	0,141			
19	0,138	0,679	39	0,514	0,283	59	0,831	0,955			
20	0,227	0,431	40	0,525	0,262	60	0,924	0,941			

Обучающая выборка из таблицы 1.1 может быть графически представлена на декартовой системе координат (рисунок 1.15).

Очевидно, что оптимальной кластерной структурой в этом для этого случая является распределение объектов, представленное на рисунке ниже (рисунок 1.16) .

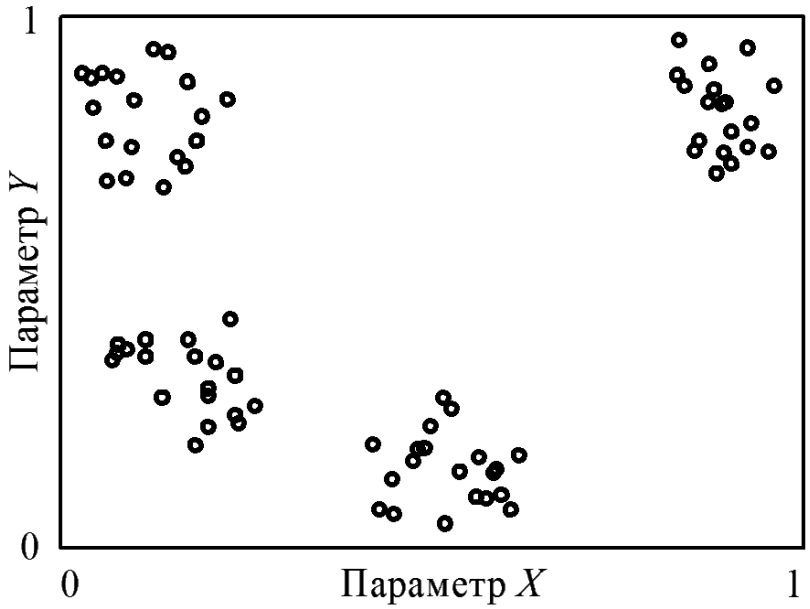


Рисунок 1.15 – Графическое представление обучающей выборки

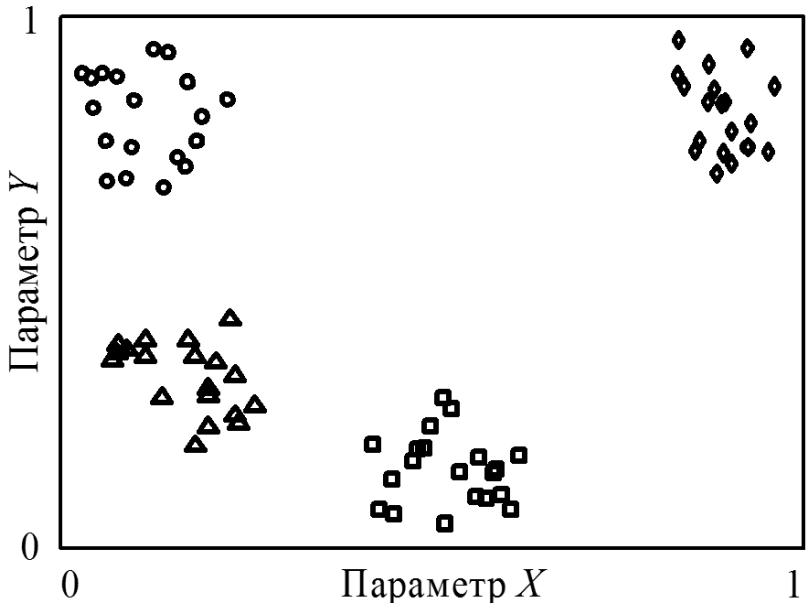


Рисунок 1.16 – Оптимальное распределение объектов по кластерам: фигурами различной формы обозначена принадлежность объектов к кластерам

Проведем кластеризацию данных из таблицы 1.1 с использованием алгоритма k-means и проверим, удастся ли автоматически получить кластерную структуру представленную на рисунке 1.16. Выбираем

случайным образом начальное расположение центров кластеров и начинаем вычисления (рисунок 1.17).

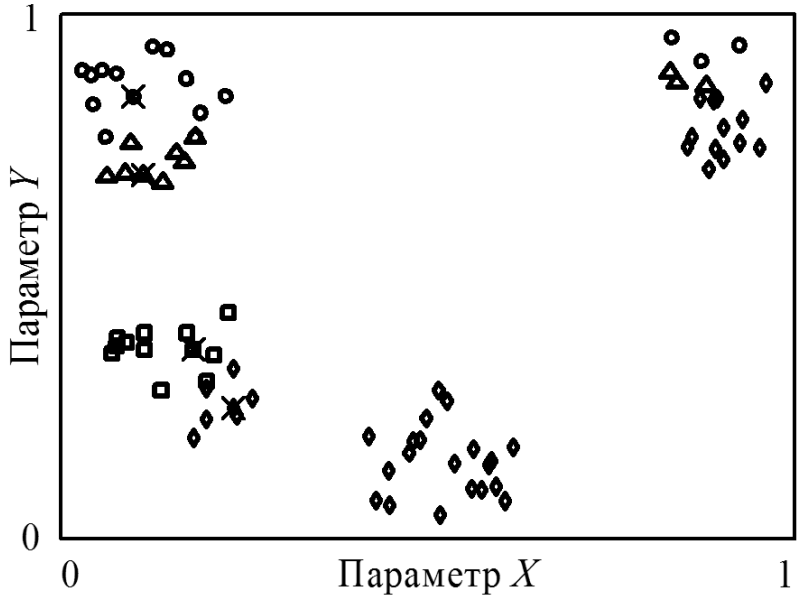


Рисунок 1.17 – Кластерная структура на итерации №1, “x” – центры кластеров на данной итерации

На рисунке 1.18 представлена кластерная структура на второй итерации выполнения алгоритма k-means.

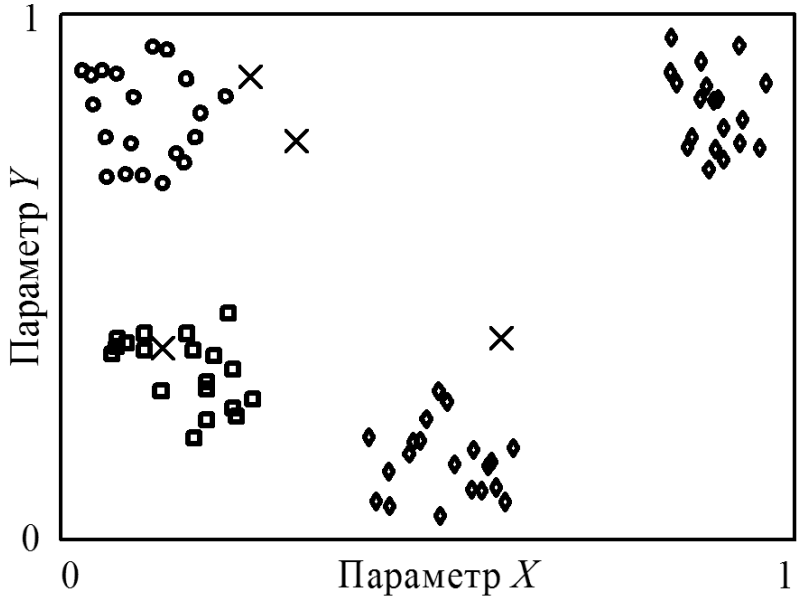


Рисунок 1.18 – Кластерная структура на итерации №2: “x” – центры кластеров на данной итерации

На рисунке 1.19 представлена кластерная структура на третьей итерации выполнения алгоритма k-means.

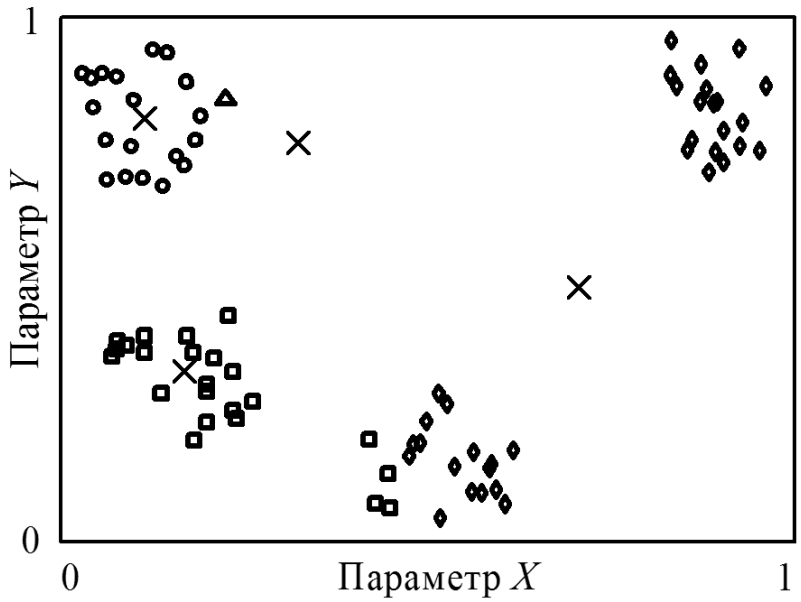


Рисунок 1.19 – Кластерная структура на итерации №3: “x” – центры кластеров на данной итерации

На рисунке 1.20 представлена кластерная структура на четвертой итерации выполнения алгоритма k-means.

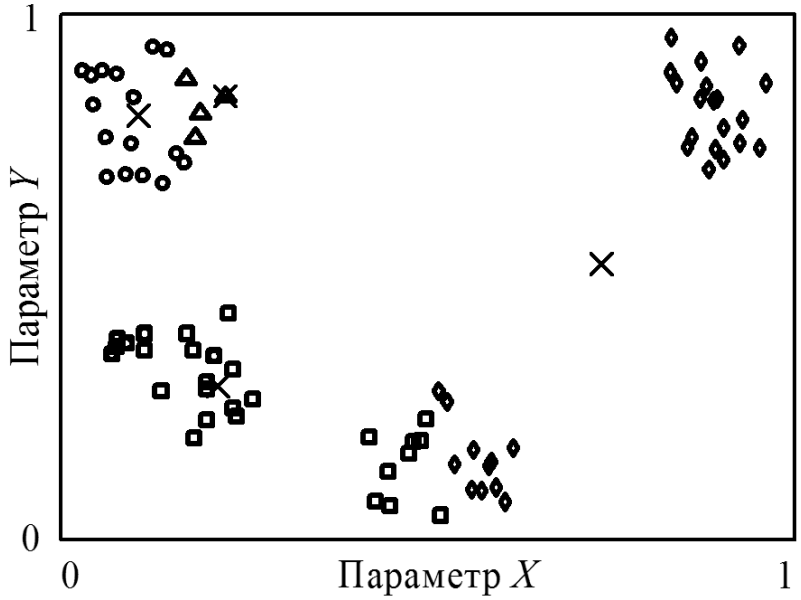


Рисунок 1.20 – Кластерная структура на итерации №4: “x” – центры кластеров на данной итерации

На рисунке 1.21 представлена кластерная структура на пятой итерации выполнения алгоритма k-means.

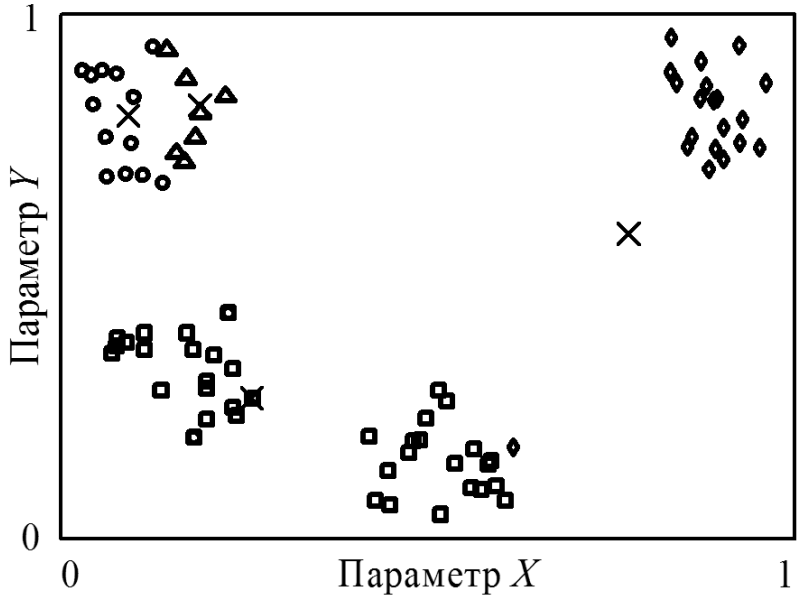


Рисунок 1.21 – Кластерная структура на итерации №5: “x” – центры кластеров на данной итерации

На рисунке 1.22 представлена кластерная структура на последней итерации выполнения алгоритма k-means.

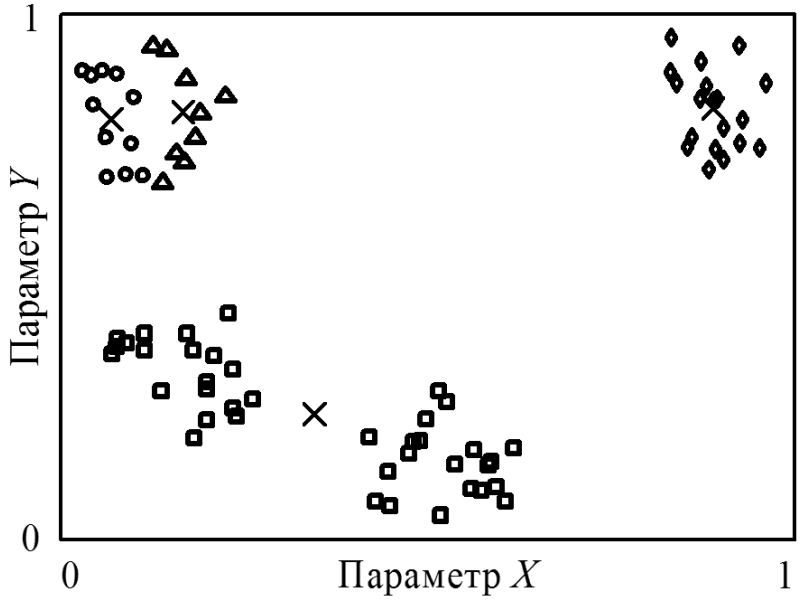


Рисунок 1.22 – Окончательная кластерная структура: “x” – центры кластеров на данной итерации

Путем сравнения рисунков 1.16 и 1.22 можно убедиться, что в данном случае найденная кластерная структура не является оптимальной. Если запустить алгоритм k-means еще несколько раз на тех же самых исходных данных, то можно добиться распределения объектов по кластерам, как это представлено на рисунке 1.22.

На основе представленных результатов можно сделать вывод, что результаты кластеризации зависят от выбора начального расположения центров кластеров. Так как в классическом варианте алгоритма k-means их расположение выбирается случайным образом, то для снижения вероятности нахождения неоптимального решения необходимо разработать методику для определения начальных центров кластеров.

Таким образом, на основе анализа литературных источнков, включая отечественные и зарубежные научные пкбликации можно сделать следующие выводы.

В машинном обучении существует алгоритм, предназначенный для кластеризации данных – k-means. Он автоматически анализирует исходные данные и подбирает оптимальную кластерную структуру. Одной из главных его проблем является возможность попадания алгоритма в локальное решение (вместо глобального). Это связано с тем, что начальное расположение кластеров (которое, впоследствии выполнения алгоритма, уточняется) выбирается случайным образом.

Сформулируем гипотезу исследования.

Гипотезой исследования является предположение, что степень автоматизации, точность и скорость работы алгоритма k-means можно повысить путем разработки метода по расчету начального расположению центров кластеров.

Таким образом актуальной можно признать цель данного исследования.

Цель исследования – повышение степени автоматизации алгоритма k-means путем разработки метода по расчету начального расположению центров кластеров.

Цель достигается путем решения следующих задач:

1. Проведение анализа состояния вопроса по теме исследования

2. Разработка метода по определению начальных положений центров кластеров (при анализе данных с помощью алгоритма k-means).

3. Разработка программной реализация данного метода для проверки его эффективности на практике.

4. Тестирование предложенного метода на практике. Формулирование выводов по результатам вычислительных экспериментов.

2 Разработка метода выбора начального расположения центров кластеров

2.1 Формальное описание задачи кластеризации

Пусть X – множество объектов, Y – множество номеров (имён, меток) кластеров. Задана функция расстояния между объектами $\rho(x, x')$. Имеется конечная обучающая выборка объектов:

$$X^m = \{x_1, ..., x_m\} \subset X \tag{2.1}$$

Требуется разбить выборку на непересекающиеся подмножества, называемые кластерами, так, чтобы каждый кластер состоял из объектов, близких по метрике ρ, а объекты разных кластеров существенно отличались. При этом каждому объекту $x_i \in X^m$ приписывается номер кластера y_i.

Алгоритм кластеризации — это функция $\alpha : X \to Y$, которая любому объекту $x \in X$ ставит в соответствие номер кластера $y \in Y$. Множество Y в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров, с точки зрения того или иного критерия качества кластеризации.

Кластеризация (обучение без учителя) отличается от классификации (обучения с учителем) тем, что метки исходных объектов y_i изначально не заданы, и даже может быть неизвестно само множество Y.

Решение задачи кластеризации принципиально неоднозначно, и тому есть несколько причин:

- Не существует однозначно наилучшего критерия качества кластеризации. Известен целый ряд эвристических критериев, а также ряд алгоритмов, не имеющих чётко выраженного критерия, но осуществляющих достаточно разумную кластеризацию «по построению». Все они могут давать разные результаты.
- Число кластеров, как правило, неизвестно заранее и устанавливается в соответствии с некоторым субъективным критерием.

- Результат кластеризации существенно зависит от метрики, выбор которой, как правило, также субъективен и определяется экспертом.

Типы входных данных

Признаковое описание объектов. Каждый объект описывается набором своих характеристик, называемых признаками. Признаки могут быть числовыми или нечисловыми.

Матрица расстояний между объектами. Каждый объект описывается расстояниями до всех остальных объектов обучающей выборки.

Матрица расстояний может быть вычислена по матрице признаковых описаний объектов бесконечным числом способов, в зависимости от того, как ввести функцию расстояния (метрику) между признаковыми описаниями. Часто используется евклидова метрика, однако этот выбор в большинстве случаев является эвристикой и обусловлен лишь соображениями удобства.

Обратная задача — восстановление признаковых описаний по матрице попарных расстояний между объектами — в общем случае не имеет решения, а приближённое решение не единственно и может иметь существенную погрешность. Эта задача решается методами многомерного шкалирования.

Таким образом, постановка задачи кластеризации по матрице расстояний является более общей. С другой стороны, при наличии признаковых описаний часто удаётся строить более эффективные методы кластеризации.

Цели кластеризации:

1. Понимание данных путём выявления кластерной структуры. Разбиение выборки на группы схожих объектов позволяет упростить дальнейшую обработку данных и принятия решений, применяя к каждому кластеру свой метод анализа (стратегия «разделяй и властвуй»).

2. Сжатие данных. Если исходная выборка избыточно большая, то можно сократить её, оставив по одному наиболее типичному представителю от каждого кластера.

3. Обнаружение новизны (novelty detection). Выделяются нетипичные объекты, которые не удаётся присоединить ни к одному из кластеров.

В первом случае число кластеров стараются сделать поменьше. Во втором случае важнее обеспечить высокую (или фиксированную) степень сходства объектов внутри каждого кластера, а кластеров может быть сколько угодно. В третьем случае наибольший интерес представляют отдельные объекты, не вписывающиеся ни в один из кластеров.

Во всех этих случаях может применяться иерархическая кластеризация, когда крупные кластеры дробятся на более мелкие, те в свою очередь дробятся ещё мельче, и т. д. Такие задачи называются задачами таксономии.

Результатом таксономии является древообразная иерархическая структура. При этом каждый объект характеризуется перечислением всех кластеров, которым он принадлежит, обычно от крупного к мелкому. Визуально таксономия представляется в виде графика, называемого дендрограммой.

2.2 Математический аппарат алгоритма EM – кластеризации

EM-алгоритм (англ. expectation-maximization) - алгоритм, используемый в математической статистике для нахождения оценок максимального правдоподобия параметров вероятностных моделей, в случае, когда модель зависит от некоторых скрытых переменных. Каждая итерация алгоритма состоит из двух шагов. На E-шаге (expectation) вычисляется ожидаемое значение функции правдоподобия, при этом скрытые переменные рассматриваются как наблюдаемые. На M-шаге (maximization) вычисляется оценка максимального правдоподобия, таким образом увеличивается ожидаемое правдоподобие, вычисляемое на E-шаге. Затем это значение используется для E-шага на следующей итерации. Алгоритм выполняется до сходимости.

Как правило, EM-алгоритм применяется для решения задач двух типов.

К первому типу можно отнести задачи, связанные с анализом действительно неполных данных, когда некоторые статистические данные отсутствуют в силу каких-либо причин.

Ко второму типу задач можно отнести те задачи, в которых функция правдоподобия имеет вид, не допускающий удобных аналитических методов исследования, но допускающий серьезные упрощения, если в задачу ввести дополнительные «ненаблюдаемые» (скрытые, латентные) переменные. Примерами прикладных задач второго типа являются задачи распознавания образов, реконструкции изображений. Математическую суть данных задач составляют задачи кластерного анализа, классификации и разделения смесей вероятностных распределений.

Постановка задачи

Пусть плотность распределения на множестве X имеет вид смеси k распределений:

$$p(x) = \sum_{j=1}^{k} w_j p_j(x), \sum_{i=1}^{k} w_j = 1, w_j \geq 0 \qquad (2.2)$$

где $p_j(x)$ — функция правдоподобия j-й компоненты смеси, w_j – ее априорная вероятность.

Пусть функции правдоподобия принадлежат параметрическому семейству распределений $\varphi(x;\theta)$ и отличаются только значениями параметра $P_j(x) = \varphi(x;\theta_j)$.

Задача разделения смеси заключается в том, чтобы, имея выборку X^m случайных и независимых наблюдений из смеси $p(x)$, зная число k и функцию φ, оценить вектор параметров

$$\Theta = (w_1,...,w_k,\theta_1,...,\theta_k) \qquad (2.3)$$

Опишем основной алгоритм.

Идея алгоритма заключается в следующем. Искусственно вводится вспомогательный вектор скрытых переменных G, обладающий двумя замечательными свойствами.

- С одной стороны, он может быть вычислен, если известны значения вектора параметров Θ.
- С другой стороны, поиск максимума правдоподобия сильно упрощается, если известны значения скрытых переменных.

ЕМ-алгоритм состоит из итерационного повторения двух шагов.

На Е-шаге вычисляется ожидаемое значение (expectation) вектора скрытых переменных G по текущему приближению вектора параметров Θ.

Обозначим через $p(x,\theta_j)$ плотность вероятности того, что объект x получен из j-й компоненты смеси. По формуле условной вероятности

$$p(x,\theta_j) = p(x)P(\theta_j \mid x) = w_j p_j(x) \tag{2.4}$$

Введём обозначение

$$g_{ij} = P(\theta_j \mid x_i). \tag{2.5}$$

Это неизвестная апостериорная вероятность того, что обучающий объект x_i получен из j-й компоненты смеси. Возьмём эти величины в качестве скрытых переменных.

$$\sum_{j=1}^{k} g_{ij} = 1, \tag{2.6}$$

для любого $i = 1...m$ так как имеет смысл полной вероятности принадлежать объекту x_i одной из k компонент смеси. Из формулы Байеса

$$g_{ij} = \frac{w_j p_j(x_i)}{\sum_{t=1}^{k} w_t p_t(x_i)} \text{ для всех } i, j \tag{2.7}$$

На М-шаге решается задача максимизации правдоподобия (maximization) и находится следующее приближение вектора Θ по текущим значениям векторов G и Θ.

Будем максимизировать логарифм полного правдоподобия:

$$Q(\Theta) = \ln \prod_{i=1}^{m} p(x_i) = \sum_{i=1}^{m} \ln \sum_{j=1}^{k} w_j p_j(x_i) \to \max_{\Theta} \tag{2.8}$$

Решая оптимизационную задачу Лагранжа с ограничением на сумму w_j, находим:

$$w_j = \frac{1}{m} \sum_{i=1}^{m} g_{ij},\ j = 1,...,k. \tag{2.9}$$

$$\theta_j = \arg\max_{\theta} \sum_{i=1}^{m} g_{ij} \ln \varphi(x_i, \theta),\ j = 1,...,k \tag{2.10}$$

Таким образом, М-шаг сводится к вычислению весов компонент w_j как средних арифметических и оцениванию параметров компонент θ_j путём решения k независимых оптимизационных задач. Отметим, что разделение переменных оказалось возможным благодаря удачному введению скрытых переменных.

Таким образом, EM – алгоритм состоит из следующих шагов:

1. Начальное приближение w_j, μ_j для всех у.

2. Е-шаг (expectation):

$$g_{ij} = \frac{w_j N(x_i; \mu_j, \sigma_j)}{\sum_{t=1}^{k} w_t N(x_i; \mu_t, \sigma_t)} \tag{2.11}$$

3. М-шаг (maximization):

$$w_j = \frac{1}{m} \sum_{i=1}^{m} g_{ij} \tag{2.12}$$

$$\mu_j = \frac{1}{mw_j} \sum_{i=1}^{m} g_{ij} x_i \tag{2.13}$$

$$\sigma_j^2 = \frac{1}{mw_j} \sum_{i=1}^{m} g_{ij} (x_i - \mu_j)^2, \; j = 1,...,k \tag{2.14}$$

4.Расчет y_i :

$$y_i = \arg\max_{y \in Y} g_{ij}, i = 1,...,m \tag{2.15}$$

5. Повторять шаги 2 и 3, пока y_i не перестанут изменяться.

Недостатки основного алгоритма:

- основной алгоритм неустойчив по начальным данным (то есть тем, которые инициализируют вектор параметров \theta на первой итерации), так как он находит локальный экстремум, значение которого может оказаться гораздо ниже, чем глобальный максимум. В зависимости от выбора начального приближения алгоритм может сходиться к разным точкам. Также может сильно варьироваться скорость сходимости.
- основной алгоритм не позволяет определять количество k компонент смеси. Эта величина является структурным параметром алгоритма.

Алгоритм k-means является упрощенным аналогом EM-алгоритма, который состоит из следующих шагов:

1. Начальное приближение центров $\mu_y, y \in Y$

2. E-шаг (expectation):

Отнести каждый x_i к ближайшему центру

$$y_i = \arg\max_{y \in Y} \rho(x_i, \mu_y), i = 1,...,l \tag{2.16}$$

3. M-шаг (maximization):

Вычислить новые положения центров:

$$\mu_{yj} = \frac{\sum_{i=1}^{l}[y_i = y]f_j(x_i)}{\sum_{i=1}^{l}[y_i = y]}, y \in Y,\ j = 1,...,n \qquad (2.17)$$

5. Повторять шаги 2 и 3, пока y_i не перестанут изменяться.

Основные отличия EM и k-means:

- Жесткость кластеризации:

  EM – мягкая кластеризация $g_{ij} = P\{y_i = y\}$

  k-means – жесткая кластеризация $g_{ij} = [y_i = y]$

- Форма растеров:

  EM – эллиптическая

  k-means – определяется метрикой

2.3 Методика определения первоначального расположения кластеров

Как было отмечено в первой главе, точность работы алгоритма k-means, а также скорость сходимости к решению кластеризации зависит от выбора первоначального положения кластеров. В связи с этим целесообразно ориентировочное определение расположение центров кластеров до начала кластеризации.

Сам алгоритм определения первоначального расположения центров кластеров должен обладать следующими особенностями:

1. простым математическим аппаратом (по сравнению с самим алгоритмом кластеризации);
2. при этом не требуется высокая точность предсказания конечного положения центров кластера, т.к. это берет на себя алгоритм k-means.

Из рассмотренного в первой главе примера (рисунок 2.1) понятно, что при распределении первоначальных центров кластеров по локальным скоплениям объектов (не допуская при этом расположения двух центров

кластеров в одном скоплении) в результате кластеризации, с высокой вероятностью будет получаться наиболее оптимальная кластерная структура.

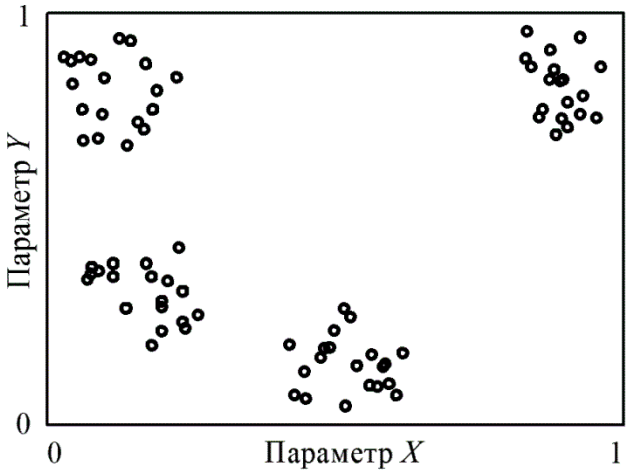


Рисунок 2.1 – Графическое представление обучающей выборки

Для нахождения скоплений объектов был предложен следующий алгоритм:

1. Каждое из измерений исследуемого пространства разбивается на две равные части, таким образом, мы получаем определенное количество R областей. Будем условно считать, что это первая степень (“глубина” D=1) разбиения исходной области на подобласти. При глубине анализа данных равной двум (D=2), каждая получившаяся подобласть аналогичным образом разбивается на дополнительные подобласти и т.д., пока не будет достигнута нужная глубина D. Расчетная формула итогового количества R подобластей следующая:

$$R = (2 \cdot D)^N, \tag{2.18}$$

где R – количество подобластей, D – глубина разбиения, N – количество атрибутов объекта.

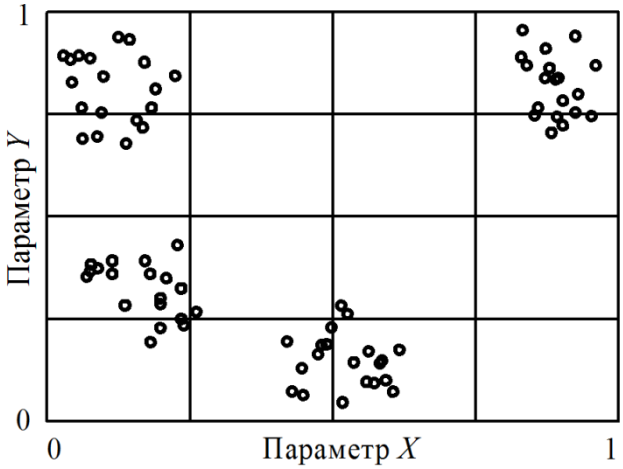
2. Затем вводится подсчет количества объектов в каждой области.
3. Первоначальное расположение центров кластеров берется как геометрические центры областей с наибольшим содержанием объектов. При прочих равных условиях предпочтение отдается областям с максимальным геометрическим удалением друг от друга

(однако, как выяснилось на практике – соблюдение этого правила дает незначительный прирост к точности кластеризации).

Выбор глубины D анализа исходных данных зависит от таких характеристик, как: количество кластеров k, количество объектов n, количество атрибутов N. Рекомендации по выбору D в зависимости от этих параметров приведены ниже.

Для ясности рассмотрим данный алгоритм для данных из таблицы 1.1. Графически эти данные представлены на рисунке 2.1. Исходные данные обладают следующими характеристиками: количество объектов n=77, количество кластеров k=4, количество атрибутов N=2. Для такого сочетания параметров рекомендуемое значение глубина разбиения D=2.

В соответствии со значением D разобьём пространство на подобласти и найдем количество объектов в каждой из них (рисунок 2.2):



14	0	0	**14**
5	0	0	5
14	1	2	0
4	8	**10**	0

Рисунок 2.2 – Определение областей двухмерного пространства с максимальным содержанием элементов обучающей выборки

Первоначальное расположение центров кластеров помещаем в центры областей наибольшим содержанием объектов (рисунок 2.3).

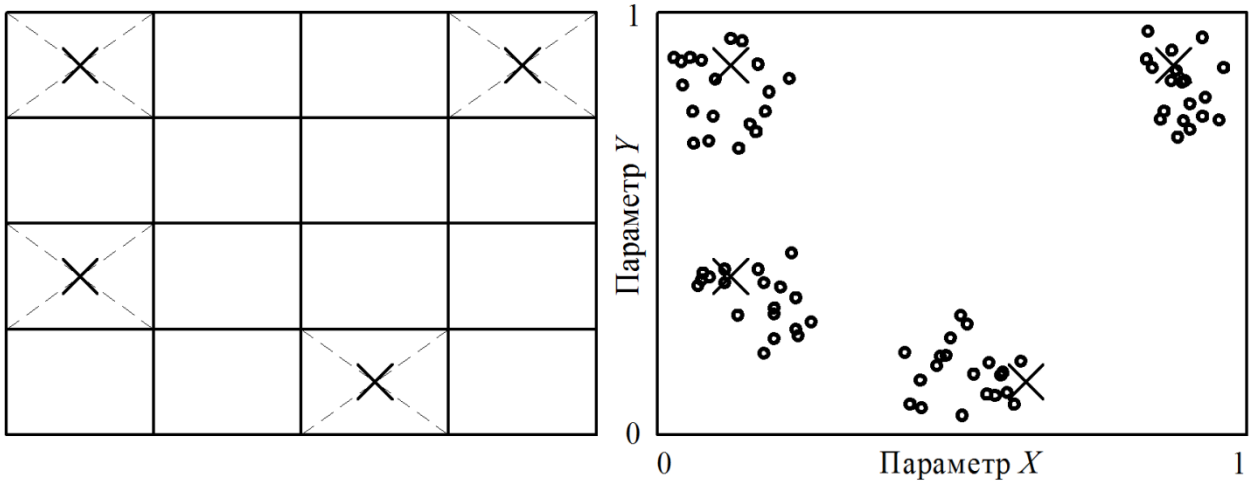


Рисунок 2.3 – Результат определения начального расположения цента кластеров

Запустив алгоритм кластеризации k-means с выбранным первоначальным расположением центров кластеров, убеждаемся в том, что найдено решение с оптимальной кластерной структурой.

2.4 Проведение вычислительных экспериментов

Для того, чтобы оценить эффективность предложенного подхода по заданию начального нахождения центров кластеров были проведены вычислительные эксперименты.

Вычислительные эксперименты проводились так:

1) Осуществлялось изменение значения размера обучающей выборки в диапазоне от 10 до 500. При этом количество измерений у векторов входных данных изменялось в диапазоне от 2 до 80. Количество требуемых групп объектов в кластерной структуре изменялось в диапазоне от 2 до 50.

2) Для каждой совокупности данных характеристик (описанных в предыдущем подпункте) осуществлялась генерация обучающей выборки.

При этом значения параметров, описывающих объектов задавалось случайным образом в диапазоне от 0 до 1.

3) Затем для каждого набора данных многократно запускался процесс кластеризации: 100 раз осуществлялась кластеризация со случайным выбором первоначальных положений кластеров, 10 раз осуществлялась кластеризация с использованием предложенной методики определения начального положения кластеров (при этом каждый раз задавалось разное значение параметра *D*).

4) При проведении экспериментов по кластеризации данных производился расчет и сравнение следующих характеристик:

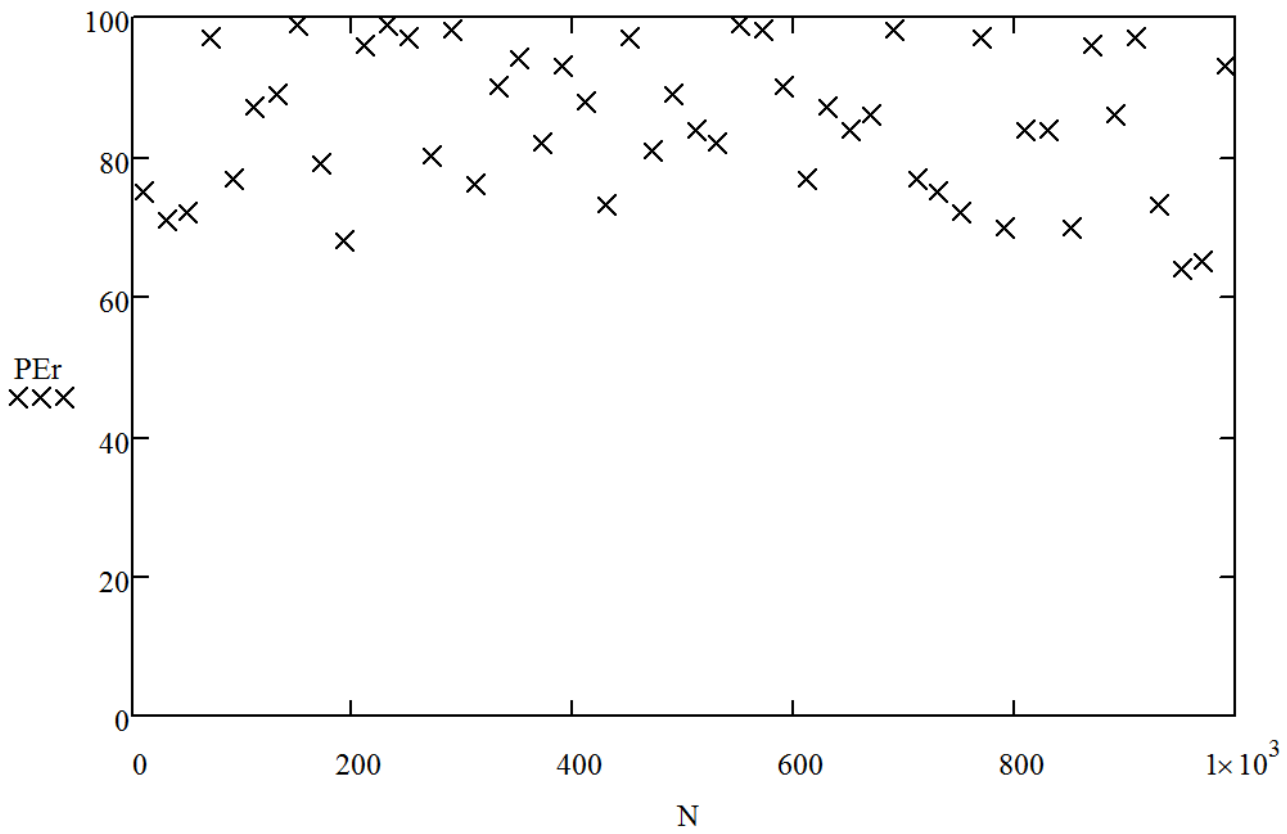
- Вероятность нахождения оптимальной кластерной структуры (с наименьшей суммой квадратов ошибок), которая рассчитывается на основе 100 экспериментов..
- Минимальная ошибка кластеризации, полученная для данного набора данных за 100 экспериментов в режиме случайного выбора первоначального расположения кластеров.
- Максимальная ошибка кластеризации, полученная для данного набора данных за 100 экспериментов в режиме случайного выбора первоначального расположения кластеров.
- Минимальное количество итераций потребовавшихся для нахождения решения, при данном наборе данных за 100 экспериментов в режиме случайного выбора первоначального расположения кластеров.
- Максимальное количество итераций потребовавшихся для нахождения решения, при данном наборе данных за 100 экспериментов в режиме случайного выбора первоначального расположения кластеров.
- Сумма квадратов ошибок при кластеризации исходного набора данных в режиме выбора первоначального расположения центров кластеров по предложенному алгоритму.

- Глубина анализа данных при расчете первоначально центра кластеров, обеспечивающая минимальную ошибку кластеризации для данного набора данных.
- Количество итераций потребовавшихся для нахождения решения при кластеризации исходного набора данных в режиме выбора первоначального расположения центров кластеров с использованием разработанного метода.

2.5 Результаты вычислительных экспериментов

Для демонстрации результатов вычислительных экспериментов применялся математический пакет MathCad 14.

При малом количестве кластеров ($k = 2..5$), при кластеризации в режиме случайного выбора первоначального расположения центроидов (как это предусмотрено классическим алгоритмом k-means) вероятность нахождения решения с минимальной возможной ошибкой колеблется в пределах от 3-30% (рисунок 2.4). Т.е. за 100 запусков алгоритма на одном и том же наборе данных лишь в 3-30 случаях была найдена оптимальная кластерная структура.

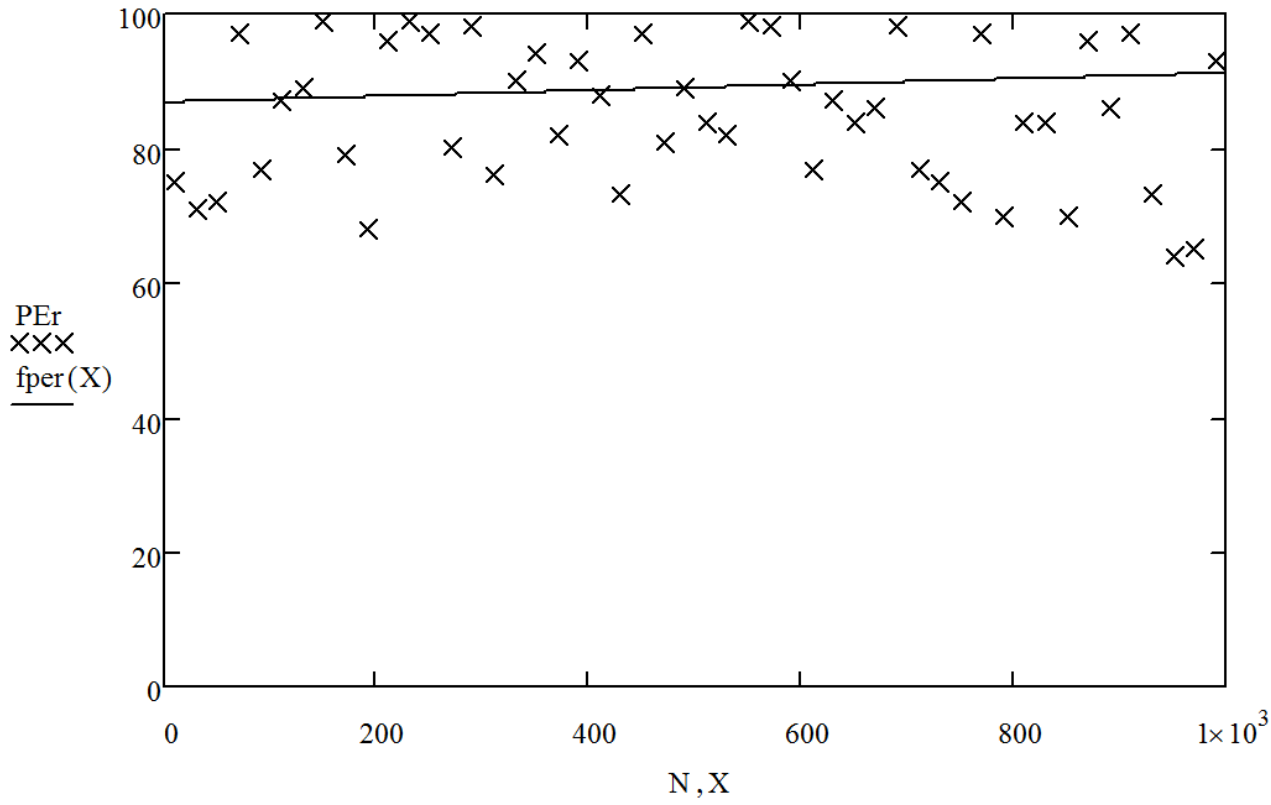


Per – вероятность нахождения неоптимальной кластерной структуры

N – количество элементов обучающей выборки

Рисунок 2.4 – Результаты вычислительных экспериментов каждого для каждого значения N (при $k = 3$ и $n = 2$)

При этом найденные максимальные и минимальные значения ошибок увеличиваются пропорционально росту количества записей (рисунок 2.5).



Per – вероятность нахождения неоптимальной кластерной структуры

fper(*x*) – линейная функция аппроксимации полученных данных

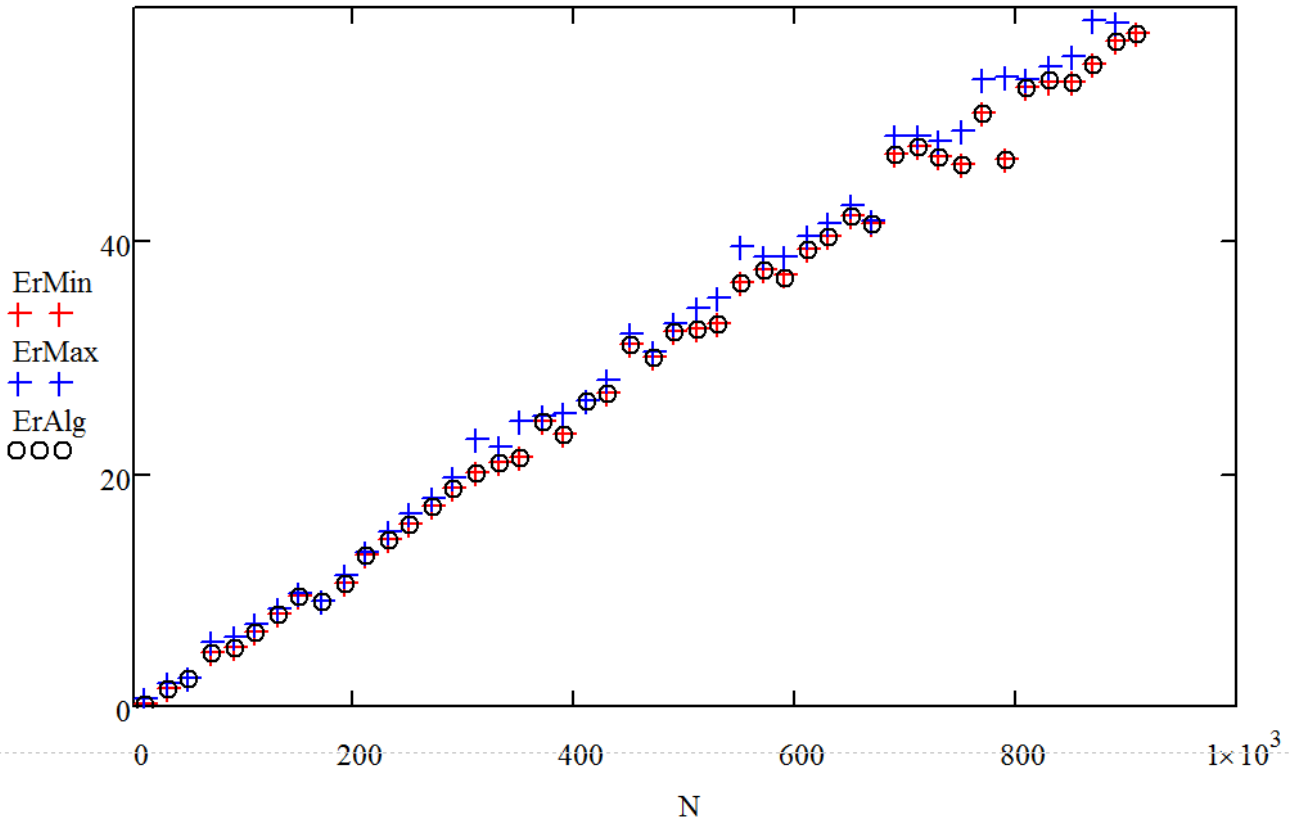
Рисунок 2.5 – Аппроксимация полученных данных (при $k = 3$ и $n = 2$)

Полученные экспериментальным путем данные были аппроксимированы линейной функцией *fper*(*x*) (рисунок 2.5). Для нахождения данной функции использовался метод наименьших квадратов:

$$fper(x) = (4{,}334\times10^{-3})\cdot x + 86{,}867 \qquad (2.19)$$

Стандартное отклонение при аппроксимации данных представленной функцией составляет:

$$S_{\text{per}} = \sqrt{\frac{1}{n-2}\cdot\left[\sum_i (fper(N_i) - PEr_i)^2\right]} = 11{,}769 \qquad (2.20)$$



ErMin – минимальная ошибка кластеризации;

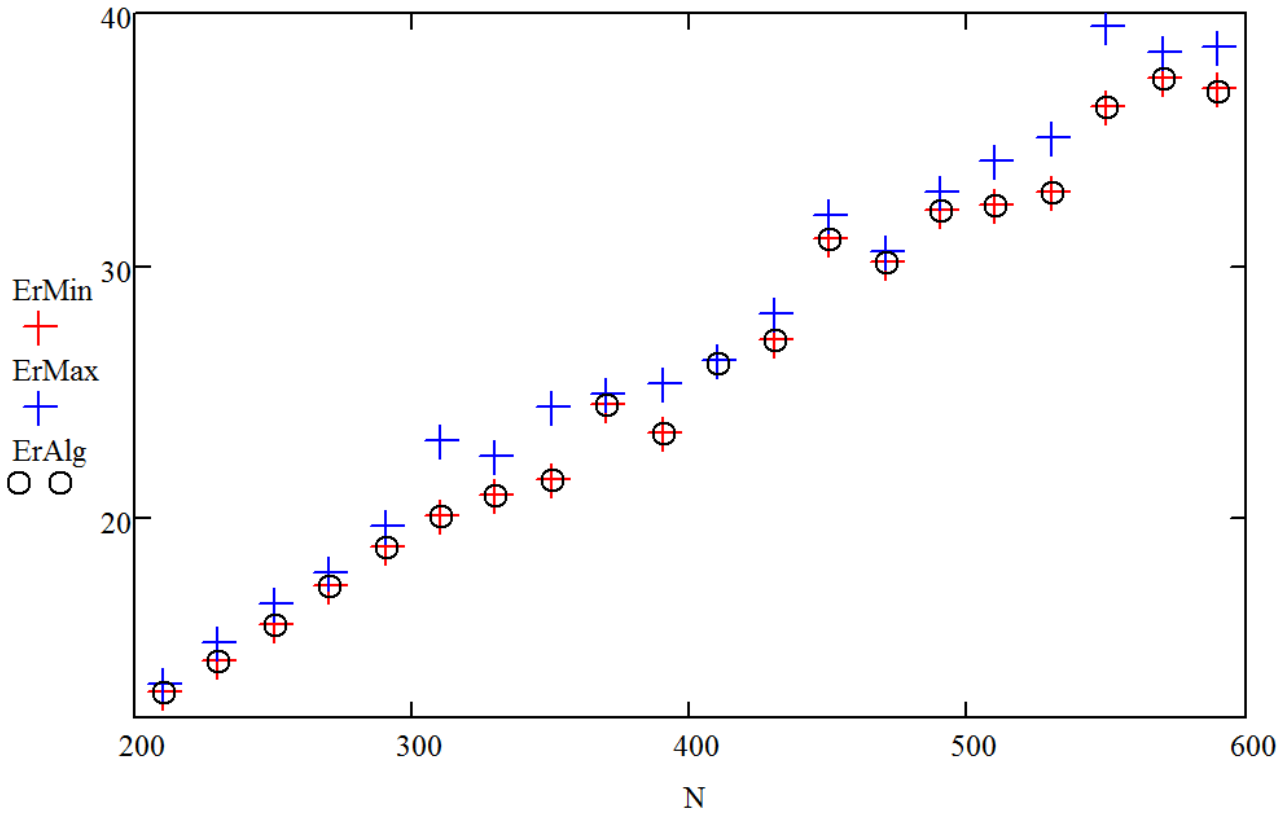
ErMax – максимальная ошибка кластеризации;

ErAlg – ошибка кластеризации при использовании предложенного алгоритма; .

N – количество элементов обучающей выборки.

Рисунок 2.6 – Результаты вычислительных экспериментов каждого для каждого значения N (при $k = 3$ и $n = 2$)

При увеличении графика, представленного на рисунке 2.6, можно заметить, что кривая *ErAlg* ошибки кластеризации данных при использовании алгоритма определения центров кластеров совпадает с кривой *ErMin* минимальной ошибки найденной за 100 запусков классического алгоритма k-means (рисунок 2.7). Это означает, что практически во всех случаях, благодаря предложенному алгоритму за один запуск была найдена оптимальная кластерная структура. Оптимальная кластерная структура – это та, у которой ошибка минимальна.



ErMin – минимальная ошибка кластеризации;

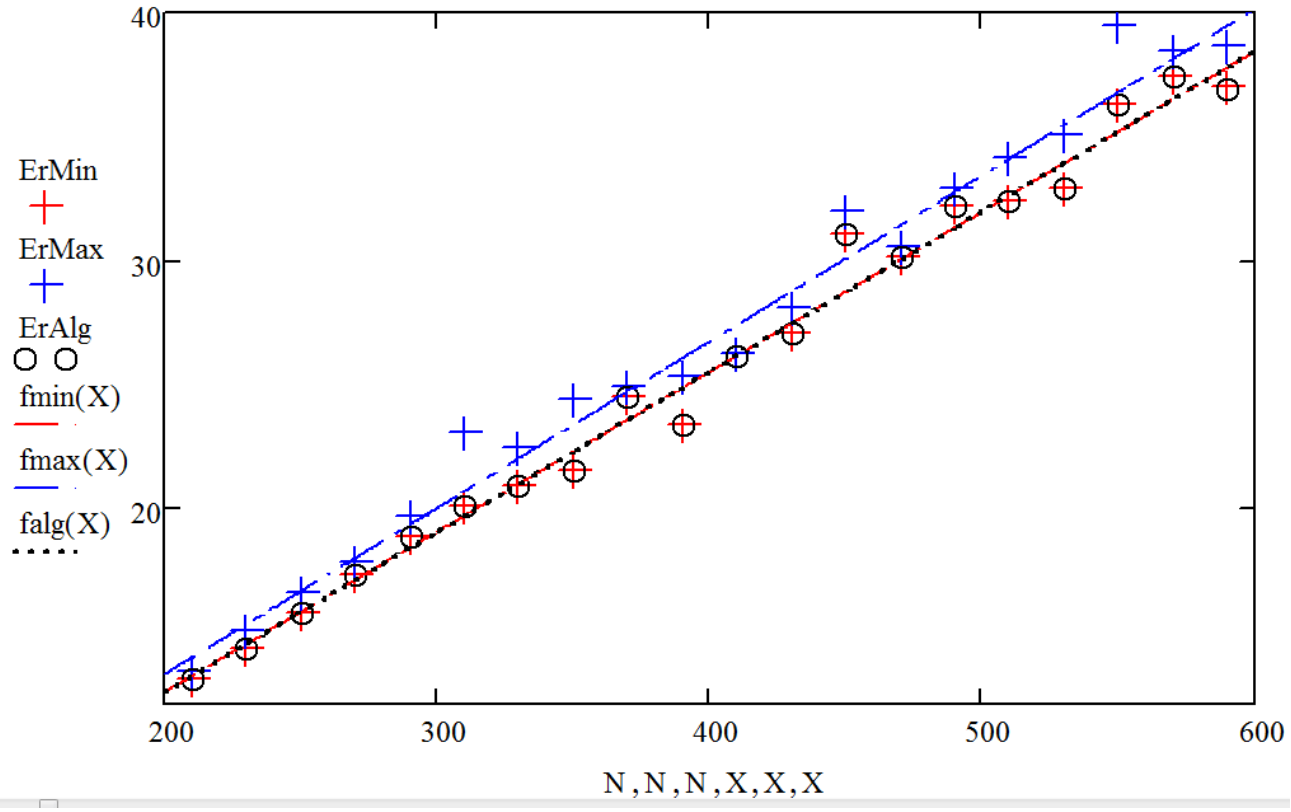
ErMax – максимальная ошибка кластеризации;

ErAlg – ошибка кластеризации при использовании предложенного алгоритма;

N – количество элементов обучающей выборки.

Рисунок 2.7 – Результаты вычислительных экспериментов каждого для каждого значения N (при $k = 3$ и $n = 2$)

Полученные экспериментальным путем данные были аппроксимированы линейными функциями *fmin*(*x*), *fmax*(*x*), *falg*(*x*) (рисунок 2.8).



ErMin – минимальная ошибка кластеризации и ее функция *fmin*(*x*);

ErMax – максимальная ошибка кластеризации и ее функция *fmax*(*x*);

ErAlg – ошибка кластеризации при использовании предложенного алгоритма и ее функция функция *falg*(*x*); *N* – количество элементов обучающей выборки.

Рисунок 2.8 – Аппроксимация полученных данных (при $k = 3$ и $n = 2$)

Для нахождения функций использовался метод наименьших квадратов:

$$fmin(x) = 0{,}065 \cdot x - 0{,}488 \qquad (2.21)$$

$$fmax(x) = 0{,}067 \cdot x - 0{,}27 \qquad (2.22)$$

$$fmin(x) = 0{,}065 \cdot x - 0{,}495 \qquad (2.23)$$

При этом значения стандартных отклонений составили:

$$S_{\min} = \sqrt{\frac{1}{n-2} \cdot \left[\sum_i (fmin(N_i) - ErMin_i)^2 \right]} = 1{,}167 \qquad (2.24)$$

$$S_{\max} = \sqrt{\frac{1}{n-2} \cdot \left[\sum_i (fmax(N_i) - ErMax_i)^2 \right]} = 1,223 \qquad (2.25)$$

$$S_{\text{alg}} = \sqrt{\frac{1}{n-2} \cdot \left[\sum_i (falg(N_i) - ErAlg_i)^2 \right]} = 1,166 \qquad (2.26)$$

Результаты исследования значений глубины *D*, обеспечивающих минимальную ошибку кластеризации, продемонстрированы на рисунке 2.9. Оптимальная глубина *D* анализа данных не зависит количества объектов. Из рисунка 2.9 видно, что при малом количестве кластеров (*k* = 2..5) минимальное значение кластеризации достигается при *D* = 2..4.

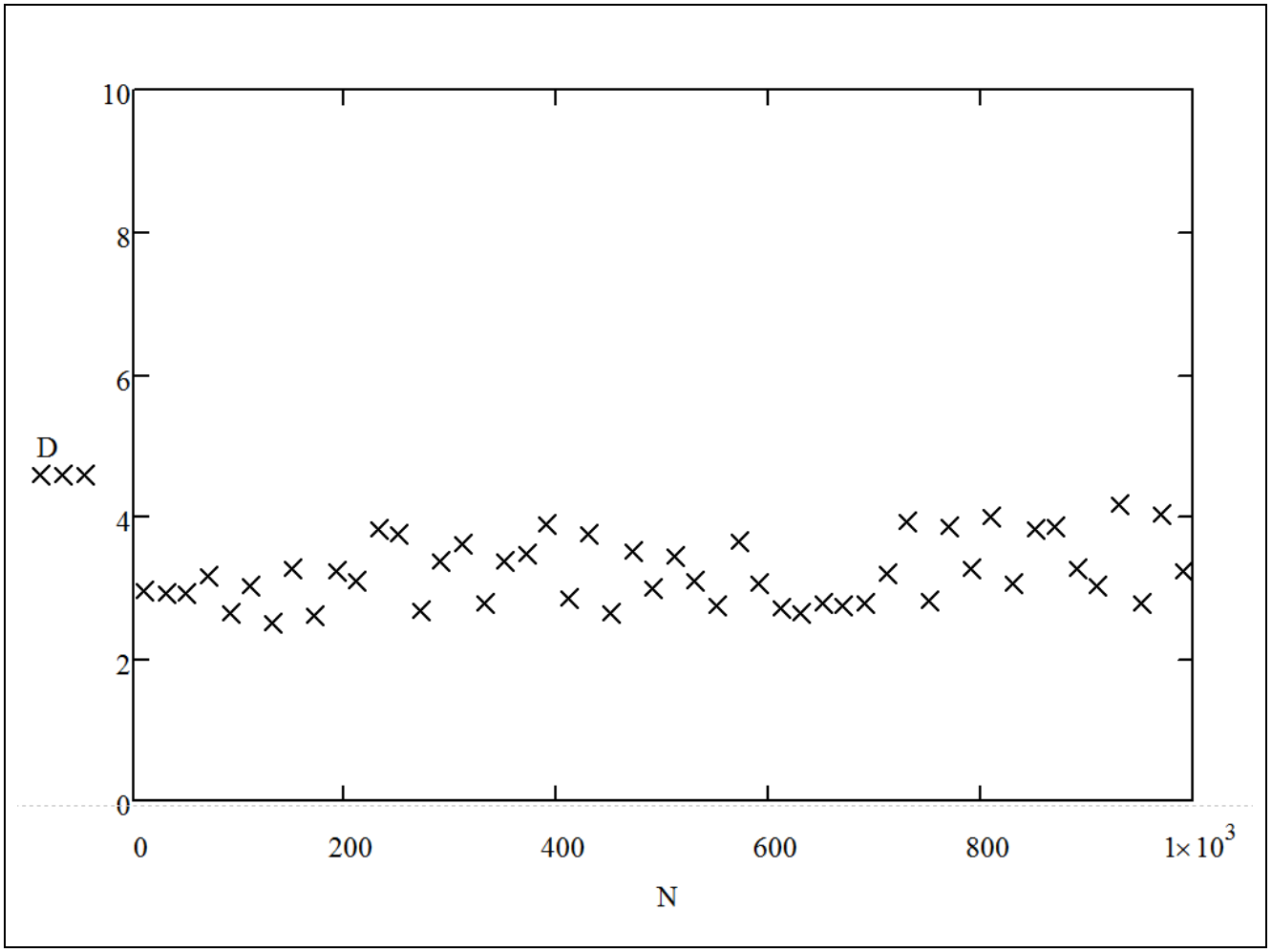


Рисунок 2.9 – Зависимость значение параметра *D* от количества *N* элементов обучающей выборки (при $k = 3$ и $n = 2$)

Полученные экспериментальным путем данные были аппроксимированы линейной функцией $fd(x)$ (рисунок 2.10). Для нахождения данной функции использовался метод наименьших квадратов:

$$Fd(x) = (4{,}658\times10^{-4})\cdot x+2{,}981 \qquad (2.27)$$

Стандартное отклонение при аппроксимации данных представленной функцией составляет:

$$S_{\text{per}} = \sqrt{\frac{1}{n-2}\cdot\left[\sum_i (fd(N_i)-D_i)^2\right]} = 0{,}445 \qquad (2.28)$$

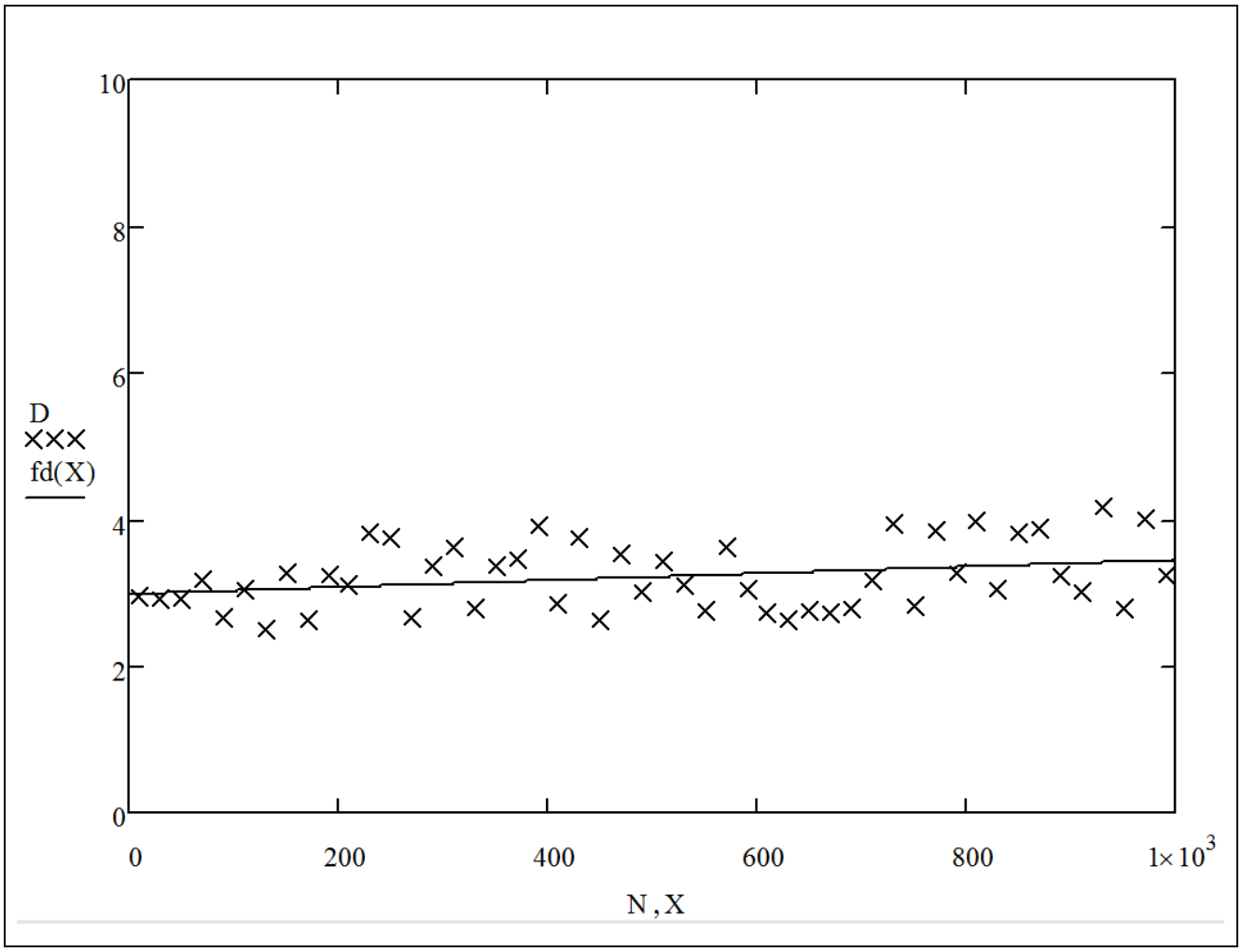
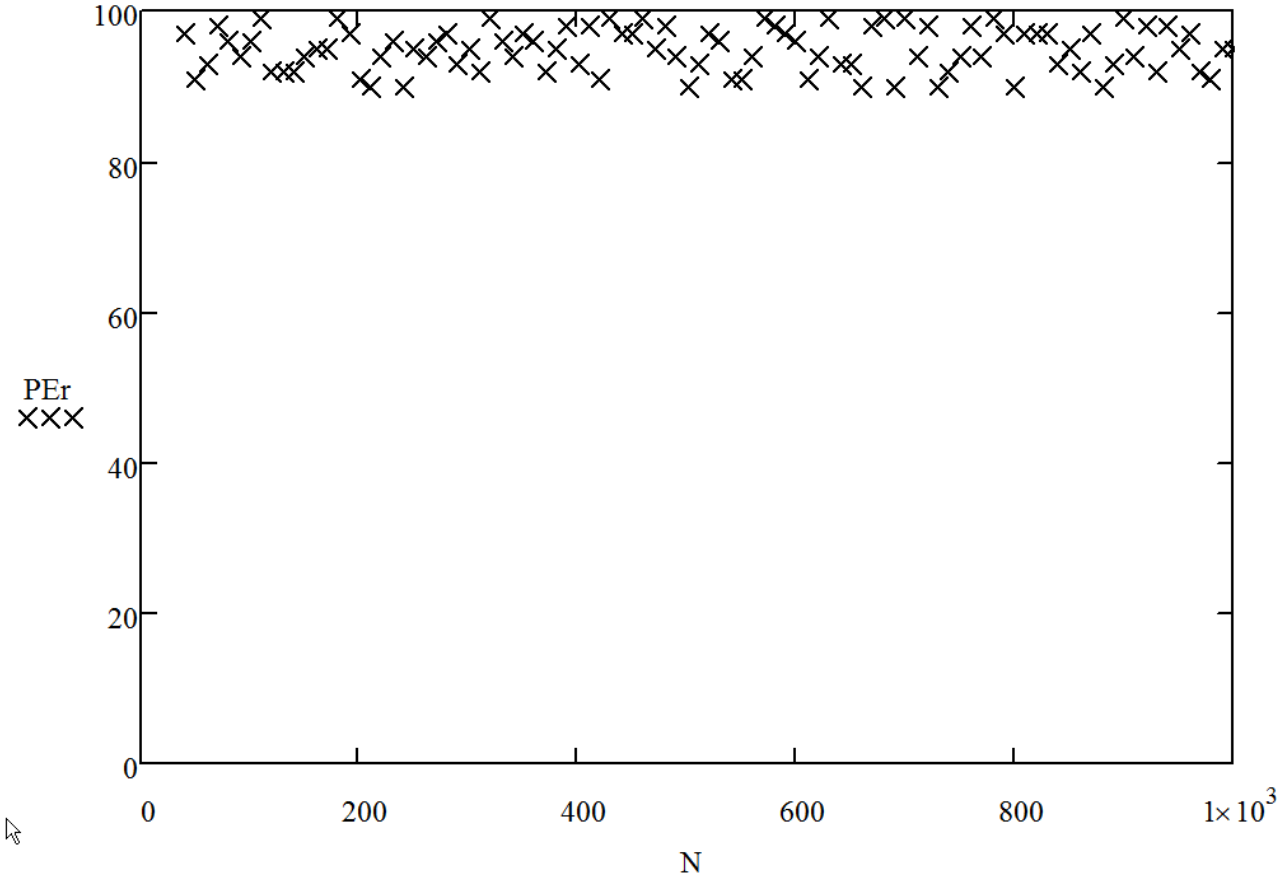


Рисунок 2.10 – Аппроксимация полученных данных (при $k = 3$ и $n = 2$)

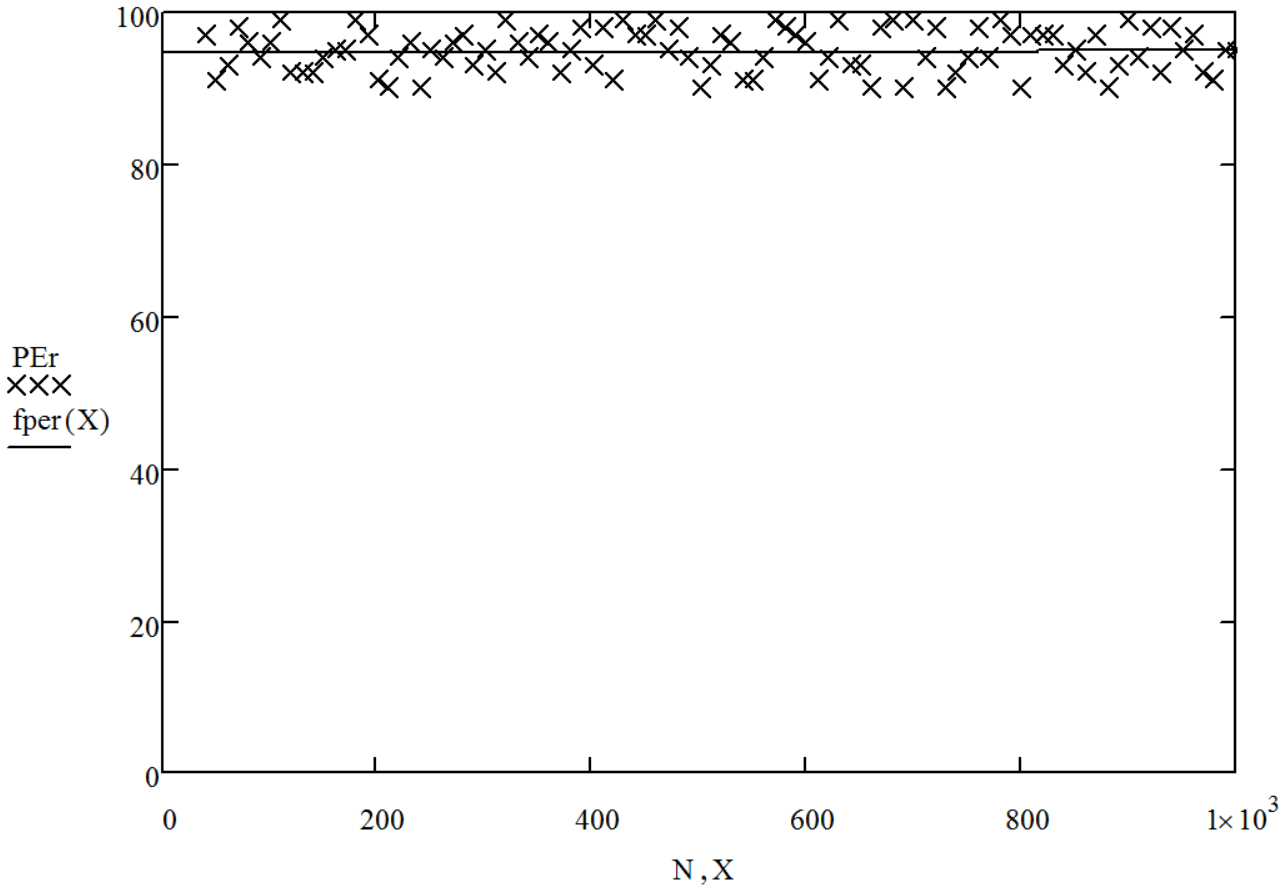
Аналогичные зависимости были получены и при большом количестве кластеров (k = 20…25). При большом количестве кластеров (k = 20), при кластеризации в режиме случайного выбора первоначального расположения центроидов (как это предусмотрено классическим алгоритмом k-means) вероятность нахождения решения с минимально-возможной ошибкой колеблется в пределах от 1-7% (рисунок 2.11).



Per – вероятность нахождения неоптимальной кластерной структуры

N – количество элементов обучающей выборки

Рисунок 2.11 – Результаты вычислительных экспериментов каждого для каждого значения N (при $k = 20$ и $n = 2$)



Per – вероятность нахождения неоптимальной кластерной структуры

fper(*x*) – линейная функция аппроксимации полученных данных

Рисунок 2.12 – Аппроксимация полученных данных (при $k = 20$ и $n = 2$)

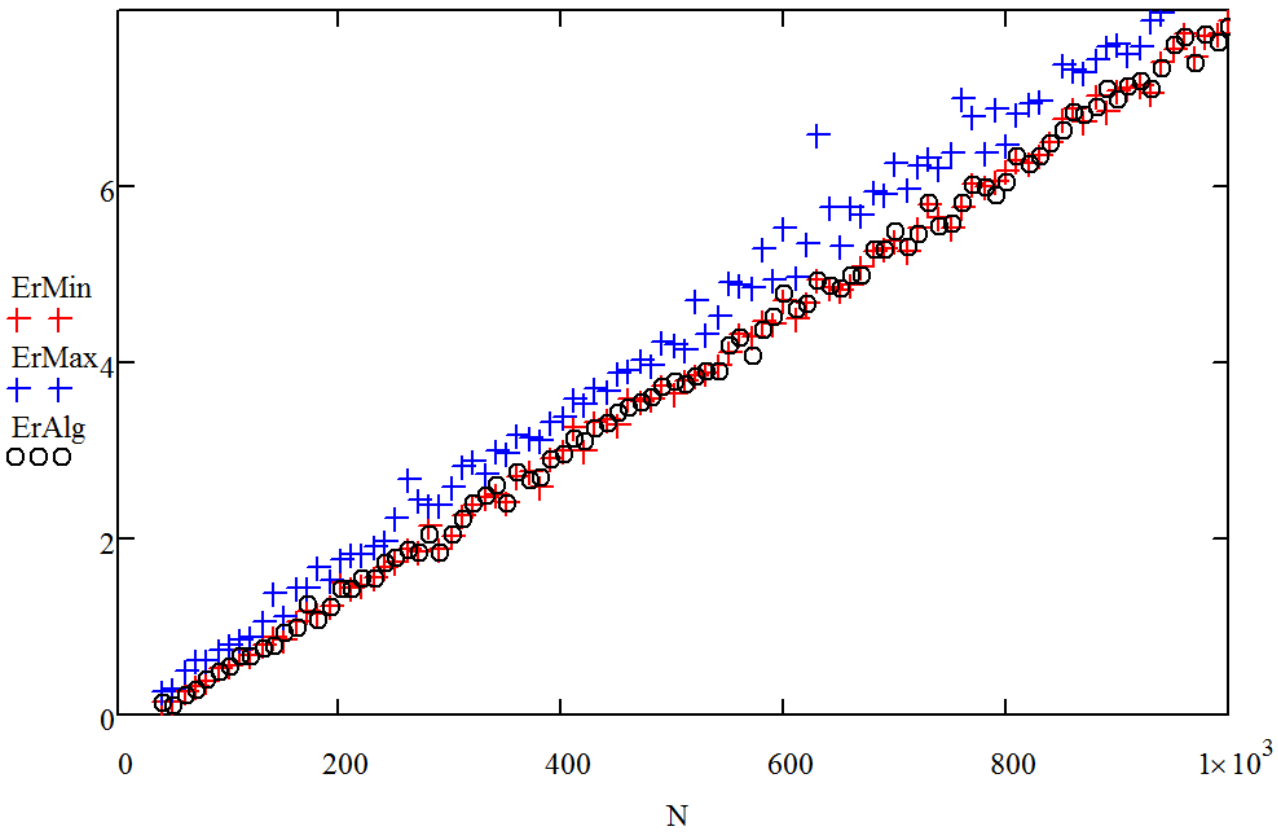
Полученные экспериментальным путем данные были аппроксимированы линейной функцией *fper*(*x*) (рисунок 2.12). Для нахождения данной функции использовался метод наименьших квадратов:

$$fper(x) = (1{,}197\times10^{-4})\cdot x + 94{,}804 \qquad (2.29)$$

Стандартное отклонение при аппроксимации данных представленной функцией составляет:

$$S_{\text{per}} = \sqrt{\frac{1}{n-2}\cdot\left[\sum_i (fper(N_i) - PEr_i)^2\right]} = 2{,}854 \qquad (2.30)$$

При этом найденные максимальные и минимальные значения ошибок увеличиваются пропорционально росту количества записей (рисунок 2.13).



ErMin – минимальная ошибка кластеризации;

ErMax – максимальная ошибка кластеризации;

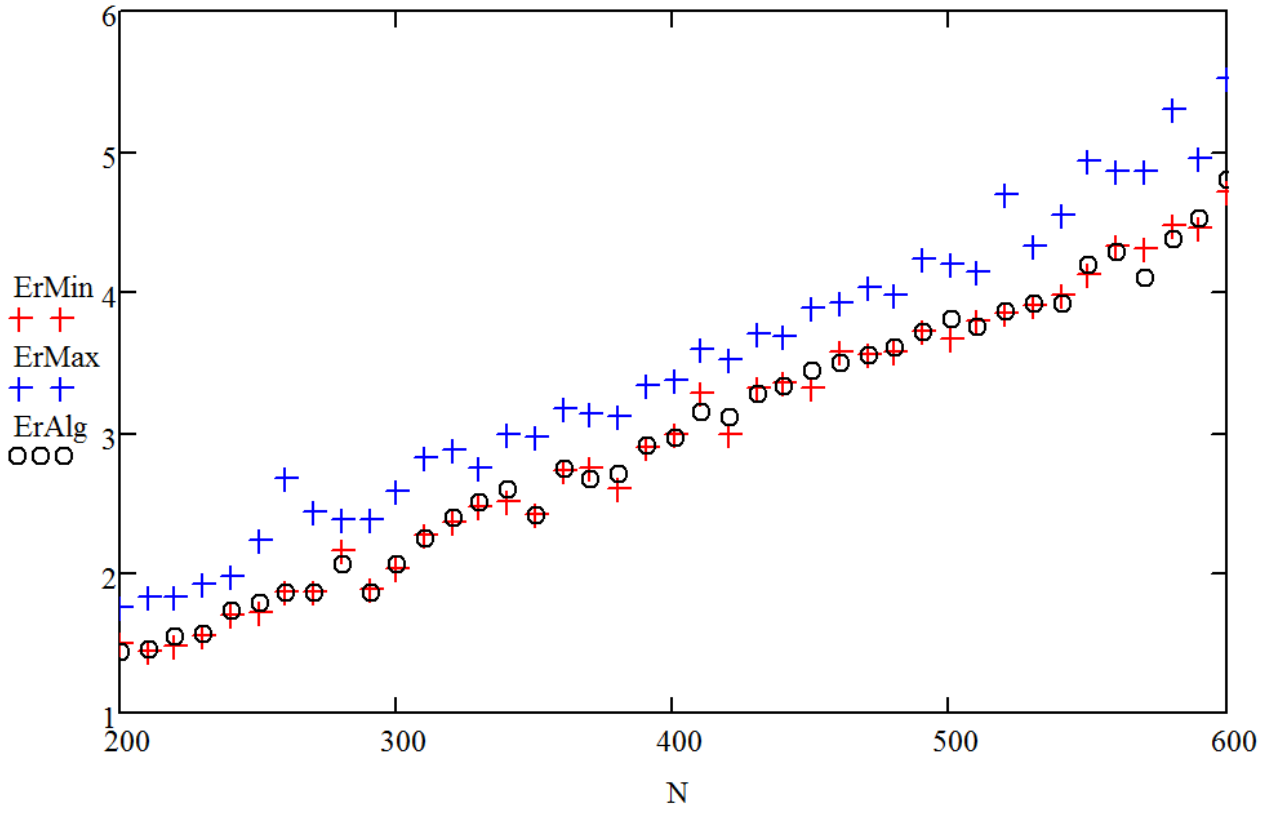
ErAlg – ошибка кластеризации при использовании предложенного алгоритма;

N – количество элементов обучающей выборки.

Рисунок 2.13 – Результаты вычислительных экспериментов каждого для каждого значения N (при $k = 20$ и $n = 2$)

При увеличении графика, представленного на рисунке 2.13, можно заметить, что кривая *ErAlg* ошибки кластеризации данных при использовании алгоритма определения центров кластеров практически совпадает с кривой *ErMin* минимальной ошибки найденной за 100 запусков классического алгоритма k-means (рисунок 2.14). Это означает, что

практически во всех случаях благодаря предложенному алгоритму за один запуск была найдена оптимальная кластерная структура.



ErMin – минимальная ошибка кластеризации;

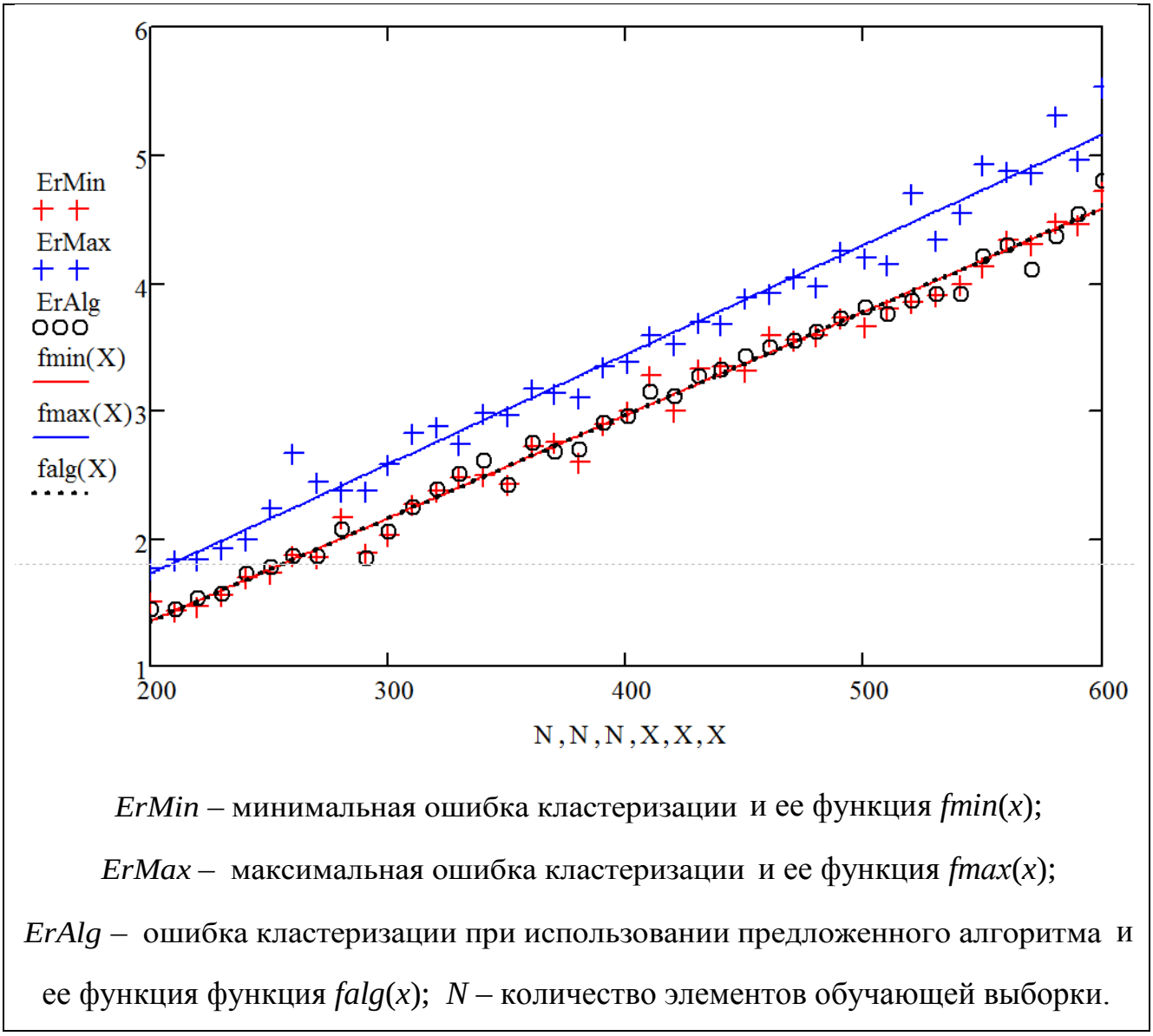
ErMax – максимальная ошибка кластеризации;

ErAlg – ошибка кластеризации при использовании предложенного алгоритма;

N – количество элементов обучающей выборки.

Рисунок 2.14 – Результаты вычислительных экспериментов каждого для каждого значения N (при $k = 20$ и $n = 2$)

Полученные экспериментальным путем данные были аппроксимированы линейными функциями *fmin*(*x*), *fmax*(*x*), *falg*(*x*) (рисунок 2.15).



ErMin – минимальная ошибка кластеризации и ее функция *fmin*(*x*);

ErMax – максимальная ошибка кластеризации и ее функция *fmax*(*x*);

ErAlg – ошибка кластеризации при использовании предложенного алгоритма и ее функция функция *falg*(*x*); *N* – количество элементов обучающей выборки.

Рисунок 2.15 – Аппроксимация полученных данных (при $k = 3$ и $n = 2$)

Для нахождения функций использовался метод наименьших квадратов:

$$fmin(x) = (8{,}071\times10^{-3})\cdot x - 0{,}264 \qquad (2.31)$$

$$fmax(x) = (8{,}582\times10^{-3})\cdot x - 5{,}971\times10^{-3} \qquad (2.32)$$

$$fmin(x) = (8{,}072\times10^{-3})\cdot x - 0{,}261 \qquad (2.33)$$

При этом значения стандартных отклонений составили:

$$S_{\min} = \sqrt{\frac{1}{n-2}\cdot\left[\sum_i (fmin(N_i) - ErMin_i)^2\right]} = 0{,}102 \qquad (2.34)$$

$$S_{\max} = \sqrt{\frac{1}{n-2}\cdot\left[\sum_i (fmax(N_i) - ErMax_i)^2\right]} = 0{,}227 \qquad (2.35)$$

$$S_{\text{alg}} = \sqrt{\frac{1}{n-2} \cdot \left[\sum_i (falg(N_i) - ErAlg_i)^2 \right]} = 0{,}101 \qquad (2.36)$$

Результаты исследования значений глубины D, обеспечивающих минимальную ошибку кластеризации, продемонстрированы на рисунке 2.16. Оптимальная глубина D анализа данных не зависит количества объектов. Из рисунка 2.16 видно, что при большом количестве кластеров (k = 20..25), минимальное значение кластеризации достигается при $D = 7…8$.

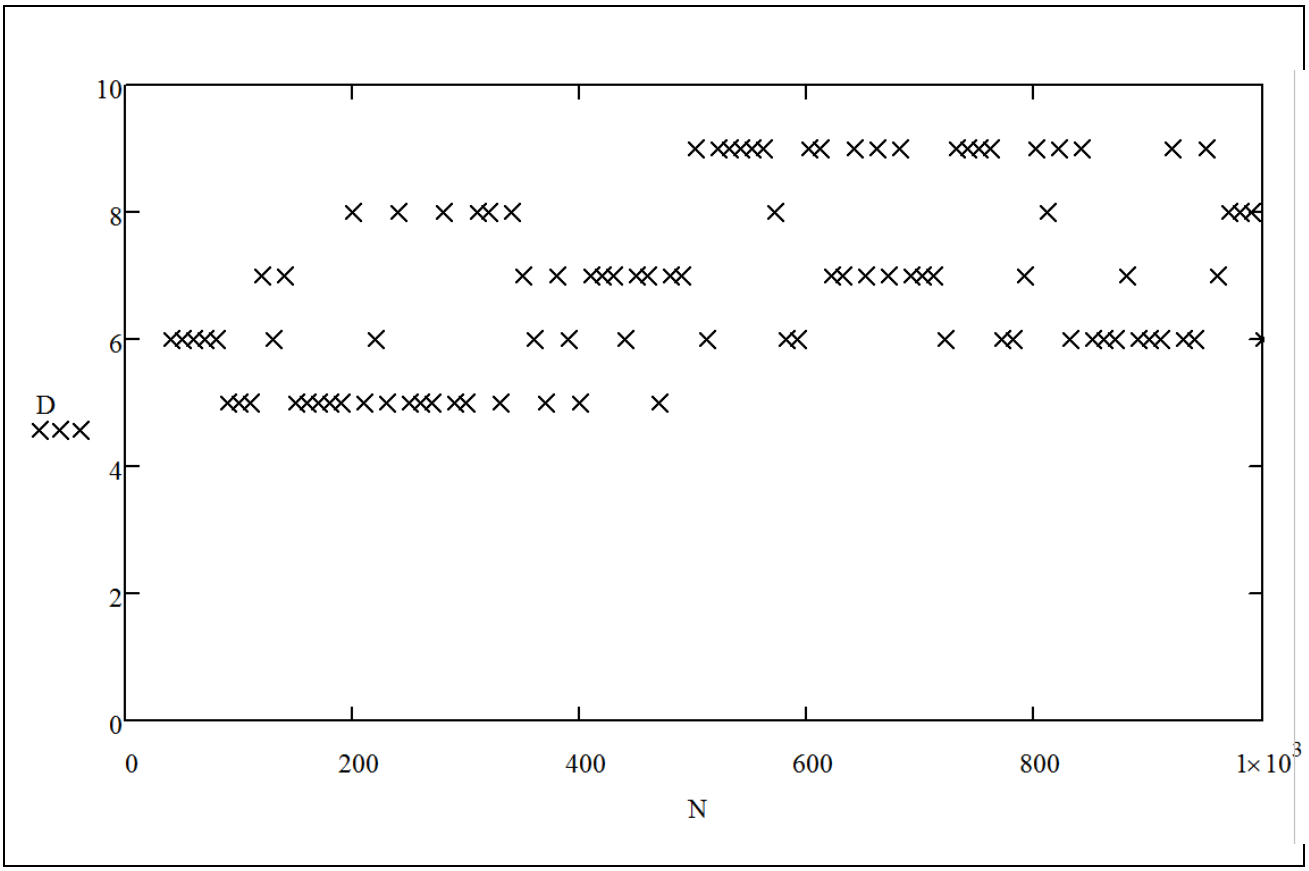


Рисунок 2.16 – Зависимость значение параметра D от количества N элементов обучающей выборки (при $k = 20$ и $n = 2$)

Полученные экспериментальным путем данные были аппроксимированы линейной функцией $fd(x)$ (рисунок 2.17). Для нахождения данной функции использовался метод наименьших квадратов:

$$Fd(x) = (2{,}149\times10^{-3})\cdot x+5{,}749 \qquad (2.37)$$

Стандартное отклонение при аппроксимации данных представленной функцией составляет:

$$S_{\text{per}} = \sqrt{\frac{1}{n-2}\cdot\left[\sum_i (fd(N_i)-D_i)^2\right]} = 1{,}289 \qquad (2.38)$$

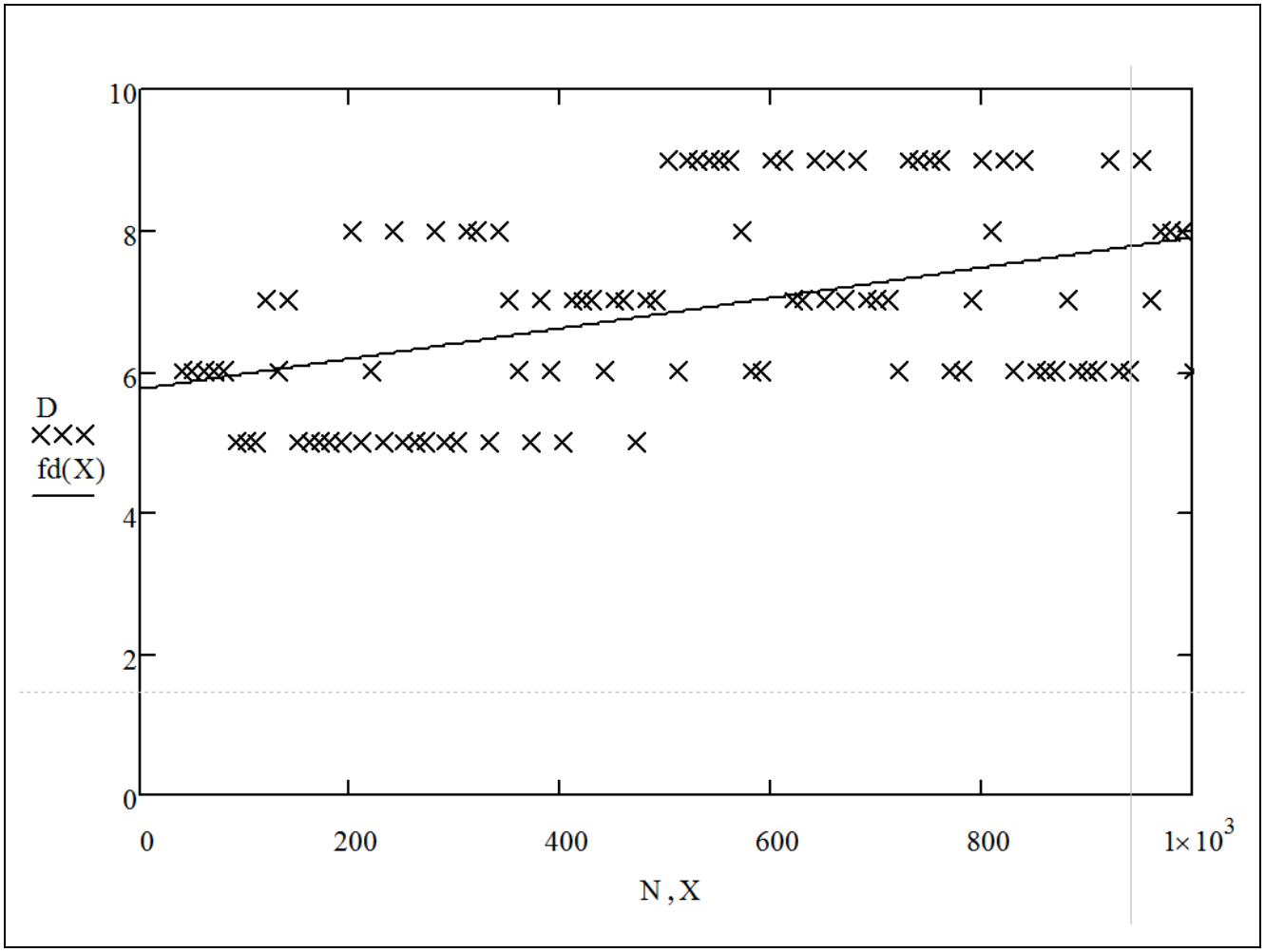


Рисунок 2.10 – Аппроксимация полученных данных (при $k = 20$ и $n = 2$)

Таким образом в ходе вычислительных экспериментов установлено, что вероятность нахождения кластерной структуры с наименьшей ошибкой кластеризации, на обучающей выборке сгенерированной случайным образом, составляет от 5% (при количестве кластеров равных 20) до 30% (при количестве кластеров равных 3).

Экспериментально доказано, что в предложенная методика выбора начального положения центров кластеров, при любом количестве кластеров и размере исходных данных позволяет: снизить ошибку кластеризации за счет обеспечения нахождения оптимальной кластерной структуры, увеличить скорость нахождения решения в среднем на 18%.

Экспериментально установлена независимость глубины анализа данных в предложенном методе от количества объектов.

По результатам множественных вычислительных экспериментов построены аппроксимирующие зависимости связывающие параметры кластеризации с получаемыми результатами. Аппроксимирующие зависимости приведены выше.

3 Программная реализация

3.1 Описание разработанного программного обеспечения

В ходе проведение описанных в данной работе исследований была было разработано программное обеспечение обладающее следующими функциональными возможностями:

1) проведение кластерного анализа данных с использованием алгоритма k-means;

2) расчет качественных характеристик многоитерационного процесса кластерного анализа данных (ошибка кластеризации, распределение объектов по кластерам);

3) ведение лога протекания процесса кластеризации с записью в файл для его последующего анализа;

4) обеспечение программной реализации предложенного алгоритма по выбору начальных центров кластеров;

5) поддержка возможности планирования и проведения массовых вычислительных экспериментов по тестированию и сравнению результатов классической вариации алгоритма k-means и предложенной вариации алгоритма при разной конфигурации исходных данных;

6) экспортирование результатов вычислительного эксперимента в XLS файл для последующего анализа полученных данных и построения графиков.

3.2 Алгоритм работы с приложением

Запуск приложения приведет к открытию основного окна программы (рисунок 3.1). Данное окно состоит из следующих элементов управления, доступных пользователю:

- В верхней части по всей ширине окна расположено меню программы, из которого можно управлять данными, процессом кластеризации, настройками проведения вычислительных экспериментов

- Большую часть окна занимает таблица, предназначенная для отображения обучающей выборки. Количество столбцов таблицы соответствует количеству измерений данных, а количество строк соответствует количеству элементов обучающей выборки;
- Текстовый список, расположенный в нижней части окна предназначен для отображения характеристик процесса кластеризации на каждой итерации выполнения алгоритма.
- В правой верхней части окна расположен блок управления генерацией исходных данных для кластеризации. Возможно задание таких параметров как количества измерений данных, размерность обучающей выборки.
- В правой средней части окна расположен блок управления параметрами кластеров. Здесь можно выбрать количество требуемых кластеров. Также можно определить способ определения начального положения кластеров: случайным образом, как это предусмотрено классическим вариантом алгоритма k-means или с использование разработанной в рамках магистерской работы методики. Также есть возможность задание параметра D – глубины анализа данных;
- В правой средней части окна расположена кнопка, осуществляющая запуск кластеризации данных, содержащихся в обучающей выборке.

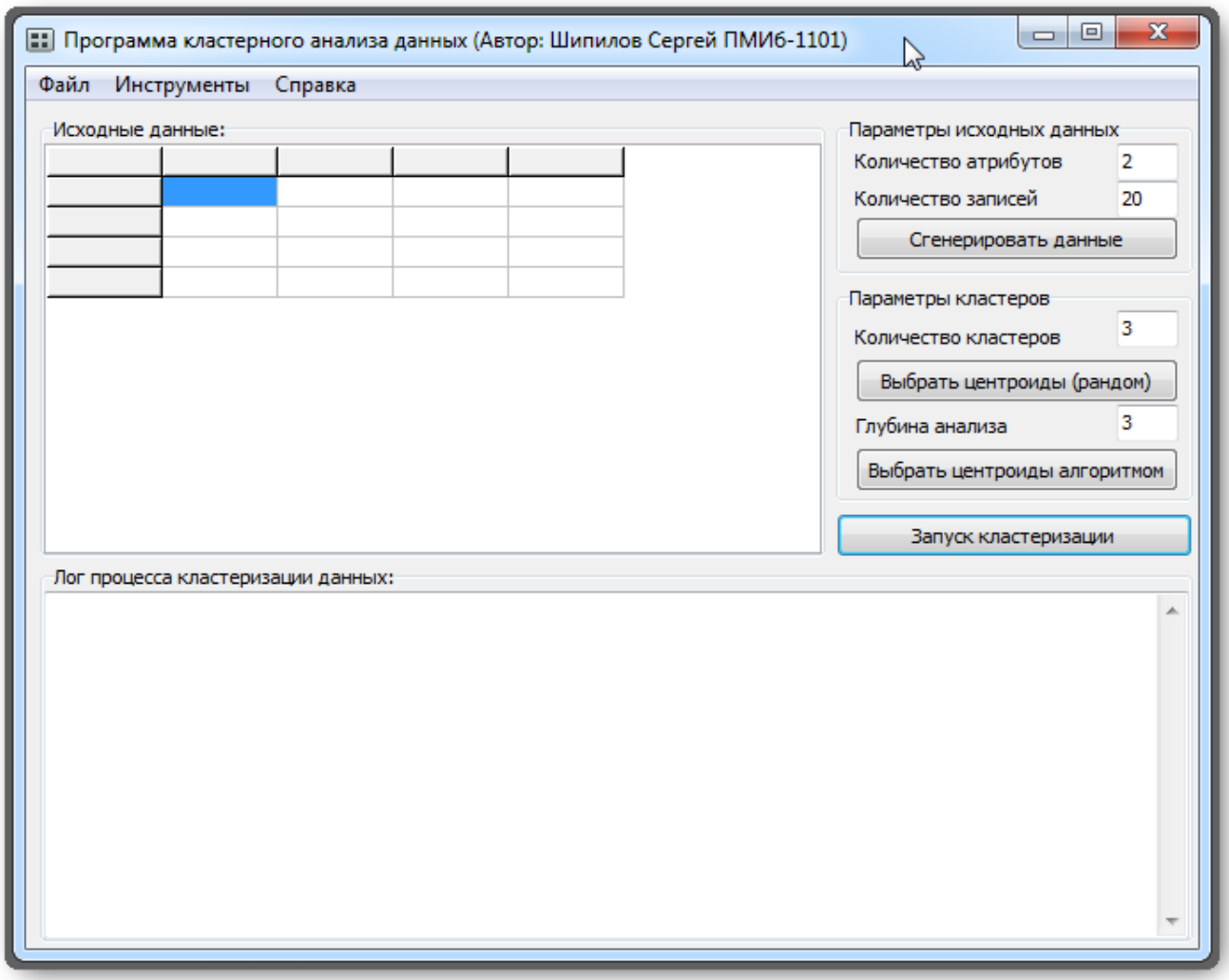


Рисунок 3.1 – Основная форма разработанной программы

Программа поддерживает функцию генерирования данных для проведения вычислительных экспериментов. Для этого необходимо задать количество числовых параметров у объектов обучающей выборки и размер обучающей выборки в группе элементов управления «Парам. исх. данных» (рисунок 3.1).

Затем после нажатия на кнопку сгенерировать обучающая выборка будет создана.

Сообщения об благополучном окончании процесса генерированния данных появится в нижней части окна (рисунок 3.2).

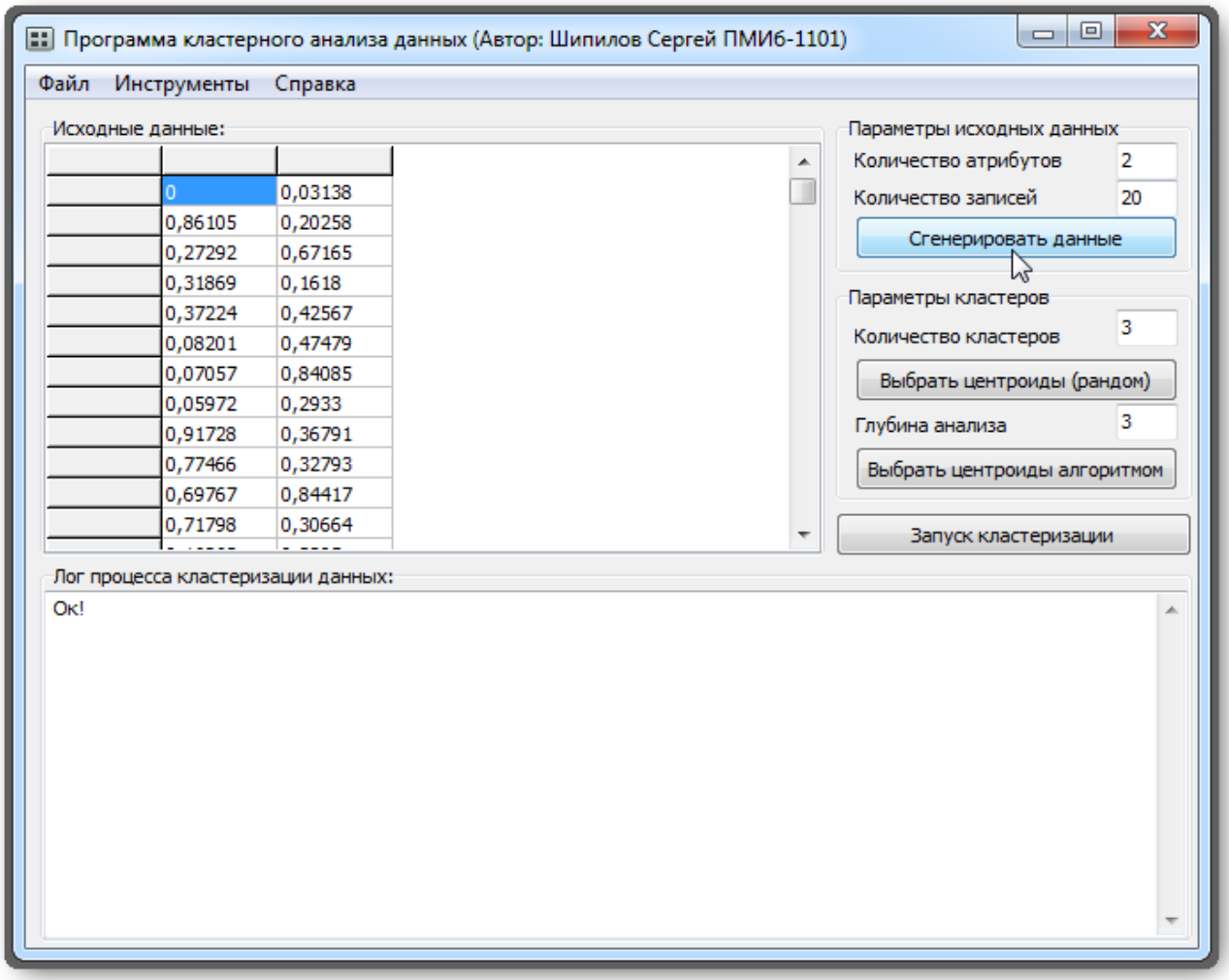


Рисунок 3.2 – Генерирование обучающей выборки

После того, как создана обучающая выборка можно необходимо задать параметры кластеризации данных.

Для этого в окне программы в расположен блок элементов управления «Параметры кластеров». Так в поле «Количество кластеров» с клавиатуры можно задать требуемую конфигурацию кластерной структуры.

Здесь же можно выбрать способ определения начального положения центров кластеров. Нажатием на кнопку «Выбрать центроиды (рандом)» можно центры кластеров задать так, как это предусмотрено классической реализацией алгоритма.

Если нажать на кнопку «Выбрать центроиды алгоритмом», то они будут выбраны способом, описанным во второй главе.

Также в блоке элементов управления «Параметры кластеров» можно задать параметр *D,* который отвечает за глубину анализа данных при выборе кластеров по предложенному алгоритму. Данное значение вводиться в поле «глубина анализа» (подробное описание данного параметра приведено во второй главе). Значение по-умолчанию – 2.

После выбора способа определения начального расположения центров кластеров в нижней части экрана отобразиться в разделе «Лог процесса кластеризации данных» отобразится отчет, содержащий в себе координаты всех центров кластеров.

Результат работы программы при выборе начальных центров кластеров разработанным алгоритмом представлен рисунке 3.3.

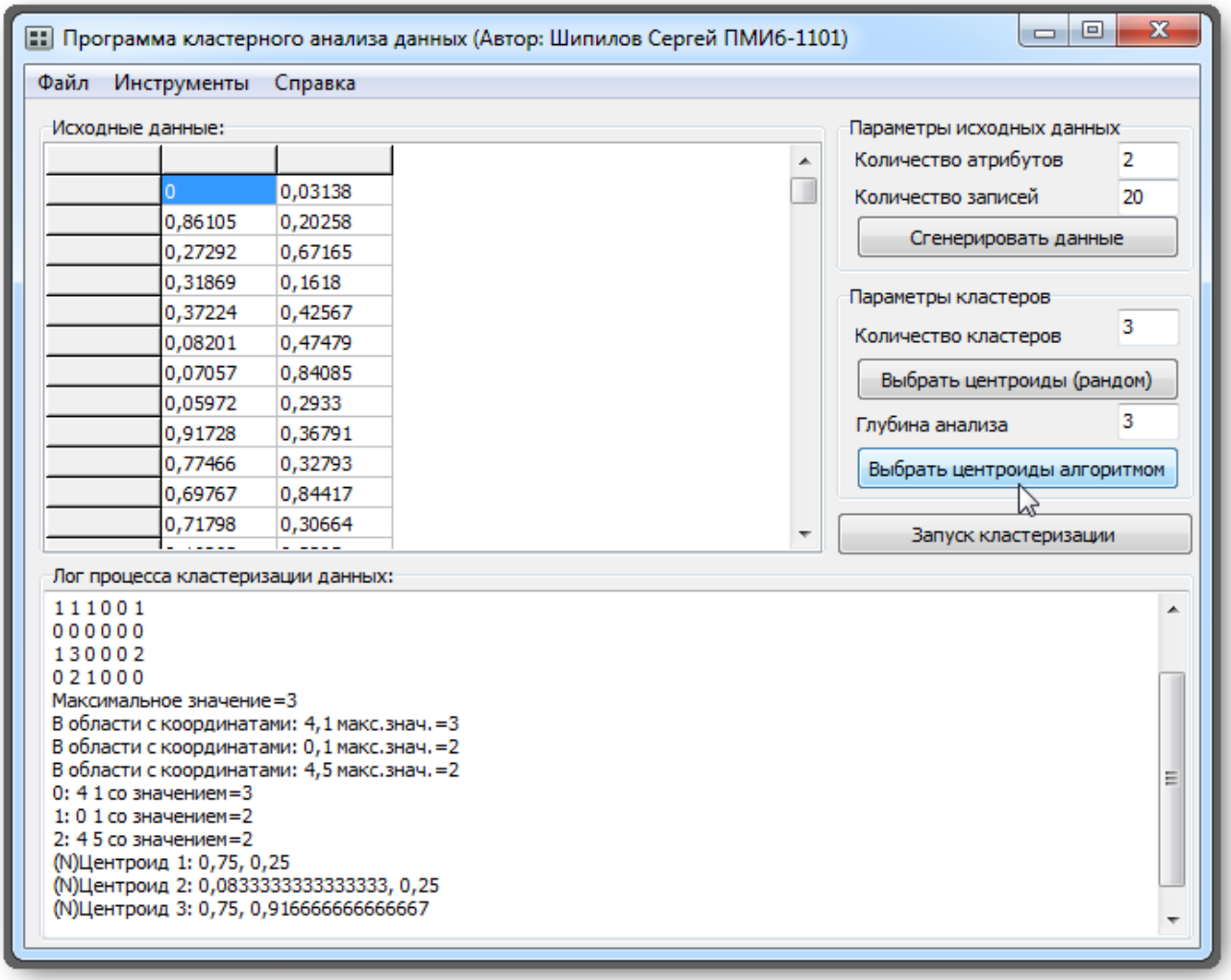


Рисунок 3.3 – Расчет начального положения центров кластеров

Теперь нажав на кнопку «Запуск кластеризации» можно запустить процесс кластеризации данных, содержащихся в обучающей выборке.

Кластеризация выполняется в соответствии с алгоритмом k-means для каждой итерации алгоритма в поле «Лог процесса кластеризации данных» буду отображены следующие данные (рисунок 3.4):

1. Порядковый номер итерации.
2. Распределение объектов по кластерам.
3. Координаты центров кластеров.
4. Ошибка кластеризации.
5. Количество объектов изменивших свою метку кластера по сравнению с предыдущей итерацией.

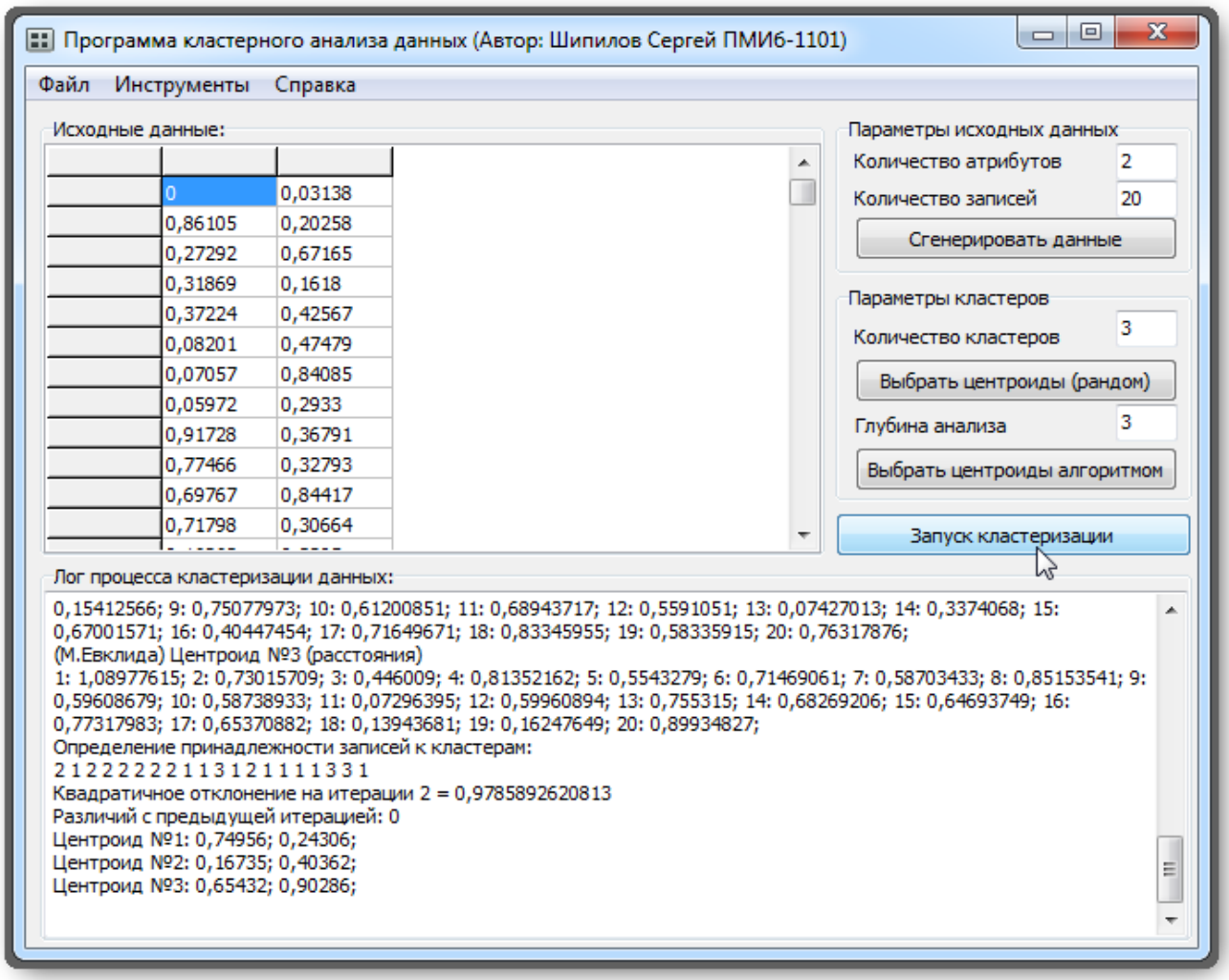


Рисунок 3.4 – Кластеризация данных

В приложении реализована функции записи исходной обучающей выборки и характеристик протекания процесса кластеризации данных в текстовый файл.

Для этого в меню программы необходимо выбрать соответствующие пункты (рисунок 3.5).

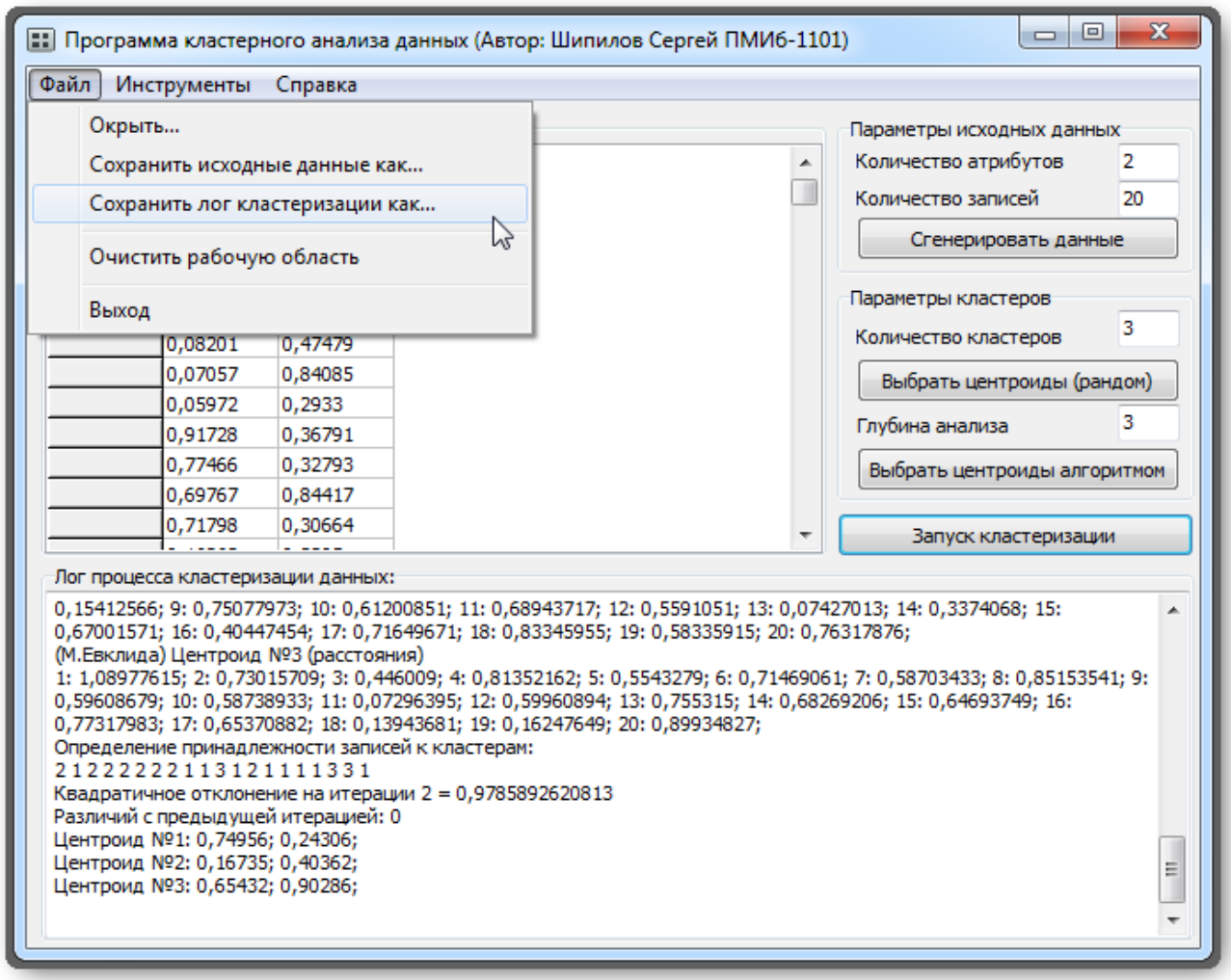


Рисунок 3.5 – Главное меню программы

3.3 Реализация модуля вычислительных экспериментов

В разработанном приложении реализован функционал автоматизированного проведения вычислительных экспериментов.

Данный функционал позволяет накапливать статистическую информацию по результатам кластеризации данных с использованием предложенного алгоритма выбора начального положения кластеров и без его использования (классическая реализация k-means).

Вся полученная с помощью данного модуля информация представлена в виде графиков во второй главе.

Чтобы открыть модуль автоматизации вычислительных экспериментов необходимо выбрать соответствующий пункт (рисунок 3.6) в главном меню программы.

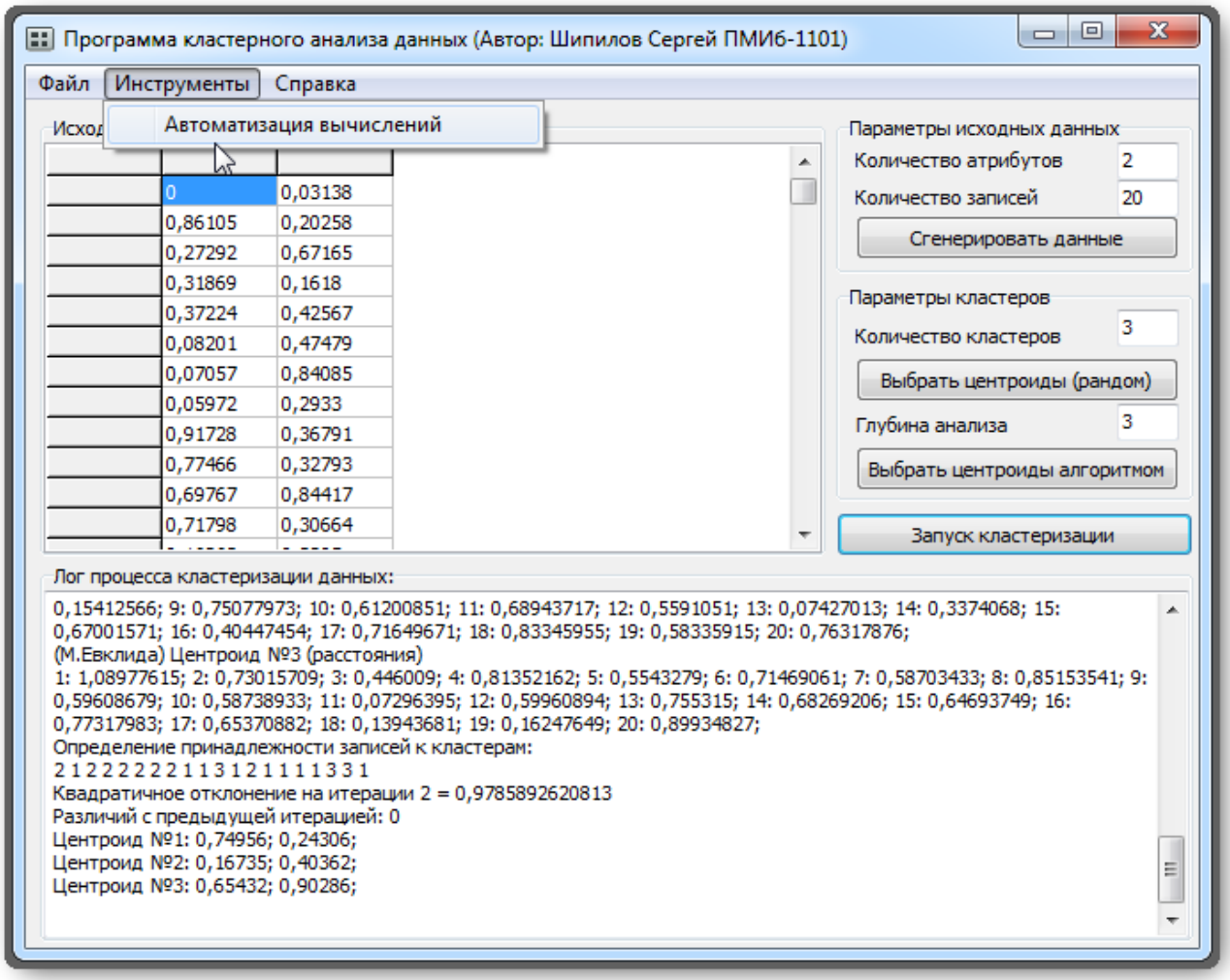


Рисунок 3.6 – Пункт меню для открытия модуля проведения вычислительных экспериментов

Основные функциональные возможности модуля автоматизации вычислительных экспериментов:

1. Автоматизированный проход по всем заданным параметрам кластеризации в выбранном интервале.

2. Поддержка возможности многократного запуска процесса кластеризации на одних и тех же данных (для проверки сходимости к локальному/глобальному решению)

3. Расчёт и фиксирование наименьшей и наибольшей ошибки кластеризации для каждого набора данных.

4. Определение наибольшей и наименьшей скорости нахождения кластерной структуры для одного и того же набора данных.

5. Возможность сравнения результатов кластеризации данных при классическом варианте алгоритма k-means и с использованием предложенного подхода.

6. Подсчет количества итераций, требуемых для нахождения кластерной структуры данных.

7. Поддержка экспорта результатов проведения вычислительных экспериментов в XLS-файл.

Все перечисленные возможности реализованы в отдельном окне приложения, которое представлено на рисунке 3.7.

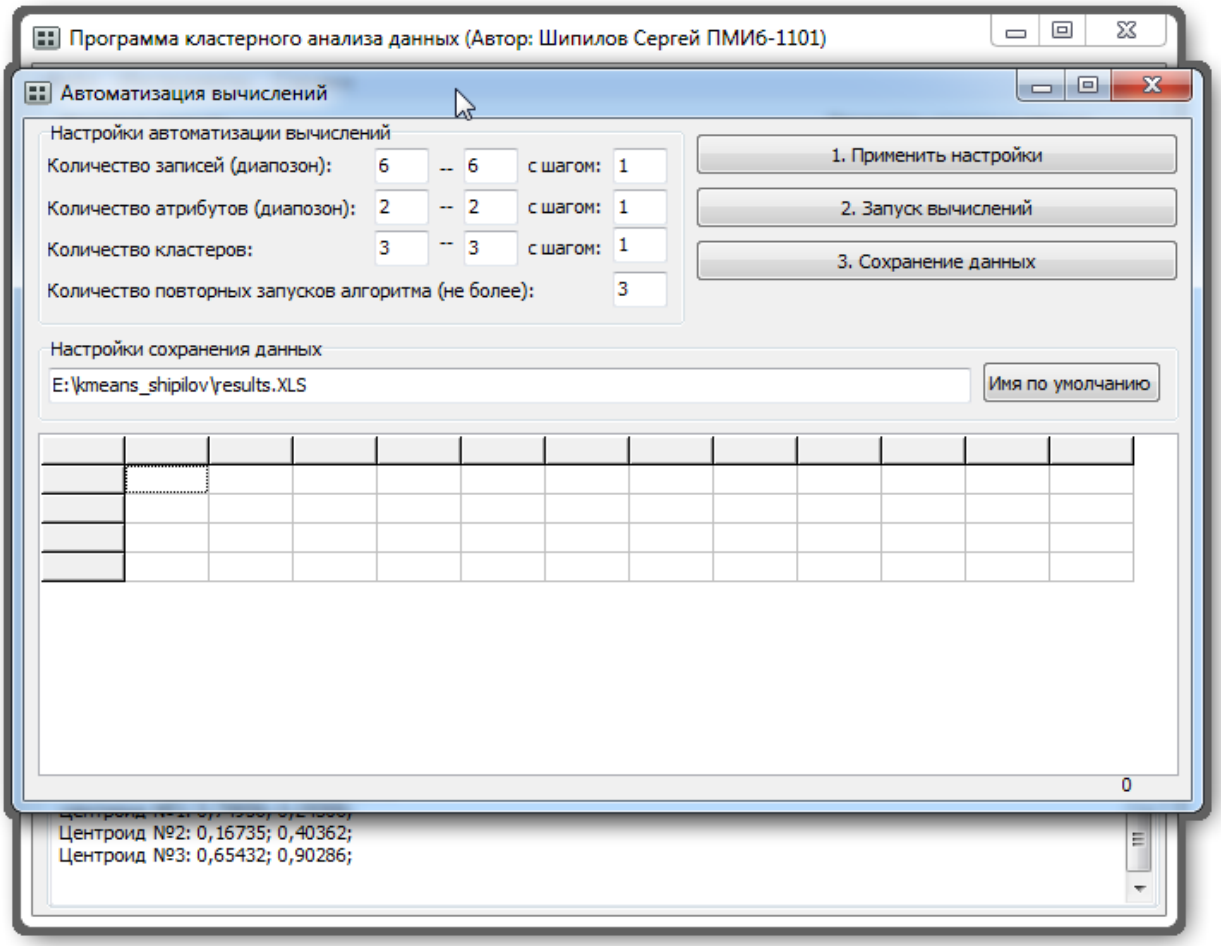


Рисунок 3.7 – Форма проведения вычислительного эксперимента

При планировании вычислительных экспериментов с использованием алгоритма кластеризации данных k-means можно задавать следующими характеристиками:

1. Диапазон и шаг изменения количества объектов в обучающей выборке данных.
2. Диапазон и шаг изменения размера вектора входных параметров, описывающих объекты обучающей выборки.
3. Диапазон и шаг изменения количества кластеров.
4. Количество запусков кластеризации на одной и той же обучающей выборке (необходимо для сбора статистических сведений).

Параметрами вычислительного эксперимента можно управлять с помощью блока элементов «Настройки автоматизации вычислений».

Окончание планирования вычислительного эксперимента осуществляется кнопкой «Применить настройки».

При этом заданные параметры сразу отобразятся в таблице результатов (рисунок 3.8).

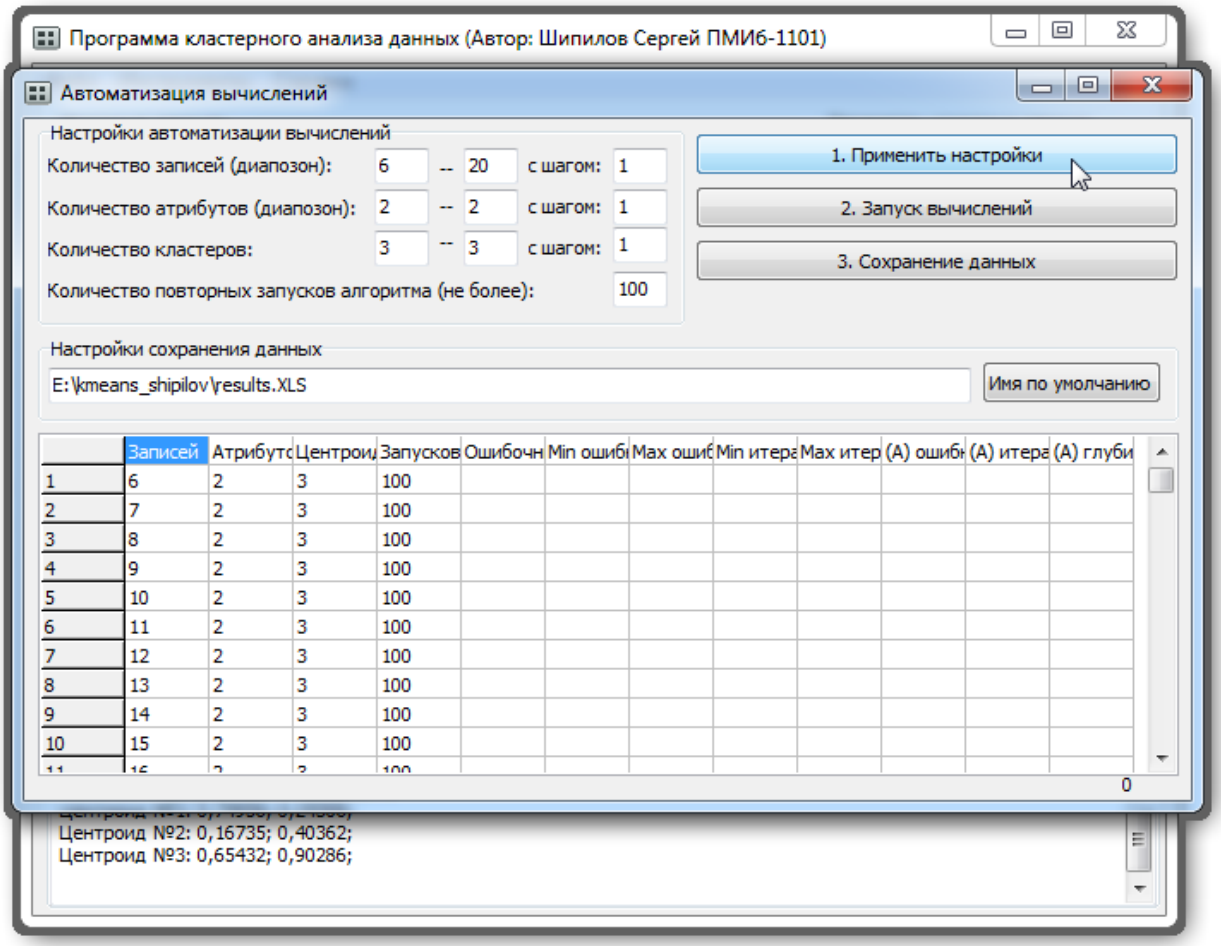


Рисунок 3.8 – Настройка параметров вычислительного эксперимента

Как только заданы параметры вычислительного эксперимента, становиться возможен запуск расчетов. Для этого в окне предусмотрена кнопка «Запуск вычислений».

Результаты проведенных вычислений будут постепенно отображаться в таблице с результатами. Пример отображения результатов вычислительного эксперимента представлен на рисунке 3.9.

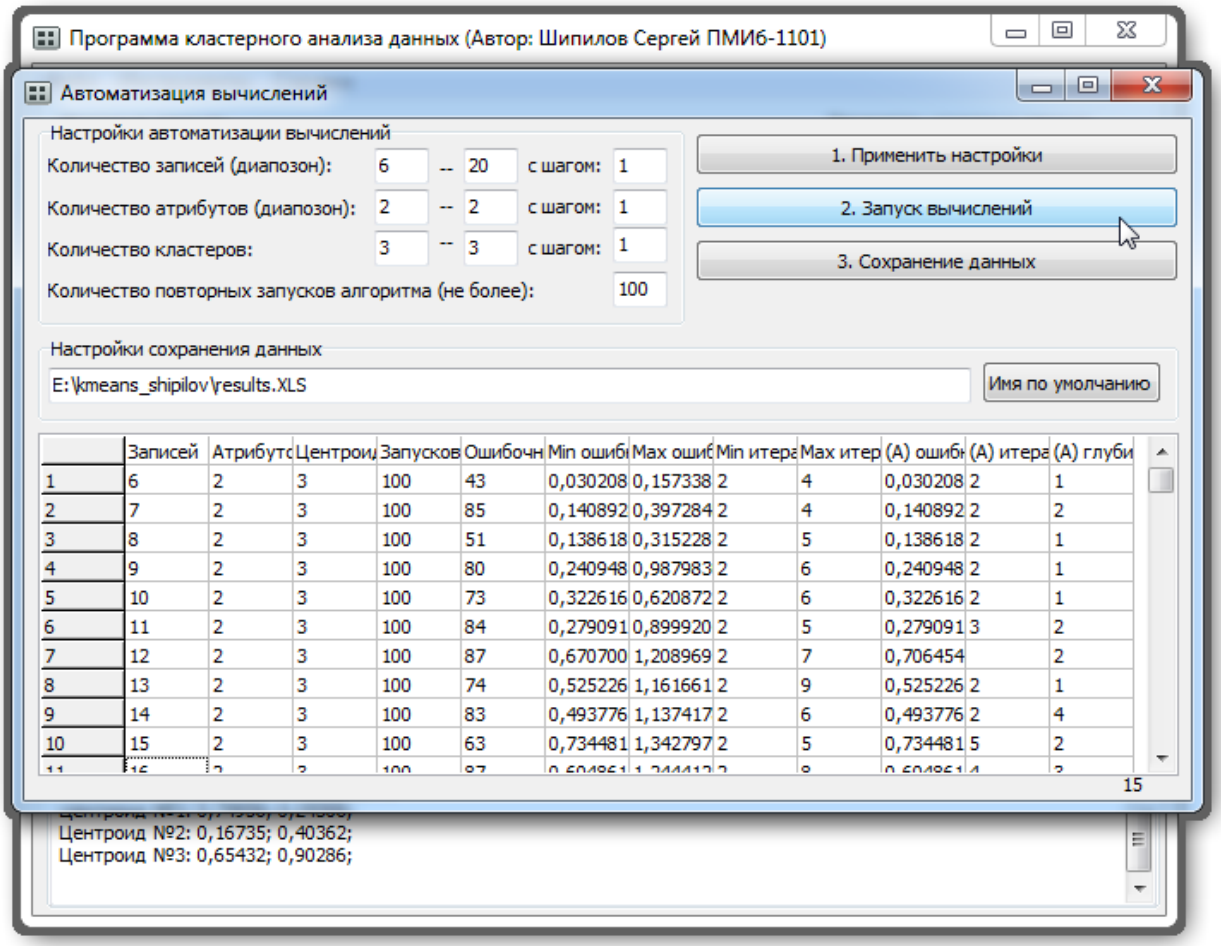
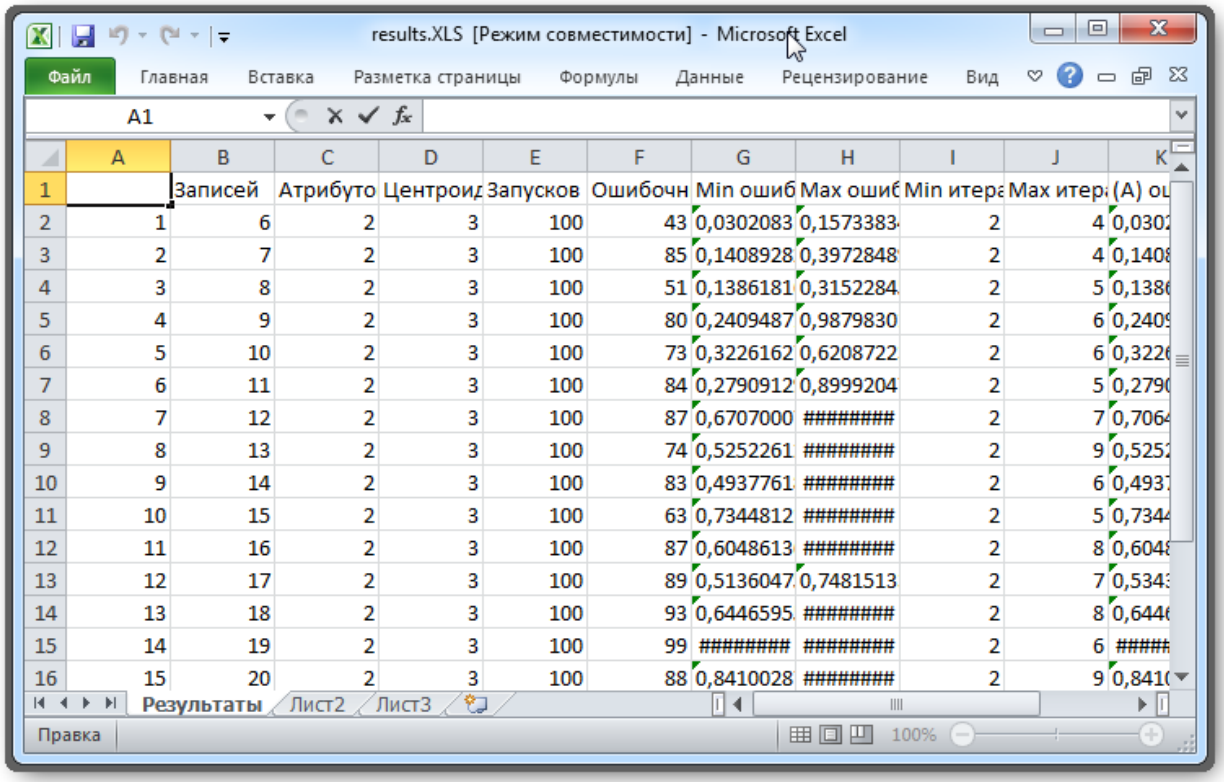


Рисунок 3.9 – Форма с результатами выполнения вычислительного эксперимента

В приложении предусмотрен функционал экспорта данных вычислительного эксперимента в XLS-файл. Данный файл создается по нажатию на кнопку «Сохранение данных».

XLS-файл с именем results сохраняется в паке с программой рядом с исполняемым файлом.

Пример такого XLS-файл представлен на рисунке 3.10 .



	A	B	C	D	E	F	G	H	I	J	K
1		Записей	Атрибуто	Центроид	Запусков	Ошибочн	Min ошиб	Max ошиб	Min итера	Max итер	(A) ош
2	1	6	2	3	100	43	0,0302083	0,1573383	2	4	0,0302
3	2	7	2	3	100	85	0,1408928	0,3972848	2	4	0,1408
4	3	8	2	3	100	51	0,1386181	0,3152284	2	5	0,1386
5	4	9	2	3	100	80	0,2409487	0,9879830	2	6	0,2409
6	5	10	2	3	100	73	0,3226162	0,6208722	2	6	0,3226
7	6	11	2	3	100	84	0,2790912	0,8999204	2	5	0,2790
8	7	12	2	3	100	87	0,6707000	########	2	7	0,7064
9	8	13	2	3	100	74	0,5252261	########	2	9	0,5252
10	9	14	2	3	100	83	0,4937761	########	2	6	0,4937
11	10	15	2	3	100	63	0,7344812	########	2	5	0,7344
12	11	16	2	3	100	87	0,6048613	########	2	8	0,6048
13	12	17	2	3	100	89	0,5136047	0,7481513	2	7	0,5343
14	13	18	2	3	100	93	0,6446595	########	2	8	0,6446
15	14	19	2	3	100	99	########	########	2	6	######
16	15	20	2	3	100	88	0,8410028	########	2	9	0,8410

Рисунок 3.10 – Экспортирование данных в XLS-файл

Заключение

По результатам проведенных исследований были сделаны следующие выводы:

1. Одно из направлений совершенствования методов машинного обучения является повышение их степени автоматизации за счет снижение участие людей в анализе данных.

2. В машинном обучении существует алгоритм, предназначенный для кластеризации данных – k-means. Он автоматически анализирует исходные данные и подбирает оптимальную кластерную структуру. Одной из главных его проблем является возможность локального решение (неоптимальной кластерной структурой). Установлено, что стохастичность получаемых результатов связана со случайным выбором начального положения центров кластеров

3. Установлено, что совершенствование алгоритма k-means возможно путем разработки научного обоснованного подхода определения начального положения кластеров при анализе данных.

4. Предложена реализация метода выбора данных, основанная на поиске локальных скоплений объектов в пространстве входных параметров данных (подробное описание представлено в пункте 2.3).

5. Установлено, что вероятность нахождения кластерной структуры с наименьшей ошибкой кластеризации, на обучающей выборке сгенерированной случайным образом, составляет от 5% (при количестве кластеров равных 20) до 30% (при количестве кластеров равных 3).

4. Экспериментально доказано, что в предложенная методика выбора начального положения центров кластеров, при любом количестве кластеров и размере исходных данных позволяет: снизить ошибку кластеризации за счет обеспечения нахождения оптимальной кластерной структуры, увеличить скорость нахождения решения в среднем на 18%.

5. Экспериментально установлена независимость глубины анализа данных в предложенном методе от количества объектов (подробное описание представлено в пункте 2.5).

6. По результатам множественных вычислительных экспериментов построены аппроксимирующие зависимости связывающие параметры кластеризации с получаемыми результатами. Аппроксимирующие зависимости приведены в пункте 2.5.

7. Разработано приложение для изучения процесса кластеризации данных и проведения вычислительных экспериментов. Приложение реализует кластеризацию данных с использование предложенного подхода.

8. Научная новизна исследования – доказано что скорость и точность результатов кластеризации данных с помощью алгоритма k-means может быть увеличена путем обоснованного задания начального положения центров кластеров. Это также приведет к снижению вероятности схождения алгоритма к локальному решению.

9. Практическая значимость работы заключается в разработке методики выбора начального положения центров кластеров, что позволяет увеличить точность и скорость кластеризации данных с использованием алгоритма k-means.

Список используемых источников

1. Langley, P. Research Papers in Machine Learning / Pat Langley // Machine Learning. – 1987. – №3. – pp. 195-198.
2. Carbonell, J. Machine Learning: A Maturing Field / Jaime Carbonell // Machine Learning. – 1992. – №9. – pp. 5-7
3. Ding, S. Extreme learning machine with kernel model based on deep learning / Shifei Ding, Lili Guo, Yanlu Hou // Neural Computing and Applications. – 2016. – pp. 1-10.
4. Zong, W. Learning to Rank with Extreme Learning Machine / Weiwei Zong, Guang-Bin Huang // Neural Processing Letters. – 2014. – №3. – pp. 155-166.
5. Zhu, C. Double-fold localized multiple matrix learning machine with Universum / Changming Zhu // Pattern Analysis and Applications. – 2016. – №3. – pp. 1-28.
6. Chen, Z. Real-time transient stability status prediction using cost-sensitive extreme learning machine / Zhen Chen, Xianyong Xiao, Changsong Li, Yin Zhang, Qingquan Hu // Neural Computing and Applications. – 2016. – №27. – pp. 330-333
7. Blum, A. Special Issue on New Theoretical Challenges in Machine Learning / Avrim Blum, Philip M. Long // Algorithmica. – 2015. – №72. – pp. 191-192
8. Bing, L. Lifelong machine learning: a paradigm for continuous learning / Bing Liu // Machine learning. – 2016. – №3. – pp. 1-3.
9. Günnemann, S. Machine Learning Meets Databases / Stephan Günnemann // Datenbank-Spektrum. – 2017. – №17. – pp. 77-83
10. Langley, P. Research papers in machine learning / Pat Langley // Machine Learning. – 1987. – №2. – pp. 195-198
11. Abdolrazzaghi, M. Fast-forward solver for inhomogeneous media using machine learning methods: artificial neural network, support vector machine

and fuzzy logic / Mohammad Abdolrazzaghi, Soheil Hashemy, Ali Abdolali // Neural Computing and Applications. – 2016. – №3. – pp. 1-9.

12. Carbonell, J. Machine Learning: A maturing field / Jaime Carbonell // Machine Learning. – 1992. – №9. – pp. 5-7.

13. Shen, Q. Decay-weighted extreme learning machine for balance and optimization learning / Qing Shen, Xiaojuan Ban, Ruoyi Liu, Yu Wang // Machine Vision and Applications. – 2017. – №3. – pp. 1-11.

14. Blocki, M.M. Clustering Methods: A History of k-Means Algorithms / M.M. Blocki // Selected Contributions in Data Analysis and Classification. – Springer Berlin Heidelberg, 2009. – 161-174 p.

15. Celebe, H.M. Partitional Clustering Algorithms / H.M. Celebe. – Springer International Publishing Switzerland, 2015. – 415 p.

16. Wuy, H. Advances in K-means Clustering : A Data Mining Thinking / H. Wuy. – Springer Berlin Heidelberg, 2012. – 182 p.

17. Liotte, D. Smart Anomaly Detection for Sensor System: Computational Intelligence Techniques for Sensor Networks and Applications / D. Liotte, M. Borsman, F. Iacce. – Springer Publisher, 2016. – 151 p.

18. Mucherin, B. Data Mining in Agriculture / B. Mucherio, M.G. Papajorgj, C.B. Pardalose. – Springer New York, 2008. – 273 p.

19. Mirkins, M. Core Concepts in Data Analysis: Summarization, Correlation and Visualization / M. Mirkins. – Springer London, 2012. – 391 p.

20. Tryone, C.B. Cluster analysis / C.B. Tryone. – London: Ann Arbor Edwards Bros, 1940. – 140 p.

21. Li, X. Multiple-kernel-learning-based extreme learning machine for classification design / Xiaodong Li, Weijie Mao, Wei Jiang // Neural Computing and Applications. – 2016. – №27. – pp. 175-184.

22. Alom, M.Z. State Preserving Extreme Learning Machine: A Monotonically Increasing Learning Approach / Md. Zahangir Alom, Paheding

SidikeTarek M. TahaVijayan K. Asari // Neural Processing Letters. – 2017. – №45. – pp. 703-725

23. Mirza, B. Weighted Online Sequential Extreme Learning Machine for Class Imbalance Learning / Bilal MirzaZhiping Lin, Kar-Ann Toh // Neural Processing Letters. – 2013. – №38. – pp. 465-586.

24. Ertuğrul, Ö.F. A novel machine learning method based on generalized behavioral learning theory / Ömer Faruk Ertuğrul, Mehmet Emin Tağluk // Neural Computing and Applications. – 2016. – №3. – pp. 1-19.

25. Stein, G. Learning in context: enhancing machine learning with context-based reasoning / Gary Stein, Avelino J. Gonzalez // Applied Intelligence. – 2014. – №41. – pp. 709-724.

26. Li, B. The extreme learning machine learning algorithm with tunable activation function / Bin Li, Yibin Li, Xuewen Rong // Neural Computing and Applications. – 2013. – №22. – pp. 531-539.

27. Perlich, C. Machine learning for targeted display advertising: transfer learning in action / C. Perlich, B. Dalessandro, T. RaederO. Stitelman, F. Provost // Machine Learning. – 2014. – №95. – pp. 103–127.

28. Chang, KC. Machine learning by imitating human learning / Kuo-Chin Chang, Tzung-Pei Hong, Shian-Shyong Tseng // Minds and Machines. – 1996. – №6. – pp. 203–228

29. Dietterich, T.G. Structured machine learning: the next ten years / Thomas G. Dietterich, Pedro Domingos, Lise Getoor, Stephen Muggleton, Prasad Tadepalli // Machine Learning. – 2008. – №3. – pp. 23-35.

30. Carbonell, J. Special Issue of Machine Learning on Information Retrieval Introduction / Jaime Carbonell, Yiming Yang, William Cohen // Machine Learning. – 2000. – №39. – pp. 99-101.

31. Tamura, H. Unsupervised learning method for a support vector machine and its application to surface electromyogram recognition / Hiroki

Tamura, Shuji Kawano, Koichi Tanno // Artificial Life and Robotics. – 2009. – №14. – pp. 23-35.

32. Langlei, M. Machine learning as an experimental science / Mat Langley // Machlne Leaming. – 1987. – №4. – pp. 5-8.

33. Wiens, J. Editorial: special issue on machine learning for health and medicine / Jenna Wiens, Byron C. Wallace // Machine Learning. – 2016. – №3. – pp. 305-307.

34. Zhou, ZH. Learnware: on the future of machine learning / Zhi-Hua Zhou // Frontiers of Computer Science. – 2016. – №10. – pp. 589-590.

35. Hemmi, H. Learning and relocation capabilities of a CAM-brain machine / Hitoshi Hemmi, Kazuhiko Shinozawa, Tomofumi Hikage, Katumori Shimohara // Artificial Life and Robotics. – 1999. – №3. – pp. 213-216/

36. Webb, G.I. Machine Learning for User Modeling / G.I. Webb // User Modeling and User-Adapted Interaction. – 2001. – №11. – pp. 19-29.

37. Zhang, D. Machine Learning and Software Engineering / Du Zhang, Jeffrey J.P. Tsai // Software Quality Journal. – 2003. – №11. – pp. 87-119.

38. Nayak, P.K. Comparison of modified teaching–learning-based optimization and extreme learning machine for classification of multiple power signal disturbances / P. K. Nayak, S. Mishra, P. K. Dash, Ranjeeta Bisoi // Neural Computing and Applications. – 2016. – №27. – pp. 2107-2122.

39. Шипилов, С.С. Применение алгоритмak-means, нейронной сети Хэмминга и метода Виолы-Джонса для распознавания текста / С.С. Шипилов [и др.] // «Научное сообщество студентов XXI столетия. Технические науки»: сборник статей по материалам XXXI студенческой международной научно-практической конференции. – Новосибирск: Изд. «СибАК». – 2015. - №5 (31) – с. 46-50.

40. Шипилов, С.С.Практическое применение алгоритмов машинного зрения к вопросу обнаружения автомобильных номерных знаков / С.С. Шипилов [и др.] // «Научное сообщество студентов XXI столетия.

Технические науки»: сборник статей по материалам XXXI студенческой международной научно-практической конференции.– Новосибирск: Изд. «СибАК». – 2015. - №5 (31) – с. 40-46.

41. Шипилов, С.С. Практическое применение алгоритмов искусственного интеллекта в приложении распознавания номерных знаков / С.С. Шипилов [и др.] // «Актуальные вопросы математического моделирования и разработки программного обеспечения»: сборник статей по материалам Всероссийской молодежной научно-практической конференции. – Тольятти: Изд. ТГУ. –2015. – с. 14-16.