**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИБЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ**

**Кафедра веб-технологий и компьютерного моделирования**

ШАРАБАЙКО

Александр Егорович

**КЛАСТЕРИЗАЦИЯ ДАННЫХ**

Дипломная работа

Научный руководитель:

кандидат физико-математических наук,

доцент

И. М. Галкин

Допущен к защите

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_201\_г.

Зав. Кафедрой веб-технологий

и компьтерного моделирования

кандидат физико-математических наук

доцент Романчик В. С.

Минск, 2015

оглавление

Реферат 4

Введение 7

Глава 1. Представление данных 12

1.1. Типы данных и система счисления 12

1.1.1. Векторная матрица 12

1.1.2. Матрица подобия 13

1.1.3. Типы данных и системы отсчёта 14

1.2. Показатели близости (расстояния) 16

1.2.1. Относительные типы 17

1.2.2. Номинальные типы 19

1.2.3. Потерянные данные 20

1.3. Нормализация 22

1.4. Линейное проецирование 24

1.5. Нелинейное проецирование 24

1.6. Внутренняя размерность 25

1.7. Много-размерное масштабирование 27

Глава 2. Методы и алгоритмы кластеризации 28

2.1. Общая информация 28

2.2. Иерархическая кластеризация 30

2.2.1. Single-link и Complete-link алгоритмы из теории графов 32

2.2.2. Дендограмма и восстановленная структура 39

2.2.3. Иерархическая структура и ультраметричность 42

2.2.4. Другие алгоритмы теории графов для single-link и complete-link 44

2.2.5. Алгоритмы преобразованной матрицы для single-link и complete-link 46

2.3. Частичная кластеризация 47

2.3.1. Критерий кластеризации квадратичной ошибок 51

2.3.2. Метода кластеризации квадратичной ошибки 56

2.3.3. Метод ближайшего соседа кластеризации 62

2.3.4. Нечёткая кластеризация 64

2.3.5. Алгоритмы семейства FOREL 67

Глава 3. Приложение, реализующее алгоритмы кластеризации 71

3.1. Краткое описание приложения 71

3.2. Основные технологии, используемые в приложении 72

3.2.1. HTML 72

3.2.2. CSS 73

3.2.3. JavaScript 75

3.3. Архитектура приложения 77

Глава 4. Описание и анализ результатов работы программы 82

4.1. Алгоритмы, реализованные в приложении 82

4.1.1. Иерархический алгоритм кластеризации 82

4.1.2. Алгоритм кластеризации k-means 83

4.1.3. Алгоритм семейства FOREL 84

4.2. Наблюдения за работой алгоритмов 85

4.2.1. Наблюдения за скоростью работы алгоритмов 85

4.2.2. Визуальное наблюдение за работой алгоритмов 89

Заключение 95

Литература 96

# Реферат

Дипломная работа, 96., 10 источников, 42 рисунка.

**Ключевые слова:** кластерный анализ, классификация, алгоритмы кластеризации, представление данных, структура данных, анализ работы алгоритмов.

**Цель работы -** выявить основные преимущества и проблемы кластеризации, факторы, влияющие на её работу, для каких задач наиболее приемлемы методы кластерного анализа.

**Методы исследования –** структурирование и упорядочивание теоретических основ кластеризации данных, анализ работы алгоритмов посредством веб-приложения.

**Результатом** является структурированная информация по основам кластеризации данных, а также результаты практических наблюдений за работой алгоритмов кластеризации и их анализ.

**РЭФЕРАТ**

Дыпломная праца, 96., 10 крыніц, 42 малюнка.

**Ключавыя словы:** кластарны аналіз, класіфікацыя, алгарытмы кластарызацыі, прадстаўленне даных, структура дадзеных, аналіз працы алгарытмаў.

**Мэта працы** - выявіць асноўныя перавагі і праблемы кластарызацыі, фактары, якія ўплываюць на яе працу, для якіх задач найбольш прымальныя метады кластэрнага аналізу.

**Метады даследавання** - структураванне і парадкаванне тэарэтычных асноў кластарызацыі дадзеных, аналіз працы алгарытмаў з дапамогай вэб-прыкладанні.

**Вынікам** з'яўляецца структураваная інфармацыя па асновах кластарызацыі дадзеных, а таксама вынікі практычных назіранняў за працай алгарытмаў кластарызацыя і іх аналіз.

**ABSTRACT**

The diploma, 96., 10 sources, 42 drawings.

**Keywords:** cluster analysis, classification, clustering algorithms, data representation, data structure analysis of algorithms.

**Purpose** - to identify the main benefits and challenges of clustering factors affecting her work, for which tasks the most appropriate method of cluster analysis.

**Methods of research** - structuring and arranging the theoretical foundations of data clustering analysis of algorithms through a web application.

**The result** is a structured information on the fundamentals of data clustering and the results of practical observations of the work of clustering algorithms and their analysis.

# Введение

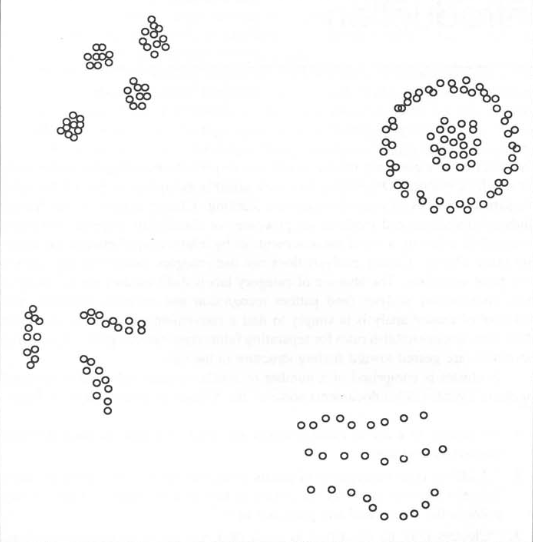
Практика классификации данных в зависимости от их сходства используется во многих областях науки. Организация данных в осмысленные группы – один из наиболее фундаментальных способов понимания и обучения. Кластерный анализ это изучение алгоритмов и методов группировки, классификации данных. Каждый объект данных описывается некоторыми мерами или отношениями между этим и остальными объектами. Кластерный анализ не имеет изначального набора классов или категорий, и отсутствие этого отличает кластерный анализ от классификации. Задачей анализа является не установка правил для разделения исходной информации на категории, а нахождение удобной и валидной организации данных. Кластерный анализ близок к нахождению структуры данных.

Кластер включает в себя некоторое число схожих объектов, собранных или сгруппированных вместе. Ниже представлены некоторые определения кластера:

1. Кластер – это множество схожих сущностей. Сущности из разных кластеров не подобны.
2. Кластер – это такое скопление точек в некотором пространстве, что дистанция между любыми двумя точками из этого кластера меньше, чем дистанция между любой точкой из кластера и любой точкой не из него.
3. Кластеры могут быть описаны как, соединённая область в многомерном пространстве, имеющая относительно большую плотность точек и разделённая от таких же областей областями с низкой плотностью точек.

В двух последних определениях предполагается, что объекты кластеризации представляются в виде точек мерного пространства. Мы различаем кластер, если мы видим его на плоской поверхности, но мы не понимаем, как мы это делаем. Легко дать функциональное определение кластера, но трудно дать оперативное представление. Это связано с тем, что объекты могут быть сгруппированы по-разному, в зависимости от целей анализа. Данные могут выявить различные «формы» и «размеры». Ещё всё более осложняет то, что принадлежность к какому-то кластеру у точки может меняться с течением времени, а число кластеров зависит от того, как и каким способом решено группировать данные.

Как видно на рисунке ниже, очень сложно определить конкретное число кластеров, и это сильно зависит от правил группировки, различных мер, расстояния и т. д. Таким образом, ключевая проблема кластеризации заключается в определении близости объектов и в том, каким образом это измерить.



Как видно, часто даже визуально трудно определить, что такое кластер

Техника кластеризации несёт в себе несколько преимуществ по сравнению с ручной группировкой.

Первое, программа кластеризации может последовательно применять указанный критерий для формирования групп. Человек превосходно находит кластеры в двух- и трёхмерном пространстве, но разные люди не всегда сгруппируют данные одинаково. Мера близости определённая между двумя объектами, зависит от образования, культуры и других особенностей человека, таким образом, одни и те же данные могут быть разделены людьми по-разному, особенно если эти данные не очень хорошо разделены.

Второе, алгоритм кластеризации формирует группы за меньшее время, чем группировка вручную, особенно если каждый объект имеет большой список особенностей. Скорость, надёжность и последовательность алгоритма кластеризации вместе являются большим преимуществом алгоритма. Кластеризация помогает учёным и аналитикам данных в коварной работе по отслеживанию векторной матрицы или матрицы подобия в процессе группировки данных. Время аналитиков данных лучше потратить на анализ и толкования результатов кластеризации, чем на саму кластеризацию.

Кластеризация также полезна в проведении стратегии «разделяй и властвуй» для уменьшения вычислительной сложности в алгоритмах шаблонного распознавания. К примеру, правило выбора ближайшего соседа является популярной техникой в шаблонном распознавании, однако нахождение ближайшего соседа может занимать очень много времени, если количество шаблонов и прототипов большое. Алгоритм Fukunaga and Narenda использует очень известный алгоритм частичной кластеризации ISODATA.

Рассмотрим группировку различных колледжей и университетов США для иллюстрации факторов проблем кластеризации. Школы могут быть сгруппированы в зависимости от географического положения, возраста учеников, размеров кампусов, оплаты за обучения или от предлагаемых программ обучения. Факторы зависят от целей анализа. Формы и размеры сформированных кластеров зависят от того, какой конкретно атрибут используется для определения близости между учебными заведениями. Сложность и интересность кластеризации возрастает, если несколько атрибутов используются вместе для группировки. Один кластер может представлять из себя множество частных, среднезападных начальных художественных школ с количество учеников, меньшим, чем 1000, а другой представляет большие государственные университеты. Атрибуты или характеристики здесь могут быть легко измерены. Проблема также существует в некоторых атрибутах, таких как качество образования, качество факультетов или качество школьной жизни, которые не могут быть легко измерены. Для каждой характеристики можно задать некоторое количество очков (к примеру, от 1 до 10) или меры схожести для каждой пары учебных заведений. Эти очки или меры должны быть приведены к среднему, так как индивидуальные показатели могут различаться.

Пример выше иллюстрирует различие между классификацией и кластеризацией. Предположим, что мы хотим разбить программы обучения программированию в США на две группы, в зависимости от таких атрибутов, как размер факультета, оснащённость компьютерами, взаимодействие с внешним миром и публикации факультета. В классификации «эксперт» должен в первую очередь определить категории, в одну из которых каждый объект данных будет попадать. Атрибуты используются для определения границы или порога, по которому данные будут относиться к одному из двух классов. В парадигме кластеризации эксперт не определяет категории. Цель кластеризации в данном случае – определить, каким образом данные будут разбиваться на две категории, в зависимости от характеристик, после чего для каждого объекта определяется, к какому кластеру он относится. Это может быть достигнуто формированием подобиями между всеми парами программ обучения программированию, основываясь на заданных атрибутах и конструировании кластеров, основываясь на том, что кластер больше подобен внутри, чем снаружи.

Кластерный анализ – один из компонентов исследовательского анализа данных, который обрабатывает данные вне зависимости от каких-либо начальных категорий, а только в зависимости от самих данных. Информация, полученная о данных в результате кластерного анализа, может привести к новым экспериментам, приобретению нового взгляда на сущности объектов или к созданию новых подходов, методов. Современные цифровые компьютеры делают всё это возможным.

Кластерный анализ – это дитя компьютерной революции. Он освобождает аналитиков от проверенных временем статистических методов, моделей и процедур, используемых, когда человеческий мозг был вооружён только карандашом и бумагой.

Разработка методологий кластеризации является поистине межотраслевой. Исследователи практически в каждой области науки, в которых собираются данные, поучаствовали в этом. Этими областями, к примеру, являются таксономия, психология, биология, статистика, социальные науки и инженерия.

Многие учёные и аналитики выпустили книги по кластерному анализу, среди них Андерберг (1973), Еверитт (1974). Трион и Бэйли были одними из первых, кто написал подобную книгу (1970), Джардин и Сибсон (1971) сконцентрировались на математических основах кластеризации, Лорр (1983) написал книгу по кластеризации специально для социологов и многие другие, Ван Ризин привёл в своей книге много интереснейших примеров (1977). Эншлен написал про вычислительные особенности кластеризации.

Задача данного дипломного проекта – дать теоретическое описание и анализ основных аспектов, методов и подходов кластеризации, классифицировать основные типы данных, алгоритмы, способы представления и методы оптимизации кластеризации. Также задачей этой работы является оценка работы наиболее распространённых алгоритмов кластеризации на основе их практической реализации.

Цель проекта – выявить основные преимущества и проблемы кластеризации, факторы, влияющие на её работу, для каких задач наиболее приемлемы методы кластерного анализа.

В первой главе этой дипломной работы будет дано широкое освещение и описание основ кластерного анализа, во второй части будет дано описание приложения, реализующего наиболее распространённые алгоритмы кластеризации, в третьей части будет дана оценка и проведён анализ работы этих алгоритмов на практике.

# Представление данных

Первой предпосылкой качественного приложения кластеризации данных является высокий учёт базовых факторов представления данных. Алгоритмы кластеризации соответствуют определённым типам данных и некоторым необязательным факторам, как масштаб, нормализация и тип меры приближения.

Кластерный анализ является инструментом для изучения данных, и он должен быть поддерживаем техниками и методами визуализации данных. Наиболее общий способ визуализации – объекты кластеризации, представленные в виде точек на двумерной плоскости. Конечно, многомерные данные не всегда можно представить двумерно, но иногда, если это валидно, представление результатов алгоритма в таком виде помогает проверить его правильность.

## Типы данных и система счисления

Группы объектов алгоритмов кластеризации или элементы данных основываются на признаках близости между парами объектов. Сами объекты называются по-разному: объекты, случаи, операционные таксономические единицы, в зависимости от приложения. Множество необработанных объектов данных для кластеризации могут быть описаны двумя стандартными способами: векторной матрицей и матрицей схожести.

### Векторная матрица

Часто каждый объект в множестве из n элементов представляется в виде множества из d характеристик (атрибутов или очков). Каждый объект тогда может иметь вид d-мерного вектора, а само множество представляется в виде векторной матрицы. Каждая строчка матрицы соответствует объекту (или его вектору), а колонка отвечает за определённую характеристику.

К примеру, если есть функции, зависящие от времени, такие как биологические сигналы или эхо радара, то характеристикой может быть простое значение функции в определённый момент времени, среднее от значений этой функции также может быть характеристикой. Множество этих значений функции – это вектор или объект. Следует также добавить требование, что одни и те же характеристики (значения) возникают в одно и то же время.

Если применить алгоритмы кластеризации к пациентам в больнице, то каждая строка отвечает за определённого пациента. Характеристиками или колонками матрицы могут быть результаты диагностики или плата за лечение. Как и в предыдущем примере, здесь есть требование того, что для каждого пациента точно определены все характеристики. Категориальные (или внешние) характеристики, такие как пол, религия, цвет волос, могут быть использованы в представлении результатов, но они не могут входить в векторную матрицу.

Размерность вектора d обычно изображается как множество ортогональных осей. N векторов, которыми представлены исходные n точек в d-мерном пространстве, называются векторным пространством (вектором в техническом смысле обычно обозначается точка векторного пространства). Кластер можно визуализировать как набор векторов, замкнутых относительно других кластеров и/или которые удовлетворяют некоторым внутренним отношениям. Задача кластеризации – определение некоторой натуральной группировки объектов в многомерном пространстве.

Учитывая, что визуальное представление ограничено тремя размерностями, нужно быть настороже, думая о проблемах кластеризации как о двух- и трёх-размерных. Наибольшая выгода от кластеризации есть в многомерных пространствах, где это очень трудно или невозможно визуализировать.

### Матрица подобия

Методы кластеризации требуют определённый показатель подобия (близости или ассоциативности), который должен быть установлен между всеми парами объектов. Этот индекс может быть вычислен из векторной матрицы или получен непосредственно из характеристик векторов. Данные в некоторых психометрических приложениях изначально собраны как некоторые близости. Например, несколько человек могут ответить, насколько хорошо они относятся к определённым брендам мыла, и близость между двумя брендами вычисляется усреднением результатов всех опрошенных. Люди также могут напрямую создать подобия между брендами, к примеру, оценив каждый по шкале от 1 до 10.

Таким образом, матрица подобия собирает попарно индексы близости в матрице, в которой каждая строка и каждый столбец представляет определённый вектор. В матрице игнорируются диагональные ячейки, так как считается, что все векторы имеют одинаковые близости между самими собой. Также считается, что все матрицы подобия симметричны, то есть все пары объектов имеют одинаковый индекс подобия, независимо от того, в каком порядке они взяты (хотя некоторые исследователи, такие как Габерт (1973) и Говер (1977) рассматривали несимметричные матрицы подобия).

Функция подобия обычно есть одно из двух: схожесть или различие. Какой-то i-ый объект подходит (или близок, или подобен) некоторому j-ому объекту, если у них есть наибольший индекс схожести или наименьший индекс различия. К примеру, Евклидово расстояние между двумя точками в векторном пространстве, есть показатель различия, тогда как коэффициент корреляции – коэффициент подобия. Можно заметить, что векторная матрица может быть легко превращена в матрицу подобия, а с помощью алгоритмов проецирования или техник «многомерного масштабирования» можно выполнить обратное преобразование.

Ниже представлена матрица подобия (Рисунок 1.2), которая использовалась Левиным (Levine 1977) для исследования чувственного подобия цифр (0-9), которые были представлены в виде - точечной матрицы и показывались разное количество раз на мониторе. Для исследования пригласили студентов, которые видели только 50 из 63 точек (раздражителей) и по ним пытались угадать число. В итоге, таблица показывает совокупность всех правильных и неправильных ответов для всех возможных комбинаций исходных цифр и ответов, данных студентами. К примеру, во второй строке, показано, что цифра 1 была показана 400 раз, из них 269 раз эта цифра была угадана. Левин определил частоту неправильных ответов между определёнными числами, как показатель подобия. Заметим, что в этом случае этот показатель – показатель схожести, так как, чем больше этот индекс, тем более подобны числа. Заметим также, что эта матрица несимметрична.

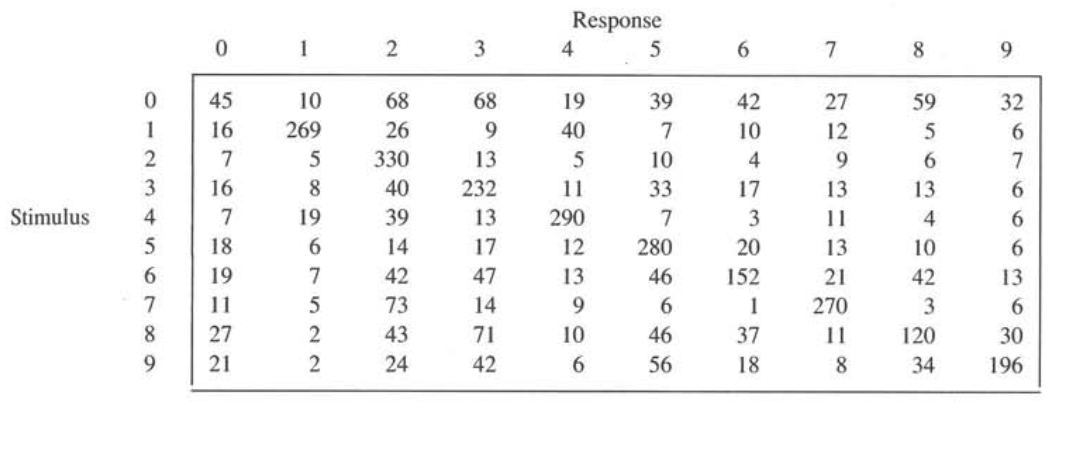
**

Рисунок .. Результаты наблюдений Левина

### Типы данных и системы отсчёта

Тип данных зависит от степени дискретизации исходных данных. Отдельный объект данных может быть представлен как бинарный, дискретный и непрерывный. Бинарные данные имеют точно два значения, к примеру, true или false. Дискретные объекты имеют конечное, обычно небольшое множество возможных значений. Например, образцы звуковых сигналов могут быть оцифрованы до 16 (или ) уровней, таким образом, характеристики звукового образца могут быть закодированы в 4 бита. Все меры и числа, хранящиеся в компьютере, имеют конечное число значимых чисел, таким образом, строго говоря, все компьютерные данные дискретны. Хотя на самом деле, часто очень удобно думать о величине некоторой характеристики, как о точке в реальном пространстве, где оно может принимать любое значение из поля действительных чисел. Такие характеристики называются непрерывными.

Показатели близости также могут быть бинарными, дискретными и непрерывными. Предположим, что множество разделено на взаимно-непересекающиеся множества. Показатель подобия равен нулю для объектов из разных подмножеств и единице для объектов их одного множества. Это бинарный индекс. Если для пары из разных множеств показатель равен 0, а для одинаковых (где n – количество объектов в подмножестве (если n = 1, то порядок равен 1)), то это дискретный индекс. Евклидово расстояние является примеров непрерывного показателя.

Следующей особенностью характеристик и показателей подобия является система отсчёта. Признаки данных можно разделить на качественные (номинальные и порядковые) и количественные (интервал и коэффициент). Номинальные признаки обычно не рассматриваются как признаки, так как они просто используются для именования объектов (например, 0 и 1, true или false), и не несут в себе никакого количественного смысла. Остальные качественные особенности и наислабейшие количественные, называются порядковыми показателями, и они только показывают некоторое отношение между данными. Например, шкалы (1, 2, 3), (10, 20, 30) и (1, 200, 300) для порядкового поля эквивалентны. Бинарные и дискретные характеристики и показатели подобия могут быть закодированы в эти качественные шкалы.

Разделение между числами имеет смысл в интервальной шкале. В ней существует единица меры, и интерпретация чисел зависит от этой единицы. К примеру, человека можно попросить оценить удовлетворённость некоторыми политиками по шкале от 0 до 100. Пары (45, 55) и (10, 90) могут идентифицировать очень разные стороны вопроса, но перед этим необходимо узнать, в какой шкале была использована цифра 10 (от 0 до 100, от 1 до 10 или от 10 до 90). Можно также упомянуть, что 90 градусов тепла по Фаренгейту и 90 по Цельсию, в общем, разные величины.

Наиболее строгая шкала – это относительная шкала, в которой каждому числу соответствует точное значение. Она также включает в себя ноль. Шкала Фаренгейта, например, является относительной шкалой.

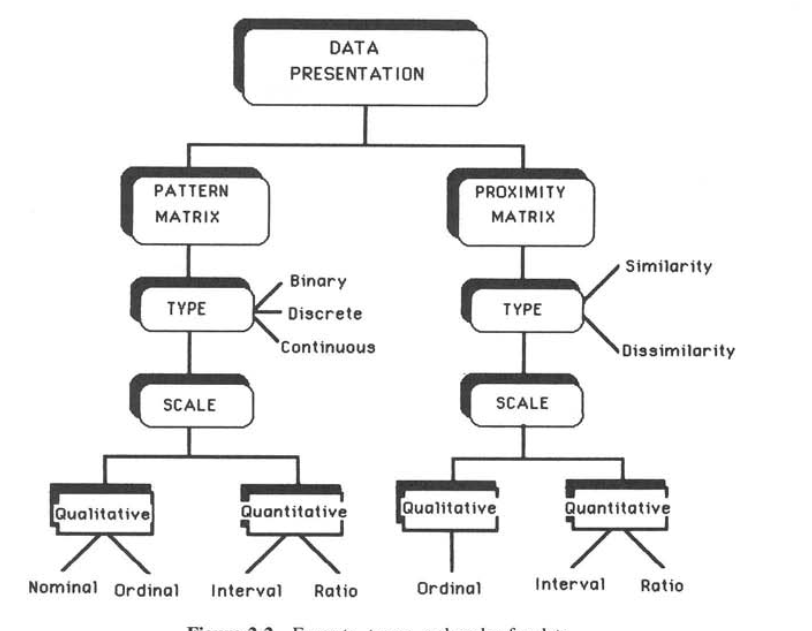


Рисунок 1.3. Структура данных

Тип данных и шкалу не всегда можно выбрать однозначно. Установка типа и масштаба важно как в определении близости объектов, так и в интерпретации результатов алгоритма.

## Показатели близости (расстояния)

Далее, говоря расстояние, будем подразумевать показатель различия между объектами. Андерберг в 1973 году опубликовал подробные исследования мер близости объектов и их взаимосвязей. Показатель близости между объектами i и k обозначается (чаще под этим подразумевается расстояние) и удовлетворяет следующим тремя условиям:

* 1. Для различия:
  2. Для схожести:

### Относительные типы

Показатель подобия может быть определён несколькими способами. Предположим, что мы имеем векторную матрицу , где определяет j-ю характеристику i-го вектора. Все характеристики непрерывны и измеримы в относительной шкале. Наиболее общий показатель подобия для таких векторов – метрика Минковского, которая является мерой различия. I-ый вектор, который представляется в матрице в виде i-й строки, обозначается

где d – количество характеристик, n – число объектов данных, T обозначает транспонированный вектор.

Метрика Минковского определяется следующим способом:

где .

Метрика Минковского удовлетворяет всем признакам показателя подобия (расстояния), и ещё двум, представленным ниже:

* только если
* (неравенство треугольника)

Учёные Говер и Легендре в 1986 году показали, что для расстояния (или показателя расхождения) необходимы только условие 1 и неравенство треугольника, остальные выводятся из этих двух.

Ниже показаны наиболее популярные и общие метрики Минковского (Рисунок 1.4):

1. (расстояние Евклида):
2. (Манхэттенское расстояние, расстояние такси или расстояние городских кварталов):
3. (расстояние максимального):

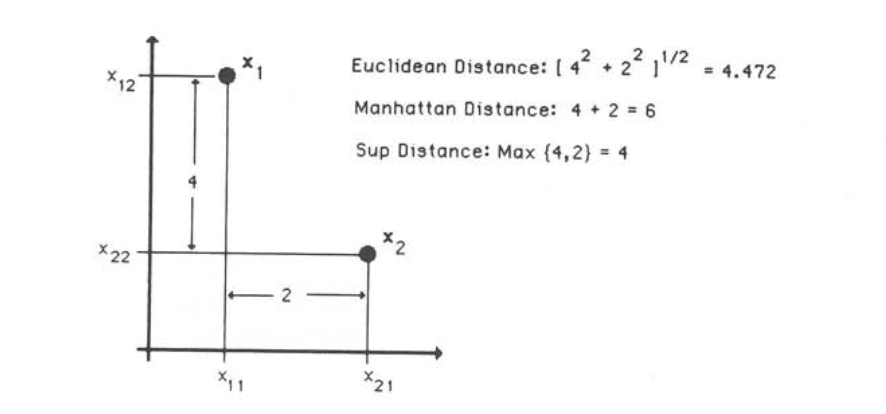


Рисунок 1.4. Разница в значениях у разных расстояниях

Расстояние Евклида наиболее общее среди всех метрик Минковского. Самые распространённые подходы и методы трансформаций, преобразований, разворотов векторов в пространстве валидны только для расстояния Евклида. Принятой практикой в приложениях кластеризации строго придерживаться выбранной метрике. Евклидово расстояние также более предпочтительно в инженерных работах.

Если все атрибуты объектов бинарные, то Манхеттенское расстояние называется расстоянием Гамильтона или количеством отличных друг от друга характеристик.

Не все показатели подобия используются в качестве метрик в приложениях. Более того, Тверский в 1977 году показал несколько примеров того, что не все показатели схожести симметричны и транзитивны.

Квадратное расстояние Махаланобиса также используется как метрика в приложениях. Это расстояние определяется следующим способом:

где – некая ковариантная матрица. Если эта матрица – единичная матрица, то квадратное расстояние Махаланобиса соответствует расстоянию Евклида.

### Номинальные типы

Если непрерывные данные с относительной шкалой являются наиболее “сильными”, то бинарные данные с номинальной шкалой – самые “слабые”. Многие множества, в особенности некоторые данные о группах людей, являются бинарными данными с номинальной шкалой. Например, информация о распределении учеников по классам, по месту жительства, по успеваемости. Для простоты, каждая характеристика принимает значения 0 или 1, что означает истину или ложь, «большой» или «маленький», «есть» или «отсутствует». Показатель подобия выводится из таблицы, представленной ниже, где обозначает количество характеристик, которые есть (принимает значение 1) у обоих векторов и , – количество аттрибутов, отсутствующих у двух векторов, - 0 у и 1 у , - 0 у и 1 у .

|  |  |  |
| --- | --- | --- |
| \ | 1 | 0 |
| 1 |  |  |
| 0 |  |  |

Таблица 1.1. Шаблон матрицы бинарного подобия

Многие исследователи, такие как Андерберг, Говер, Холл работали над определением наиболее подходящего показателя подобия для данных такого рода. В результате наиболее общие, популярные коэффициенты близости представлены ниже:

1. Простой коэффициент сходства:
2. Коэффициент Джакарта:

Какой из этих коэффициентов применять, зависит от исследователя и целей исследования.

Например, допустим, что есть результаты исследования, в котором респонденты отвечали на вопросы теста, выбирая один из двух ответов «да» или «нет». Результаты можно выразить следующей таблицей:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|  | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
|  | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |

Таблица .. Результаты наблюдений

После чего получаем таблицу совместимости:

|  |  |  |
| --- | --- | --- |
| \ | 1 | 0 |
| 1 | 8 | 1 |
| 0 | 4 | 7 |

Таблица 1.3. Полученная матрица подобия

После чего вычисляем коэффициент сходства: 26/36 = 0.722, и коэффициент Джакарта: 4/14 = 0.286.

Если эти коэффициенты равны единице, то это означает, что векторы идентичны.

### Потерянные данные

Проблема потерянных данных очень часто встречается в практических приложениях. Предположим, что некоторый вектор (объект) имеет потерянные значения некоторых характеристик, как здесь:

где третья и пятая характеристика были утеряны или стёрты, что случается из-за вычислительной ошибки, поломкой оборудования или другими факторами. Вопрос состоит в том, что делать с незаполненными объектами. Ответы на эти вопросы искали в своих исследованиях такие учёные, как Снетт, Сокал, Киттлер, Диксон, Ёлкина.

В 1979 году Диксон описал несколько простых, незатратных, лёгких для имплементации и общих техник для обработки потерянных данных, некоторые из которых представлены ниже:

1. Простое удаление векторов с потерянными данными. Это не всегда правильное решение и оно должно быть использовано только тогда, когда число потерянных данных очень небольшое.
2. Предположим, что утеряна j-я характеристика k-го вектора. Далее берём K ближайших соседей данного вектора, находим среднее значение j-й характеристики у этих объектов, и вставляем это на место . Значение K зависит от размера векторной матрицы.
3. Дистанция между двумя векторами и вычисляется следующим способом. Сначала определим дистанцию между двумя векторами вдоль аттрибута j:

Затем расстояние между векторами и запишем в виде:

где – число характеристик, потерянных в или в обоих. Заметим, что если потерянных данных нет, это расстояние эквивалентно расстоянию Евклида.

1. Обозначим среднее расстояние между всеми парами векторов вдоль j-й характеристики, которая определяется следующим образом:

где n – количество векторов. Теперь определим расстояние между двумя векторами вдоль j-й характеристики следующим образом:

И наконец, определим расстояние между объектами:

Основываясь на результатах экспериментов, Диксон рекомендовал метод 3 как наиболее общий и полный.

## Нормализация

Предположим, что изначальные данные представляются в виде векторной матрицы, в которой все характеристики непрерывны и представлены в относительной шкале. Эти данные или фактические измерения редко используются в том виде, в котором они записаны. Обычно некоторая нормализация требуется анализом. Подготовка данных к кластерному анализу включает в себя некий вид нормализации, благодаря которому показатель подобности будет работать корректно. Общепринятое и популярное Евклидово расстояние неявно добавляет больше значимости характеристикам с большим диапазоном по сравнению с характеристиками с малыми диапазонами. Если взять одну величину в милях, а другую в метрах, то вторая численно очень сильно возрастает по сравнению с первой. Эти проблемы и решает нормализация.

Как было показано выше, каждому объекту данных можно поставить в соответствие вектор d, скалярные компоненты которого называются характеристиками (атрибутами). При этом i-й вектор обозначается , а его j-ая характеристика обозначается . Звёздочка обозначает «сырые» или ненормализованные данные. Таким образом, получаем ненормализованную векторную матрицу:

Среднее значение j-й характеристики и её отклонение определяются следующими способами:

Самый простейший приём нормализации данных приведён в формуле ниже:

Эта нормализация делает значения атрибутов неизменяемыми при жёстком перемещении координат. Следующая нормализация преобразовывает и масштабирует оси координат, и в результате все значения имеют нулевое среднее и единичное расхождение:

Удаление звёздочки обозначает, что вектор был нормализован, хотя тип нормализации обычно не зависит от контекста программы. Некоторые другие типы нормализации включают в себя диапазонное масштабирование (Кармикаел, 1968) или неоднородную меру (Холл, 1969). Люменский (1982) включил нормализацию непосредственно в процесс кластеризации.

Нормализация и масштабирование не всегда необходимы, более того, иногда нежелательны. К примеру, иногда нормализация нарушает разрозненность между кластерами и внутреннее расстояние между точками в группе, как показано на рисунке ниже (Рисунок 1.5):

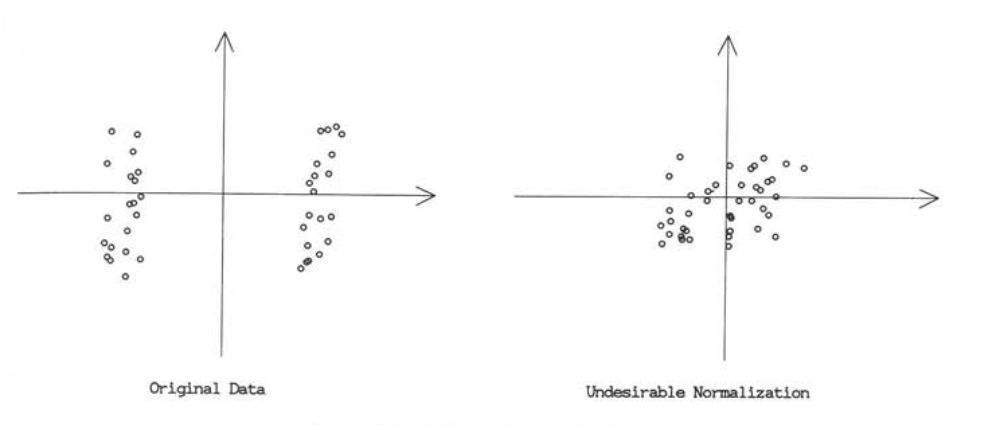


Рисунок 1.5. Пример неподходящей нормализации

Матрица размерностью , которая определяется в терминах нормализованных данных следующим образом:

где:

называется ковариационной матрицей, где - коэффициент корреляции между атрибутами i и j, и .

Каждый вектор может быть представлен как единица массы в некотором пространстве. Матрица А показывает данные как кучу точек в пространстве, где строка соответствует вектору. Примеры нормализаций, описанные выше, делают диагональные элементы матрицы R моментами инерции этих скоплений точек относительно осей координат.

## Линейное проецирование

Алгоритмы проецирования отражают (проецируют) множество из векторов размерности d на m-мерное пространство. Главное основание для использования алгоритмов проецирования в кластерном анализе – позволить визуализировать многомерные данные (таким образом, далее будем считать m = 2). Когда данные представлены в двумерном пространстве, их легко оценить зрительно и дать квалифицированную оценку результатам работы алгоритма. Эта проблема относится только к кластеризации в многомерном пространстве.

Линейная проекция выводит m новых характеристик в виде линейных комбинаций исходных d атрибутов.

где – m-мерный вектор, – d-мерный вектор и H – матрица размерностью . Алгоритмы линейной проекции относительно простые в использовании, сохраняют характер данных и имеют понятные математические свойства. Тип линейного проецирования, используемый на практике, зависит от доступности информации о векторах. Если такая информация доступна, обычно используется проекция собственного вектора (также называемая методом Кароена-Лоэва или методом основного вектора). Дискриминантный анализ обычно используется, если информация доступна.

## Нелинейное проецирование

Невозможность алгоритмов линейного проецирования сохранять структуру «сложных» данных, сделало методы нелинейного проецирования более популярными в последние годы. Под «сложной структурой данных» мы понимаем ситуации, где векторы лежат на кривой плоскости. Например, метод главного вектора не очень подходит для представления данных в двумерном пространстве, которые лежат на трёхмерной спирали. Большинство нелинейных проецирований основано на максимизации или минимизации функции большого числа переменных. Эта категория оптимизации зависима от данных и не имеет явной функции преобразования. Таким образом, изменения в числе векторов требуют больших перевычислений. Таким образом, алгоритмы нелинейного проецирования дороги в использовании, поэтому здесь могут использоваться некоторые методы уменьшения времени поиска (например, метод главного вектора может быть использован в начале работы нелинейных проецирований).

Алгоритмы нелинейного проецирования могут быть выбраны в зависимости от того, доступна ли начальная информация о векторах. Если информация о категориях известна или известна информация о списке категорий, то целью является поиск нелинейного проецирования, которая уменьшает размерность уже максимизированной отделимости между категориями. Если такая информация отсутствует, то нужно преобразовать векторы на пространство меньшей размерности, как можно более точно сохраняя структуру данных (для примера, целью может быть сохранность всех внутрикластерных дистанций).

## Внутренняя размерность

Внутренняя или топологическая размерность n векторов в d-размерном пространстве ссылается на минимизацию «свободных» параметров, необходимых для создания вектора. Внутренняя размерность фактически определяет, как d-размерный вектор может быть правильно описан в подпространстве, размерностью, меньшей d.

Прилагательное «внутренняя» подчёркивает, что ищутся такие свойства данных, которые не зависят от размерности пространства d. Например, d-размерные векторы вдоль некоторой гладкой кривой имеют внутреннюю размерность 1, независимо от величины d. Аналогично, d-мерные векторы вдоль плоскости или гладкой поверхности, имеют внутреннюю размерность 2. Понятие внутренней размерности в основном отличается от понятия линейной размерности подпространства включением в себя всех n объектов данных (векторов). Линейная размерность – это глобальное понятие, которое обычно ссылается на число независимых собственных значений ковариационной матрицы. Другой понятие размерности описывается через фрактальную размерность, которая очень популярна в компьютерной графике.

На Рисунке 1.6 (a) показаны 22 точки, расположенные на плоскости в трёхмерном Евклидовом пространстве. Внутренняя размерность этих точек равна 1, а линейная 2. На рисунке (b) внутренняя размерность – 1, а линейная – 3.

Внутренняя размерность является важной характеристикой данных, так как это может определить необходимое (подходящее) число характеристик для представления данных. Знание внутренней размерности может сделать задачу уменьшения размерности более полезной и правильной путём определения нужного количества размерностей для функций проецирования. Понятие внутренней размерности также может быть использовании в определении числа параметров, необходимых для описания сложных систем, когда сами системы доступны через ограниченное число ресурсов.

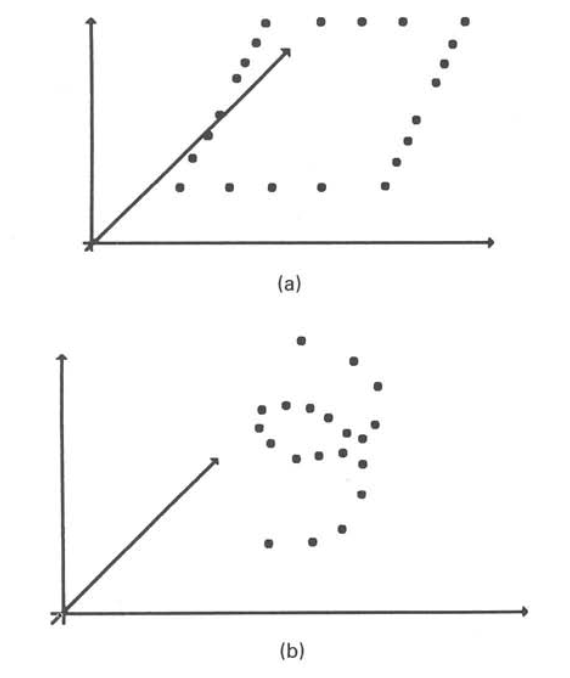


Рисунок 1.6. На первом рисунке внутренняя размерность равна 1, на втором - 2

Существует два общих подхода вычисления внутренней размерности через заданную векторную матрицу.

Первый подход динамический и он пытается «раскрыть» данные, или выровнять векторы в d-мерном пространстве. Внутренняя размерность может быть найдена через линейную размерность, которая вычисляется нахождением независимых собственных значений ковариантной матрицы «выровненных» данных. Этот метод общий и не всегда работает хорошо, особенно если «выровненные» данные сильно искажены.

Второй подход не двигает векторы и вычисляет внутреннюю размерность прямо из информации у локальных соседей этого вектора. Этот подход является статическим.

## Многомерное масштабирование

Многомерное масштабирование – общее понятие для ряда процедур и алгоритмов, которые начинают работать с простыми векторными матрицами и генерируют конфигурации точек в одной, двух или трёх размерностях. Цели и задачи многомерного масштабирования такие же, как и у итераций методов нелинейных проекций (хотя эти методы начинают работать с матрицами расстояний). Многомерное масштабирование преобразует обычную шкалу в множество относительных шкал и это есть пример ординации (статический метод, в котором данные из большого числа источников представлены в виде точек двух- или трёхмерной системы координат). Конфигурации точек предоставляют некоторую векторную матрицу, которую в дальнейшем можно преобразовать и интерпретировать.

Существует очень много разнообразной литературы по многомерному масштабированию. Также эти методы используются почти во всех разработках и приложениях в социальной и поведенческой психологии. Много великолепных исследований в области многомерного масштабирования было проведено Грином, Каменским, Крускалом, Карролл, Ромни. Шефардом, Торгерсоном, Кармоном.

Впервые это понятие ввёл в 1958 году исследователь Торгерсон. Шепард в 1962 году написал работу об анализе (обычных) приближений и Крускал в 1964 году на основе этого разработал алгоритм MDSCAL, который тогда был реализован на языке FORTRAN и который был популяриизирован как способ анализа данных и был установлен как важный инструмент социальной и поведенческой психологии.

Задачей методов многомерного масштабирования является создание множества шкал или размерностей, которые представляют данные. Эти методы тесно связаны с внутренней размерностью и алгоритмами нелинейной проекции.

# Методы и алгоритмы кластеризации

## Общая информация

Кластеризация – классификация, установленная на конечном числе объектов. Как было сказано в предыдущей главе, отношения между объектами данных могут быть представлены в виде матрицы подобия, в котором все строки и столбцы отвечают определённому объекту. Если объект описан как вектор или точка в d-мерном метрическом пространстве, приближение может выражаться через расстояние между ними, как, например, расстояние Евклида. До тех пор, пока не будет принята некая определённая мера дистанции, невозможно корректно осуществить кластерный анализ на данных. Матрица подобия наиболее правильная форма входных данных алгоритмов кластеризации.

Кластеризация – особая форма классификации. Си Кендел в 1966 году исследовал отношение между классификацией и кластеризацией, на основе чего исследователи Лэнс и Вильямс в 1967 году разработали дерево классификаций и их задач, изображённое ниже. Каждый лист этого дерева обозначает отдельный вид классификации, узел – определённую задачу (Рисунок 2.1). Узлы описаны ниже:

1. Исключительная и неисключительная. Исключительная (или однозначная) классификация относит каждый объект данных только к одной группе или кластеру, неисключительная (или параллельная) может относить объекты к разным классам одновременно. Примерам исключительной классификации может служить группировка учеников по возрасту или месту жительства, а если пациенты распределяются по болезням – то это пример неисключительной классификации. К исключительным классификациям относятся иерархический алгоритм кластеризации и алгоритм k-means, к неисключительным – c-means метод.
2. Внутренняя и внешняя. Внутренняя классификация использует только матрицу подобия для выполнения группировки. Внутренняя классификация относятся к алгоритмам «обучением без учителя» в методах распознавания, так как категории, на которые будут разбиваться данные, заранее не определены. Внешняя классификация заранее знает о категориях и по ним разбивает объекты. Задача таких алгоритмов состоит в нахождении такой дискриминантной поверхности, которая разделяет объекты соответственно их категориям. Другими словами, такие методы относятся к «обучению с учителем»

Один из способов оценки внутренней классификации – это посмотреть насколько кластерные категории, полученные в течение алгоритма, совпадают с заранее определёнными категориями. Для примера, предположим, что есть некоторое количество показателей персонального здоровья, собранных с курящих и некурящих. Внутренняя классификация может группировать пациентов, основываясь на показателях здоровья, и попытаться определить курящих на основе предрасположенности к определённым болезням. Внешняя классификация будет группировать людей, основываясь на заранее определённых характеристиках и показателях курящих людей. Внутренняя классификация – сущность кластеризации.

1. Иерархическая и частичная. Исключительная внутренняя классификация, в зависимости от структуры данных, делится на иерархическую и частичную. Иерархическая классификация – это очередь вложенных частей данных, тогда как частичная классификация определяет единственное разделение. Далее будем называть кластеризацией исключительные, внутренние, частичные классификации, а иерархической кластеризацией – исключительные, внутренние, иерархические классификации. Учёные Снет и Сокал в 1973 году приняли сокращение ПАИН (последовательные, агломеративные, иерархические, неповторяющиеся) для исключительных, внутренних, иерархических, агломеративных алгоритмов (англ. SANH (sequential, agglomerative, hierarchical, nonoverlapping)).

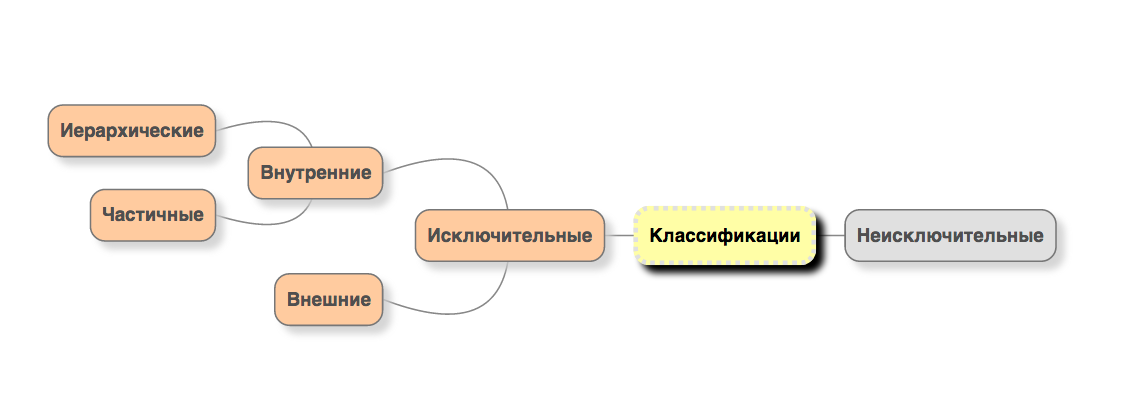


Рисунок 2.1. Структура видов классификации

Определённые алгоритмы могут быть использованы для проведения исключительной, внутренней классификации. Очень часто алгоритмы кластеризации реализованы различными вычислительными методами. Основные алгоритмические вариации представлены ниже:

1. Агломеративные и разделяющие. Агломеративная иерархическая классификация распределяет каждый объект данных в отдельный кластер и шаг за шагом объединяет их в большие кластеры, пока число групп не достигнет единицы. Разделяющая иерархическая классификация, наоборот, разделяет начальный кластер, включающий все элементы данных, и постепенно разделяет их на меньшие. Таким образом, эта опция соответствует выбору процедуры, а не вида классификации. Частичные классификации могут быть охарактеризованы таким же образом. Единственное разделение может быть установлено объединением воедино малых кластеров (агломеративная процедура) или фрагментацией единой группы, включающей все элементы (разделяющая).
2. Последовательные и параллельные. Последовательные методы обрабатывают векторы один за другим, а параллельные работают с ними одновременно.
3. Монотетичные и политетичные. Эта опция наиболее применима в проблемах таксономии, где элементы данных кластеризации представлены в виде векторов или точек пространства. Монотетичные алгоритмы используют характеристики последовательно, один за другим, тогда как политетичные используют все аттрибуты сразу.
4. Теория графов и алгебра матриц. Можно описать некотрые алгоритмы, используя термины теории графов, такие как объединённость и замкнутость для описания кластеров, а некоторые, используя термины алгебры, такие как среднеквадратичная ошибка. Выбор лежит на ясности, чёткости определения, а также от индивидуальных предпочтений. Когда алгоритм имплементируется на компьютер, это может также зависеть от того, что сделает алгоритм более производительным и менее затратным. В некоторых случаях одинаково удобно и правильно использовать методы алгебры и теории графов.

## Иерархическая кластеризация

Иерархическая кластеризация – это процедура трансформации матрицы подобия в последовательность вложенных разбиений. Для описания алгоритмов, нужно определить математическую структуру данных, на которую опираются иерархические методы.

Для начала определим последовательность вложенных разбиений. Данные, представленные в виде n объектов, обозначаются X:

где - i-объект.

Разбиение , которое разделяет X на подмножества , которые удовлетворяют следующим условиям:

где обозначает пересечение множеств, - объединение, - пустое множество. Здесь кластеризация является разбиением, а компоненты разбиения называются кластерами.

Разбиение вложено в разбиение , если каждая компонента является подмножеством некоторой компоненты , таким образом, сформировано объединением компонент . Для примера, пусть есть разбиение с пятью компонентами и разбиение с тремя компонентами, тогда вложено в , если они имеют вид, описанный ниже (оба разбиения действуют на множестве ):

Напротив, ни не вложена в , ни не вложена в , если или имеют вид:

Иерархическая кластеризация – это последовательность вложенных разбиений, в котором каждое разбиение вложено в следующее. Агломеративный алгоритм иерархической кластеризации начинает работать с несоединёнными кластерами, каждый из которых состоит их одного элемента данных. Алгоритм, используя матрицу подобия, вычисляет, какие два или более тривиальных кластеров должны быть соединены, таким образом вкладывает тривиальное разбиение в следующее. Этот процесс он повторяет, строя последовательность вложенных разбиений до тех пор, пока не останется один кластер, хранящий в себе все элементы данных. Разделяющий алгоритм выполняет эти действия в обратном порядке.

Изображение иерархической кластеризации более лёгкое для понимания людьми, чем набор абстрактных символов. Дендограмма – это специальный тип древовидной структуры, который удобно использовать для изображения иерархической кластеризации. Дендограмма содержит список начальных узлов, каждый из которых изначально является кластером. Линии, соединяющие кластеры, представляют их вложения.

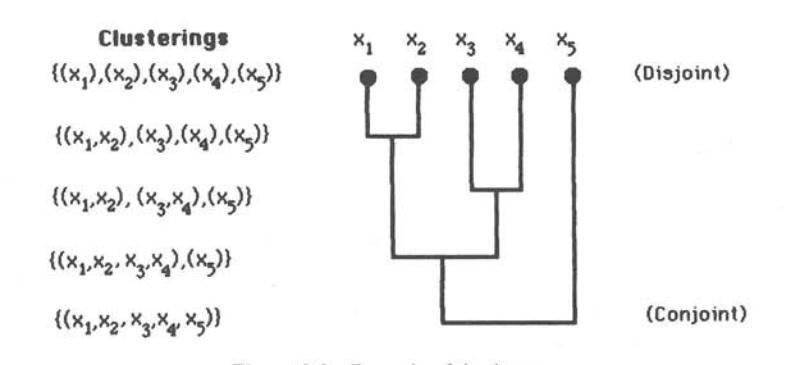


Рисунок 2.. Пример простейшей дендограммы

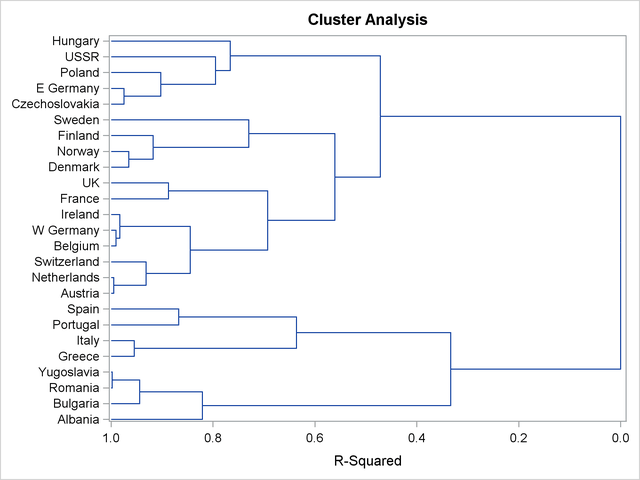


Рисунок 2.3. Пример дендограммы кластеризации стран

Наиболее популярными методами иерархической кластеризации являются методы single-link и complete-link.

### Single-link и Complete-link алгоритмы из теории графов

Пусть есть матрица подобия размерностью , которая была определена в предыдущей главе. Элементы матрицы по каждую сторону от главной диагонали - числа от 1 до , без каких-либо пустых ячеек. На показатель подобия берём расстояние, это значит, что если , то объект 1 более похож на объект 2, чем на 3.

**Пример**

Возьмём для примера матрицу подобия при :



Граничный граф – это ненаправленный, невзвешенный граф с n узлами, который не является циклом и не имеет составных рёбер. Каждый узел представляет объект данных. Граничный граф определён для каждого уровня различия включением ребра между узлами i и j, если показатель разности между ними меньше :

Как было сказано выше, . Таким образом, определяет бинарную связь для любого действительного числа v, что рефлексивно и симметрично. Бинарная связь – это подмножество произведения , где – множество объектов данных. Объекты и «связаны», если их различие меньше, чем порог v. Ниже показано, показано это отношение из матрицы с порогом v=5. Звёздочками показаны те позиции, объекты, относящиеся к которым «связаны».



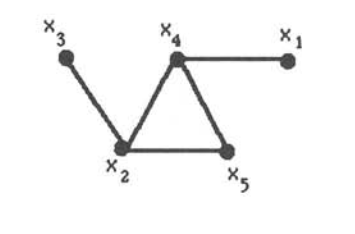


Рисунок 2.4. Граничный граф данного множества

Алгоритмы кластеризации single-link и complete-link основаны на граничном графе. Оба алгоритма используют только матрицу различия (матрицу расстояний) и создают вложенную последовательность кластеров, которую можно изобразить на дендограмме.

**Агломеративный алгоритм single-link**

1. Начинаем с несоединённого разбиения. Каждый объект данных входит в отдельный кластер. Граничный граф с порогом k = 0 () не содержит рёбер. Устанавливаем к = 1.
2. Если в граничном графе число компонент (максимально соеденённых подграфов) меньше, чем число кластеров в текущей кластеризации, переопределяем текущую кластеризацию, назначая каждую компоненту графа новым кластером.
3. Если граничный граф содержит один соединённый подграф, останавливаем алгоритм. Иначе устанавливаем k:=k+1 и идём в шаг 2.

**Агломеративный алгоритм complete-link**

1. Начинаем с несоединённого разбиения. Каждый объект данных входит в отдельный кластер. Граничный граф с порогом k = 0 () не содержит рёбер. Устанавливаем к = 1.
2. Если в граничном графе два кластера текущей кластеризации образуют клики (полный подграф), переопределить текущую кластеризацию объединением этих кластеров в один.
3. Если , это значит, что - полный граф на n элементах, тогда остановить работу алгоритма. Иначе, устанавливаем k:=k+1 и идём в шаг 2.

Эти алгоритмы могут быть продолжены для матриц расстояний на интервале или относительной шкалы. Если – граничный граф, содержащий рёбра, соответствующие расстояниям меньше k, граничная дендограмма отражает то, как и в каком порядке формируется каждая кластеризация. Дендограмма приближения показывает уровень, на котором каждая кластеризация формируется.

Алгоритмы кластеризации single-link, основанные на граничном графе , определены в терминах соединённых подграфов в ; методы complete-link используют полные подграфы.

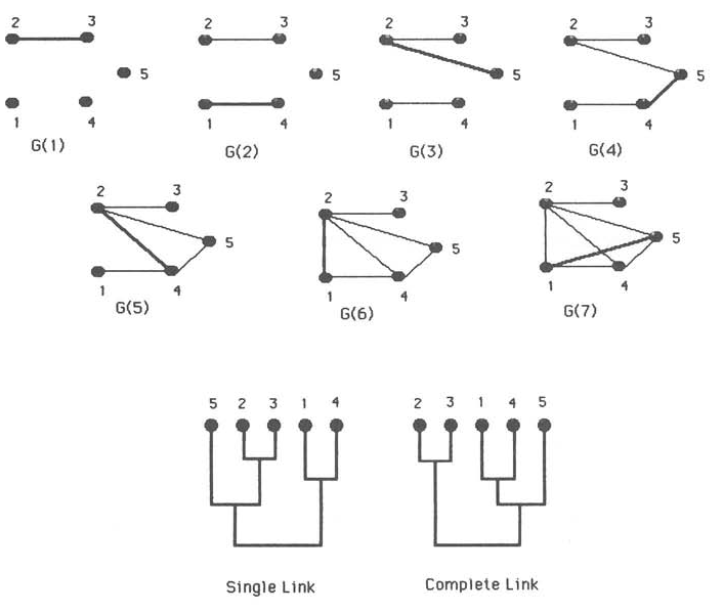


Рисунок 2.5. Процесс построения граничных графов для single-link и complete-link, а также их дендограммы

Из Рисунка 1.2 по граничным графам видно, как формировались кластеризации. Первые четыре графа необходимы для работы single-link алгоритмов кластеризации, а первые семь – для complete-link. Объекты сформировали клику в граничном графе , но не были объеденены в кластер, так как никакие из этих точек не формировали группу (кластер). К тому же, можно заметить, что не было дано чёткого ответа, к какому кластеру отнести объект .

Губерт в 1974 году предложил алгоритмы, генерирующие иерархические кластеризации используя методы single-link и complete-link. Если матрица подобия не содержит никаких пустых ячеек, кластеризации нумеруются и m-ая кластеризация содержит n-m кластеров.

**Алгоритмы Губерта для single-link и complete-link**

1. Устанавливаем m:=0. Это формирует несоединённую кластеризацию с номером m.
   1. Чтобы найти следующую кластеризацию (с номером кластеризации m+1) с помощью single-link метода, определяем функцию для всех пар кластеров в текущей кластеризации следующим образом:

где такое, что максимальный подграф граничного графа , определённый объединением кластеров и соединён.

Кластеры и соединяются для формирования следующего кластера в иерархии single-link, если:

* 1. Чтобы найти следующую кластеризацию (с номером кластеризации m+1) с помощью complete-link метода, определяем функцию для всех пар кластеров в текущей кластеризации следующим образом:

где такое, что максимальный подграф граничного графа , определённый объединением кластеров и полон.

Кластеры и соединяются для формирования следующего кластера в иерархии single-link, если:

1. Устанавливаем m:=m+1 и повторяем шаг 2. Повторяем до тех пор, пока объекты не окажутся в одном кластере.

Слово “максимальный” в определениях функций и обозначает, что все узлы из двух кластеров и должны быть рассмотрены при установке соеденённости и полноты. Только существующие кластеры могут быть соединены в следующий уровень.

**Пример**

Один из способов понимания функций и – рассмотреть последовательность граничных графов, хотя это не обязательно для определения этих функций. Например, пусть есть первые семь граничных графов для патрицы подобия , которые изображены на рисунке выше. Третья кластеризация (m=2) может быть пронумерована следующим образом:

Эти три кластера определены так:

Чтобы определить , когда m=2, найдём наименьшее расстояние, которое соединит два из существующих кластеров. Кластеры и становятся соеденёнными в граничном графе G(3). Таким образом, и . Другой способ понимания этой функции, это представить, что - это наименьшее расстояние, соеденяющее кластеры и и наименьшее из таких расстояний определяет следующую кластеризацию. В этом случае, кластеры и впервые соединяются на уровне 3, или в G(3), кластеры и соеденяются в G(4) и кластеры и - в G(5). Минимум из этих уровней (3, 4, 5) – 3.

Определение функции такое же, как и , только вместо соеденённости критерием выступает полнота. Например, определяется при m=2 перебором граничных графов в очереди до тех пор, пока некоторый граничный граф не соединит текущие кластеры (из ) в полный подграф. Этого не происходит до G(7), поэтому и . Кластеры и впервые образуют полный подграф в G(9), а и – в G(10). Минимум из этих уровней (7, 9, 10) – 7, поэтому четвёртая complete-link кластеризация получена из седьмого граничного графа соединением кластеров и . Факт того, что некоторые другие подграфы были полными ранее () не существенен.

Single-link кластеры характеризуются как максимально соединённые подграфы, в то время как complete-link кластеры являются кликами, или максимально полными подграфами. Исследователи Джардин и Сибсон в своих работах описали некоторые привлекательные теоретические свойства single-link кластеризации, но некоторые учёные описали трудности с кластерами, сформированными с помощью single-link методов. Например, такие кластеры легко соединяются с помощью единственного ребра, что делает их «всколоченными». С другой стороны, complete-link кластеры консервативны. Все пары объектов должны быть соотнесены, перед тем как такие группы будут сформированы. Полнота - более сильное свойство, чем соединённость. Однако, как отмечают многие исследователи, single-link кластеры имеют небольшую однородность, тогда как complete-link кластеры не так хорошо разделены.

Из сказанного выше следует, что эти кластеризации не всегда удовлетворяют условиям задачи, и существует много альтернатив им. Пия предложил исключительный, повторяющийся, иерархический метод кластеризации, основанный на кликах и использующий несимметричные матрицы подобия. Матула в 1977 году предположил, что число возможных клик слишком большое, таким образом, кластеризация на основе клик нужно использовать только для малых n.

Каждый соединённый подграф в граничном графе соответствует single-link кластеру, но не каждая клика в нём – complete-link кластер.

Предположим, что самая последняя кластеризация на множестве в некоторой иерархии, была сформирована объединением кластеров и в кластеризации:

Следующая характеристика может помочь различать два метода кластеризации. Если кластеризация была выполнена single-link методом, тогда:

Если кластеризация была выполнена complete-link методом, тогда:

Эти характеристики показывают, почему single-link методы называются «минимальными» методами, а complete-link – «максимальными». Хотя если за показатель подобия взять показатель схожести, а не различия, терминология меняется.

Эта характеристика показывает, почему complete-link также называют «диаметральным» методом. Диаметром полного подграфа называют наибольший показатель подобия среди всех пар подграфа. Также, complete-link метод не создаёт кластеров с минимальными диаметрами; диаметры таких кластеров известны, и они равны тем уровням, на которых эти кластеры сформировались.

И напротив, single-link кластеры основаны на соединённости и характеризуются минимальной длиной пути между всеми парами в кластере.

### Дендограмма и восстановленная структура

Важной целью иерархической кластеризации является изображение данных для лучшего понимания работы алгоритма (дендограмма – наиболее популярный способ изображения иерархической кластеризации). Дендограмма перечисляет кластеризации последовательно. Информация из любого уровня дендограммы определяет текущую кластеризацию и сами кластеры.

Граф подобности – это граничный граф, в котором каждое ребро имеет вес, соответствующее своему приближению. Приближения, использованные в предыдущей секции, порядковые, поэтому веса рёбер в графе подобности, построенном относительной тех данных, - целые числа, от 1 до n(n-1)/2. Дендограмма, построенная из графа подобности, называется дендограммой подобия, и она отмечает в себе как кластеризации, так и показатели приближения, по которым они были сформированы. Дендограммы подобия особенно используются, когда подобие определено на интервале или относительной шкале.

**Пример**

Пусть дана матрица подобия :



Дендограмма подобия нарисована на шкале показателей из последовательности графов подобия и выделяет кластеры, которые «появились» рано и «продолжаются» на дендограмме долгое время.

Любой иерархический алгоритм кластеризации может быть рассмотрен как способ преобразования матрицы в дендограмму. Граничные дендограммы и дендограммы подобия представляют структуру, которую иерархические алгоритмы строят на исходных данных. Это строит структуру, которая может быть охвачена другим видом матрицы подобия, называемой кофенетической матрицей. Такая матрица определена для того, чтобы лучше понять разницу между single-link и complete-link методами.

Начнём с иерархической кластеризации:

в которой m-ая кластеризации содержит в себе n-m кластеров:

Уровневая функция L следит за показателем подобия, на котором формируется каждая кластеризация. Для граничной дендограммы , так как уровни в дендограмме равномерно расположены. В общем:

где такое, что в нём определяется .

Кофенетическое приближение – это мера на n объектах, которая показывает уровень, на котором объекты и впервые попадают в один кластер.

где:

для некоторого q.

Матрица значений называется кофенетической матрицей. Не может быть более, чем n-1 уровней в дендограмме, поэтому не может быть более, чем n-1 кофенетических показателей. Так как такая матрица имеет n(n-1)/2 ячеек, то она имеет очень много пустых мест.

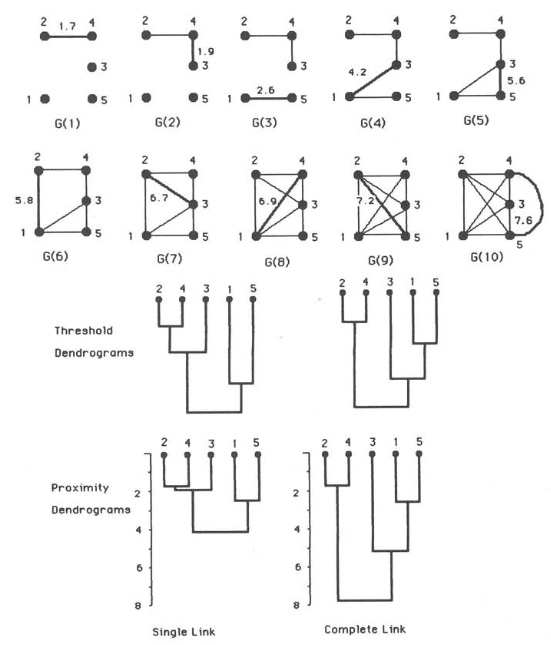
**

Рисунок 2.6. Процесс построения взвешенного графа

Кофенетическая матрица для single-link методов изображена ниже:



Применение single-link метода кластеризации к матрице воспроизводит дендограмму single-link, которая изображена выше. Хотя применения complete-link алгоритма к этой матрице приводит к таким же результатам (complete-link метод может привести к сомнительным результатам, если матрица подобия содержит пустые ячейки).

Кофенетическая матрица является примером матрицы приближения с превосходной иерархической структурой. Как алгоритмы single-link, так и complete-link создают точно такую же дендограмму, если их применить к кофенетической матрице.

Повторим те же действия с дендограммой метода complete-link иерархической кластеризации и получим следующую кофенетическую матрицу :



Кофенетическая матрица также имеет превосходную иерархическую структуру.

### Иерархическая структура и ультраметричность

Доказано, что как single-link, так и complete-link методы генерируют одинаковые дендограммы, если работают с матрицей подобия или с кофенетической матрицей, так как кофенетические матрицы устанавливают «правильные» или «превосходные» иерархические структуры. Подтверждение именования иерархической структуры кофенетической матрицы «правильной» идёт из того факта, что мера кофенетического подобия определена следующим отношением эквивалентности, обозначенным , на множестве объектов:

Отношение может быть представлено как отношение эквивалентности для любого . Для этого проверим три условия отношения эквивалентности – рефлексивность, симметричность и транзитивность.

1. Так как:

то:

тогда рефлексивно.

1. Так как:

то:

тогда симметрично.

1. Если:

тогда:

Это условие должно выполняться для всех .

Это также может быть записано в виде (это выражение называется ультраметрическим неравенством):

Кофенетические матрицы действительно удовлетворяют ультраметрическому неравенству, значит, – отношение эквивалентности для всех . Вложенность в формировании кластеризации гарантирует транзитивность.

Напомним, что только n-1 ячеек из n(n-1)/2 заполнены в кофенетической матрице. Тогда как кофенетическая матрица точно представляет иерархическую структуру, показатели приближения без пустот редко соответствуют правильной иерархической структуре.

Два аспекта должны быть отмечены при рассмотрении ультраметрического неравенства. Первое, кофенетические показатели подобия, полученные из single-link и complete-link кластеризаций, удовлетворяют ультраметрическому неравенству (хотя существуют методы иерархической кластеризации, где кофенические подобия не ультраметричны). Второе, геометрическая интерпретация ультраметрического неравенства демонстрирует, почему меры подобия в приложениях очень редко ультраметричные. Предположим, что каждый объект представлен в виде вектора d-мерного пространства. Если Евклидово пространство – мера подобия, тогда все треугольники, сформированные любыми тремя точками, должны быть равнобедренными треугольниками, такими, что сторона, которая не равна двум другим, имела длину, не большую, чем у остальных сторон.

Исследователи Джардин и Сибсон в 1971 году охарактеризовали иерархические методы кластеризации как отображения из класса матриц подобия в класс мер ультраметрического подобия. Таким образом, методы иерархической кластеризации строят дендограмму по данной матрице подобия, которая устанавливает кофенетическую матрицу, которая удовлетворяет ультраметрическому неравенству.

Свойство ультраметричности также называют монотонностью, так как мера кофенетического приближения удовлетворяют ультраметрическому неравенству, только если кластеры, как и расстояния между ними, монотонно возрастают. Другими словами, кластеризация является вложенной в иерархии. Single-link и complete-link методы всегда монотонны.

### Другие алгоритмы теории графов для single-link и complete-link

Ниже представлен алгоритм, который реализует в себе методы кластеризации и имеет относительно малые вычислительные затраты.

Алгоритм для кластеризации single-link начинает работать с минимальным охватывающим деревом (МОД) для , для которого граф приближения содержит все n(n-1)/2 ребра. Single-link иерархия может быть определена из МОД, но МОД не может быть найден из single-link кластеризации. Для удобства, предположим, что никакие два ребра их МОД не имеют одинаковый вес. Агломеративный алгоритм для single-link кластеризации представлен ниже.

**Алгоритм теории графов для single-link кластеризации**

1. Начинаем с разделённой кластеризации, которая относит каждый объект к отдельной группе.

Повторяем шаги 2 и 3 до тех пор, пока все объекты не окажутся в одном кластере.

1. Соединяем два кластера ребром из МОД с наименьшим весов для определения следующей кластеризации.
2. Заменяем вес выбранного во втором шаге ребра весом, большим, чем наибольшее расстояние (наибольший показатель подобия).

Этот разделяющий алгоритм очень прост в своей реализации. В нём просто вырезаются рёбра МОД в порядке их веса и заменяются рёбрами с очень большим весом. Каждое удаление ребра определяет новую кластеризацию: те объекты, между которыми было удалённое ребро, относятся к одному кластеру. Этот алгоритм создают те же single-link кластеризации, что и алгоритмы, представленные ранее.

**Пример**

Пусть определена следующая матрица подобия :



Пример работы разделяющего алгоритма single-link изображён на Рисунке 2.7:

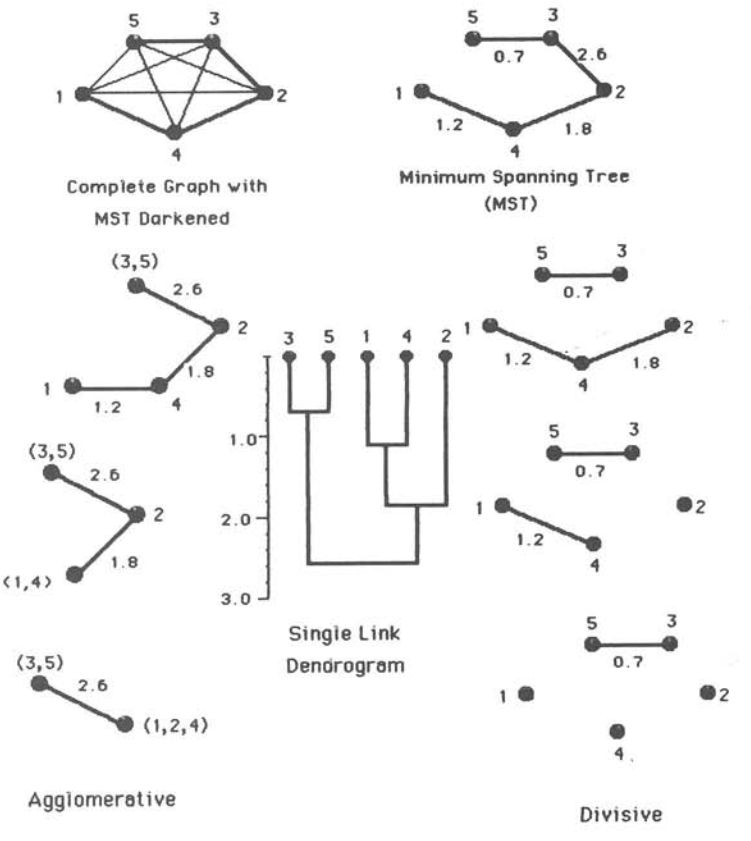


Рисунок 2.. МОД для single-link иcomplete-link

### Алгоритмы преобразованной матрицы для single-link и complete-link

В этой части будет говориться об алгоритмах single-link и complete-link кластеризаций в терминах преобразования матрицы подобия. Этот подход предложен исследователем Кингом в 1967 году, и в том же году сформулирован Джонсоном. Это агломеративный алгоритм, который удаляет строки и столбцы матрицы подобия, когда старые кластеры соединяются в один новый. Упростим алгоритм предположением того, что матрица подобия не имеет пустых ячеек.

Пусть есть матрица подобия размерностью . Кластеризации помечаются последовательностью целых чисел и - это уровень k-й кластеризации. Кластер с номеров m в последовательности помечается как и подобие между кластерами и будем обозначать .

**Алгоритм Джонсона для single-link и complete-link кластеризаций**

1. Начинаем работу с несоединённой кластеризации, которая имеет уровень и номером последовательности .
2. Находим наименее отличающуюся пару кластеров в текущей кластеризации, обозначаем её , соответственно

то есть, расстояние между этими двумя кластерами является наименьшим.

1. Увеличиваем номер последовательности: . Объединяем кластеры и в один кластер, формируя следующую кластеризацию. Устанавливаем уровень кластеризации:
2. Преобразовываем матрицу подобия, удалив строки и колонки, соответствующие и , и добавив строку и столбец, которые соответствуют недавно сформированному кластеру. Мера подобия между новым кластером, обозначенным и другим «старым» кластером определяется следующими способами. Для single-link:

Для complete-link:

1. Если все объекты находятся в одном кластере – стоп. Иначе – идём в шаг 2.

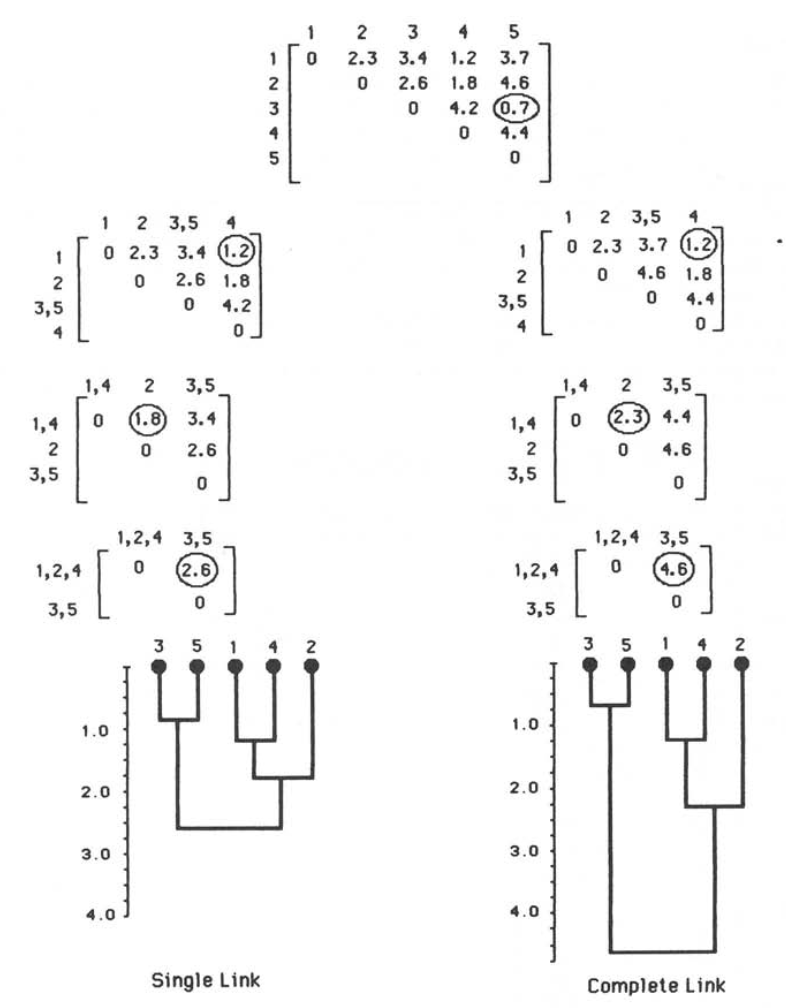
­

Рисунок 2.. Пример работы алгоритма Джонсона

## Частичная кластеризация

Техники иерархической кластеризации организовывают данные во вложенные последовательности групп. Важной характеристикой методов иерархической кластеризации является их визуализация в виде дендограммы, которая позволяет оценить, как данные формировали кластеры или разделить их по уровням подобия. Аналитики данных могут вручную выбрать наиболее верную кластеризацию или сделать какие-то изменения и оценки. Далее будем называть неиерархические методы кластеризации частичными методами кластеризации. Они генерируют единственное разделение данных и пытаются восстановить естественные группы данных. Обе стратегии кластеризации имеют одинаковые сферы приложений. Методы иерархической кластеризации обычно требуют только матрицу подобия, относящуюся к объектам, тогда как частичные методы обрабатывают данные в форме векторной матрицы. В общем требуется, чтобы данные были измеримы в относительной шкале.

Иерархические техники популярны в биологии, социологии и поведенческой психологии, потому что там требуется строить таксономии. Частичные методы часто используются в инженерных приложениях, где единственность разбиения важно. Частичные методы кластеризации используются особенно для эффективного представления и сжатия больших объёмов данных. Дендограммы не практичны в работе с более, чем несколькими сотнями векторов.

Проблема частичной кластеризации может быть формально определена следующим образом. Пусть дано n векторов в d-мерном метрическом пространстве, нужно определить разделение векторов на K групп или кластеров так, чтобы векторы в кластере были больше подобны друг другу, чем векторам из других кластеров. Величина K может быть как определена, так и не определена. Должны быть приняты критерии кластеризации, такие как, например, среднеквадратичная ошибка. Критерий может классифицировать как глобальный или локальный. Глобальный критерий представляет каждый кластер через прототип и относит векторы к кластерам с наиболее подобным прототипом. Локальный критерий формирует кластер, соотнося локальную структуру данных. Например, кластеры могут формироваться распознаванием областей с высокой плотностью объектов или выборкой векторов и соотнесением ближайших соседей в один кластер.

Теоретическое решение этой проблемы простое. Выбрать критерий, оценить все возможные разделения, содержащие K кластеров и выбрать среди них самое оптимальное. Первая трудность, которая встречается, выбрать критерий, который переводит чей-то взгляд о кластере в разумную математическую формулу. Критерий сильно зависит от параметров и должен быть простым для вычислительных задач, в то же время должен быть комплексно достаточным для отражения различных структур данных. Вторая трудность данного подхода – это то, что число возможных разделений огромное, даже для небольшого числа векторов, поэтому оценивать простейший критерий на всех разделениях непрактично.

Обозначим через число возможных разбиений n объектов на K кластеров. Порядок объектов в кластерах и самих кластеров неважен. Пустые кластеры не учитываются. Предположим, что все кластеризации для n-1 объектов были перечислены. Кластеризация для n объектов может быть сформирована из этого списка двумя способами:

1. Объект под номером n может быть добавлен как одиночный кластер, если число кластеров равно K-1.
2. Объект может быть добавлен в какой-либо из существующих кластеров, если их число равно K.

Дифференциальное уравнение в частных производных может быть записано для следующим образом:

Граничные условия для этого уравнения:

Решение этого уравнения для требуют, чтобы значения были известны для множества: .

Решения для этого дифференциального уравнения в частных производных, называемые числами Стирлинга, выглядят следующим образом:

Существует 34105 определённых разбиений для десяти объектов на четыре кластера, и оно возрастает до, приблизительно, 11259666000 разбиений для девятнадцати объектов на четыре кластера. Проще говоря, полное перечисление всех разбиений вычислительно невыполнимо даже для малого количества векторов. Кроме того, число K может быть не выбрано априори.

Чтобы избежать этого комбинаторного возрастания, критерий оценивается только для небольшого множества «разумных» разбиений. Вопрос стоит в том, как определить небольшое множество разбиений, которые имеют наибольший шанс оказаться оптимальным. Наиболее общий подход – оптимизировать функцию критерия, используя итеративную технику. Начинаем с начального разбиения и пытаемся улучшить значение функции перемещением объектов из одного кластера в другой. Таким образов, удачное разбиение – это возмущение предыдущего и, таким образом, только малое число разбиений будет рассмотрено. Алгоритмы, основанные на этой технике, вычислительно эффективны, но очень часто охватывают локальный минимум функции критерия.

Другой способ избежать комбинаторного взрыва – каким-то образом идентифицировать и исключить разбиения, которые не подходят под данную задачу. Дженсен в 1966 году использовал динамический программный подход для исключения большого количества разбиений, и это по-прежнему может быть оптимальным решением. Значительная вычислительная экономия реализована особенно сильно для больших задач кластеризации. Для примера, для разделения девятнадцати объектов на четыре кластера, нужно оценить только два процента от всех возможных разбиений, используя этот подход. Но даже этого снижения сложности не достаточно для эффективного снижения сложности.

Не существует единственного «лучшего» подхода для получения оптимального разбиения, потому что не определено точное или рабочее понятие кластера. Кластеры могут быть произвольной формы или размера в векторном пространстве. Каждый критерий кластеризации налагает точную структуру на данные и, если данные не удовлетворяют требованиям конкретных критериев, настоящий кластер может быть перекрыт. Только небольшое число независимых критериев кластеризации может быть понятно как математически, так и интуитивно. Литература по кластерному анализу столь разнообразна и покрывает так много областей науки, что некоторые функции критерия переоткрываются постоянно.

Некоторые популярные критерии используют квадратичные ошибки. Некоторые находят наиболее подходящую плотность объектов в кластере. Другие создают кластеры, используя гиперэллиптические фигуры. Некоторые частичные кластеризации основаны на плотности, частичной оценке, соединённости графа.

Наиболее распространённые техники частичной кластеризации неявно представляют в себе непрерывно-значные характеристики, поэтому векторы, над которыми будет работать алгоритм, должны быть обёрнуты в метрическое пространство. Если атрибуты представлены в номинальной или относительной шкале, расстояние Евклида и центры кластеров не очень значимы, поэтому иерархические алгоритмы применимы. Вонг и Ванг в 1979 году предложили алгоритм кластеризации для дискретно-значных данных.

Техника абстрактной кластеризации или обучения из примеров может быть использована с объектами данных, которые представлены нечисленными или символическими идентификаторами. Подход этой техники – группировка объектов в абстрактные, простые классы. Например, кластеризацию автопоездов, использующую такие атрибуты, как количество машин, количество колёс, цвета колёс или кластеризацию микрокомпьютеров, использующую в качестве характеристик скорость ЦПУ, объём памяти или тип процессора, уместно обрабатывать связыванием каждого кластера с простой «концепцией». Концепции определены в терминах атрибутов. К примеру, в проблеме классификации автопоездов, автопоезда с двумя красными машинами – это концепция. Объекты, организованные в иерархии классов, описываются с помощью концепций.

### Критерий кластеризации квадратичной ошибок

Наиболее часто используемые стратегии частичной кластеризации основаны на критерии квадратичной ошибки. Общий подход – определить разбиение, которое, для фиксированного числа кластеров, минимизирует квадратичную ошибку.

Предположим, что данное множество из векторов, заданных в размерностях, некоторым образом разделено на K кластеров таким образом, что кластер имеет векторов и каждый вектор принадлежит точно одному кластеру. Тогда:

Средний вектор или центр кластера определён как центроид кластера, или:

где - это i-й вектор, принадлежащий кластеру . Квадратичная ошибка для кластера – это сумма возведённых в квадрат расстояний Евклида между каждым вектором в и его центром . Эта ошибка также называется внутрикластерной вариацией.

Расстояние Махланобиса также может быть использовано в определении квадратичной ошибки.

Квадратичная ошибка для всей кластеризации, содержащей K кластеров – это сумма всех внутрикластерных вариаций:

Подход метода кластеризации квадратичной ошибки – это найти такое разбиение, содержащее K кластеров, которое минимизирует для фиксированного K. Итоговое разбиение также может быть рассмотрено как разбиение с минимальной вариацией. Рисунок ниже иллюстрирует, что критерий квадратичной ошибки представляет центроид кластера как прототип. Другими словами, векторы представлены как множество из K сферически-образных куч. Этот метод пытается сделать K кластеров как можно более компактными и разделёнными.

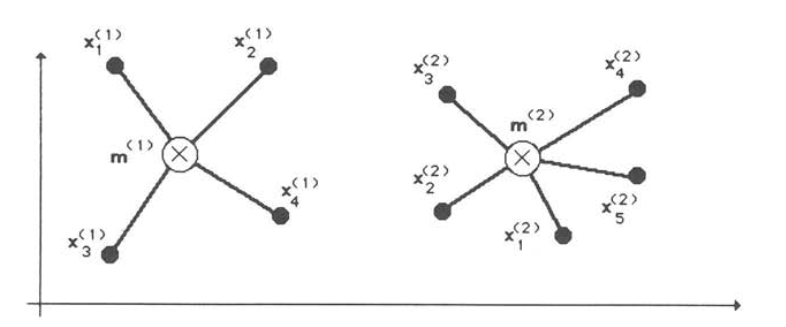


Рисунок 2.9. Центроиды некоторых объектов

Учёные Гордон и Хендерсон также определили проблемы кластеризации в терминах минимизации суммы внутрикластерных расстояний. Обозначим j-ю характеристику i-го вектора, . Будем считать, что , если i-й вектор принадлежит k-му кластеру, и 0, если не принадлежит, . Центроид k-ого кластера обозначим , который записывается как , где:

Суммарная внутрикластерная вариация, обозначенная также , записывается также:

Эти исследователи предположили две формулировки минимизации :

1. Пусть дана векторная матрица и число кластеров K. Тогда нужно найти матрицу размерностью которая минимизирует . Заметим, что эта матрица состоит из нулей и единиц, и в каждой строке содержится только одна единица, в каждом столбце единица встречается минимум один раз.
2. Более общая формулировка минимизации лежит в предположении в ограничении:

В терминах, обозначает дробь i-го вектора, который принадлежит k-му кластеру. Эта концепция схожа с нечёткой кластеризацией, в котором вектор принадлежит к кластеру со «степенью членства». Лемма Гордона и Хендерсона показывает, что матрица , которая минимизирует с ограничениями того, что она должна содержать в себе только 0 и 1. Таким образом, первая формулировка – это специальный случай второй формулировки. Алгоритм, используемый для минимизации , основан на пошаговом понижении.

Число критериев кластеризации, относящихся к квадратичной ошибке, определено из матрицы разбиений, используемой в дискриминантном анализе. Они основаны на разложении общей матрицы разбиений на внутрикластерную матрицу разбиений и на межкластерную матрицу разбиений :

Общая матрица разбиений зафиксирована и не имеет значения, как данные векторы разделены. Эти матрицы скалярные в одной размерности, поэтому очевидно, что увеличение межкластерного разбиения уменьшает внутрикластерное разбиение , и наоборот. Для определения критерия кластеризации в терминах матриц разбиений, нужно представить «размер» кластеров с помощью оператора определителя и трейсах. Число критериев кластеризации раскрыто ниже в терминах разбиения.

Сначала, покажем, что критерий кластеризации, определённого на трейсах и идентичен критерию квадратичной ошибки:

Заметим, что – это сумма вариаций вдоль направления характеристики для кластера k и мера компактности кластера . Минимизация идентична максимизации , так как:

и одинаков для всех разбиений. Критерий трейсов и эквивалентный ему критерий квадратичной ошибки инвариантны ортогональным трансформациям в векторном пространстве, таким как вращение, но не инвариантны несингулярным линейным трансформациям. Это значит, что разбиение с минимальной квадратичной ошибкой может измениться, если координатные оси масштабируются.

Определитель матрицы не очень хороший критерий кластеризации, так как становится сингулярной, если число кластеров меньше чем число характеристик . Минимизация определителя может иметь преимущество перед критерием квадратичной ошибки, так как инвариантна для несингулярных линейных трансформациях векторов. Хотя становится сингулярной, если или если векторы лежат в подпространстве пространства характеристик. Линейные трансформации могут использоваться для уменьшения размерности данных, если сингулярна.

Собственные векторы определяет проекцию векторов на пространство, размерностью . Эти собственные векторы инвариантны для несингулярных линейных трансформаций векторной матрицы. Далее, так как собственные значения определяют отношение между межкластерным и внутрикластерным разбиением, можно определить два дополнительных критерия кластеризации. Предположим, что существует значимых собственных значений , тогда эти критерии могут быть выражены следующими способами:

Методы кластеризации, основанные на этих критериях, выбирают разбиение для каждого или , которое эти значения максимизирует. Заметим, что максимизация эквивалентно минимизации , потому что величина не зависит от выбора разбиения данных.

Критерии кластеризации, описанные выше, ищут шарообразные или гипер-эллипсоидные кластеры. Различные критерии ведут к различным кластеризациям. Критерий квадратичной ошибки менее требовательны к вычислениям, чем критерий, основанный на соотношении различия, так как последний требует вычисления собственного значения после каждого разделения. К сожалению, не существует общей инструкции, какой критерий следует выбрать. На практике следует создавать разбиения, используя различные критерии и выбирать «лучший», используя некоторую схему валидации.

**Пример**

Предположим, что четыре вектора в двумерном пространстве, показанных на рисунке ниже, должны быть кластеризированы. Достаточно просто вычислить матрицу разбиения для следующих трёх разбиений:

Таким образом, третье разбиение лучшее из представленных трёх в соответствии с критерием относительно , который эквивалентен критерию квадратичной ошибки. Хотя первые два разбиения выбраны с помощью критерия, использующего в качестве параметра .

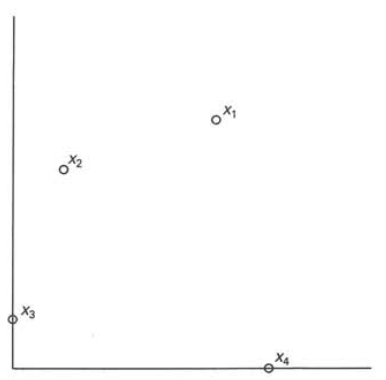


Рисунок 2.10. Множество векторов

### Метода кластеризации квадратичной ошибки

Основная идея итерактивного алгоритма кластеризации – начать с исходного разбиения и распределять векторы по кластерам так, чтобы уменьшать квадратичную ошибку. Квадратичная ошибка стремится к уменьшению, если число кластеров возрастает и может быть минимизировано только для фиксированного числа кластеров. Частичный итерактивный метод кластеризации может быть имплементирован разными способами. Разные реализации могут вести к разным разбиениям. Исследователи Дубес и Джейн подчёркивали различие между методами кластеризации и алгоритмами кластеризации. Метод кластеризации определяет генеральную стратегию для группировки векторов в кластеры, как, например, минимизация квадратичной ошибки или максимизация . Алгоритм кластеризации, с другой стороны, – это компьютерная программа, которая реализовывает стратегию или различные эвристики. Общий алгоритм для итерактивного частичного метода кластеризации описан ниже. Андересн провёл широкое обсуждение деталей этого подхода.

**Алгоритм итеративной частичной кластеризации**

1. Выбираем исходное разбиение на K кластеров.
2. Создаём новое разбиение определением каждого вектора к ближайшему центру кластера.
3. Вычисляем новые центры или центроиды кластеров.
4. Повторяем шаги 2 и 3 до тех пор, пока не найдётся оптимальной значение функции критерия.
5. Приспосабливаем число кластеров соединением или разбиением существующих кластеров или удалением малых, незначительных кластеров.
6. Повторяем шаги 2-5, пока не стабилизируется кластерное «членство»

Детали шагов этого алгоритма должны быть включены либо пользователем в качестве параметров, либо неявно скрыты в компьютерной программе. Хотя эти детали являются ключевыми для успеха программы. Большое разочарование в использовании программ кластеризации – это недостаток инструкций выбора деталей. Далее будут кратко рассмотрены доступные ключевые параметры и опции.

**Исходное разбиение.** Исходное разбиение может быть сформировано точным определением множества K начальных точек. Начальными точками могут быть первые K векторов или K векторов, выбранных случайно из векторной матрицы. Множество из K векторов, которые хорошо разделены от других, могут быть получены с помощью фиксирования центроида всех данных, как первой начальной точки, и выбором последовательно точек, расстояние от которых до начальных точек уже выбрано. Начальное разбиение или кластеризация формируется определением каждого вектора к ближайшей начальной точке. Центроиды полученных кластеров – это исходные центры кластеров. Иерархическая кластеризация данных также использует выбор начального разбиения.

Разные начальные разбиения могут привести к разным конечным кластеризациям, так как алгоритмы, основанные на квадратичной ошибке могут сходиться к локальному минимуму. Это особенно правдиво, если кластеры не очень хорошо разделены. Один из способов преодолеть локальный минимум – выполнить частичный алгоритм кластеризации с различными исходными разбиениями. Если они все приводят к одному и тому же разбиению, есть уверенность, что глобальный минимум был достигнут.

**Обновление разделения.** Разбиения обновляются с помощью перераспределения векторов в кластеры в попытке уменьшить квадратичную ошибку. Термин «проход» или «цикл» означает процесс исследования каждого объекта в кластере один раз. МакКвин определил K-mean проход как распределение всех векторов к ближайшему центру кластера. Центр полученного кластера перевычисляется после каждого нового распределения. Метод Форги вычисляет центры кластеров после рассмотрения всех векторов данных. Евклидова метрика – наиболее общая метрика для вычисления расстояния между вектором и центром, но расстояние Махаланобиса также используется. Хотя расстояние Махаланобиса требует вычисления обратной ковариантной матрицы каждый раз, когда вектор изменяет некоторый кластер.

Исследователи Фриедман и Рабин определили hill-climbing проход и forcing проход в алгоритме кластеризации, основанном на инвариантном критерии, который использует матрицу разбиения. Hill-climbing проход изменяет кластер, относящийся к алгоритму, только для улучшения функции критерия. Forcing проход возмущает разбиение для избежания попадания в локальный минимум функции критерия. Forcing проход относит каждый вектор к разным кластерам.

**Регулировка числа кластеров.** Некоторые алгоритмы кластеризации могут создавать новые кластеры или соединять существующие, если определённые условия выполнены. Эта возможность позволяет алгоритмам переделывать плохие исходные разбиения и выбирать более «натуральное» или «подходящее» число кластеров, особенно, если число полученных кластеров не очень соответствующее. В одном из наиболее популярных алгоритмов кластеризации, называемом ISODATA, эти условия определены как параметры пользователем программы. Кластер разбивается, если он имеет слишком много векторов в себе и необычно большое расхождение вдоль характеристики с наибольшим распространением. Два кластера соединяются, если их центры очень близки, и это также основано на параметрах пользователя.

Изолированным называется вектор, который находится настолько далеко от остальных данных, что можно подозревать, что он был включён в данные по ошибке. Очень часто, изолированный вектор – это ошибка измерительного процесса или начальных данных. Изолированные векторы могут предоставить полезную информацию о процессе создания лежащих в основе данных, но принуждение такого вектора к принадлежности некоторому кластеру может привести к искажению формы этого кластера. Рисунок ниже демонстрирует, что изолированный вектор может заставить частичную кластеризацию отнести две компактные и хорошо разделённые группы в один кластер. Таким образом, лучше всего идентифицировать изолированный вектор и удалить его из дальнейшего рассмотрения. Некоторые алгоритмы кластеризации относят также «малые» кластеры к изолированным.

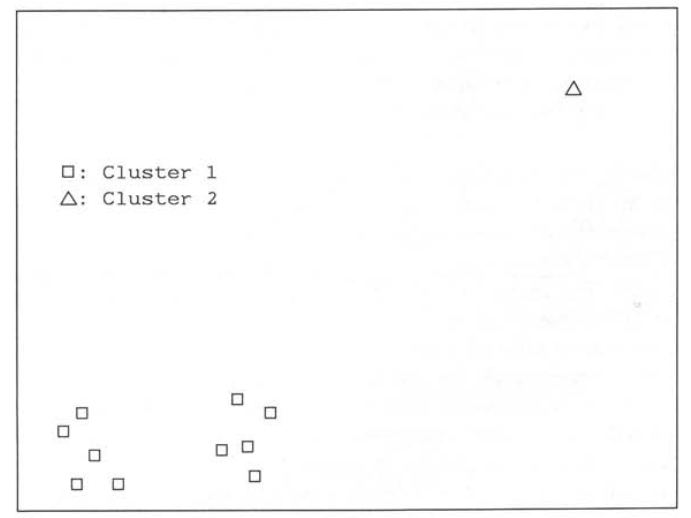


Рисунок 2.11. Пример ошибочной работы алгоритма из-за наличия изолированной точки

**Сходимость.** Стоит вопрос в том, когда нужно остановить работу алгоритма. Частичные алгоритмы кластеризации останавливаются в тот момент, когда определяют, что улучшить функцию критерия уже нельзя. Не существует гарантии, что итеративный алгоритм достигнет глобального минимума. Некоторые алгоритмы останавливаются, когда кластеры, относящиеся ко всем векторам, не изменяются между двумя успешными итерациями. Максимальное число итераций может быть определено, чтобы предотвратить бесконечные колебания. На практике, алгоритмы кластеризации K-means сходятся быстро. На рисунке ниже показаны два хорошо разделённых кластера в двумерном пространстве. Не смотря на то, что две начальных рассматриваемых вектора принадлежат к одному кластеру, сходимость k-means алгоритма к корректному разбиению требует только две итерации.

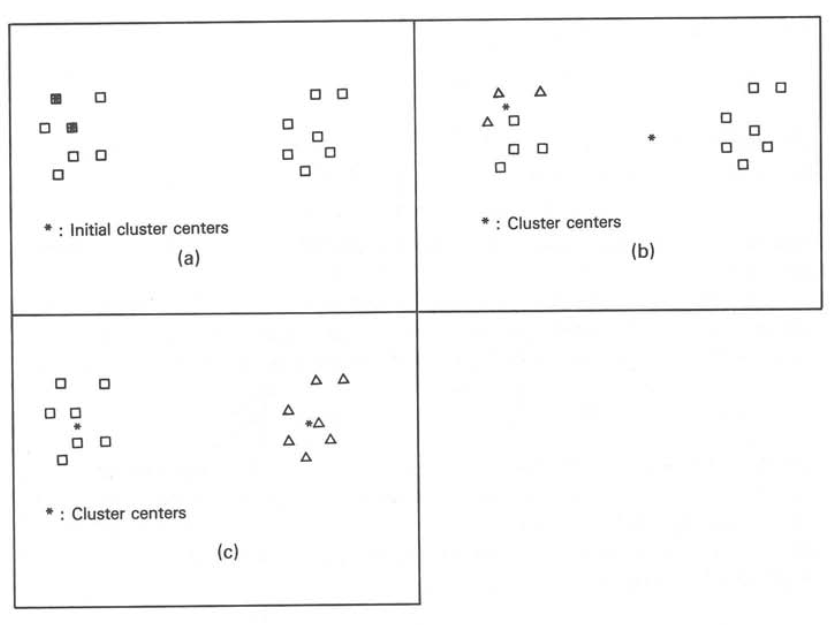


Рисунок 2.12. Итерации частичной кластеризации

Учёные Селим и Измаил в 1984 году строго доказали сходимость K-means алгоритма. Проблема разбиения n d-мерных векторов на K кластеров может быть сформирована в виде следующей математической программной задачи. Минимизировать взвешенную сумму Евклидовых расстояний между векторами и центрами кластеров:

с граничными условиями:

Матрица размерностью - это матрица веса каждого вектора в каждом кластере и - матрица размером центров кластеров:

Функция является невыпуклой, и её локальный минимум должен быть глобальным.

Рассмотрим данные, показанные на рисунке ниже. Если два исходных центра кластеров – это:

тогда K-means алгоритм остановится через одну итерацию и сойдётся к решению: , и: , которые производят два кластера и . Селим и Измаил показали, что это решение даже не локальный минимум, так как немного возмутив M получаем , которое получается из следующих центров кластеров:

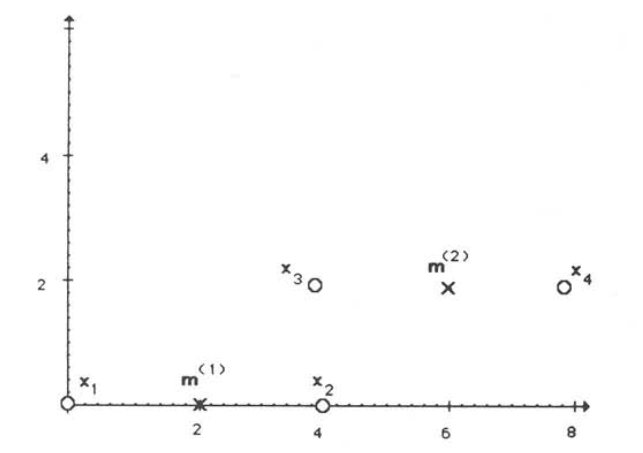


Рисунок 2.13. Данные, представленные в двумерном пространстве

**Вычисление.** Вычислительная сложность этого алгоритма – это , где n – это число векторов, d – число характеристик, K – количество кластеров и T – количество итераций. Значение T зависит от исходных центров кластеров, распространения векторов и от размера задачи кластеризации. Хотя на практике, пользователь устанавливает верхнюю границу T. Итеративная сущность кластеризации квадратичной ошибки требует значительного процессного времени, даже для нескольких сотен векторов.

### Метод ближайшего соседа кластеризации

Естественный способ определить кластер – это использование свойств ближайшего соседа; вектор должен быть помещён в тот же кластер, что и его ближайший сосед. Два вектора следует считать подобными, если они разделяют соседство. Понятие ближайшего соседа неотъемлемо в конструкциях различных графов, поэтому методы кластеризации, основанные на теории графов очень близки к методам ближайшего соседа. Хотя они отличаются в том, как формируются кластеры, и, что более важно, в финальным разбиением.

Очень простой алгоритм, который основан на правилах ближайшего соседа, рассмотрен ниже. Множество векторов нужно разбить на K кластеров. Пользователь указывает границу для расстояния ближайшего соседа.

**Алгоритм ближайшего соседа**

1. Устанавливаем и . Присваиваем вектор кластеру .
2. Устанавливаем . Находим ближайшего соседа вектору среди объектов, уже отнесённых к некоторым кластерам. Обозначим расстояние от до его ближайшего соседа. Предположим, что ближайший сосед находится в кластере m.
3. Если , то относим к . Иначе, устанавливаем и относим к новому кластеру .
4. Если каждый вектор уже отнесён к некоторому кластеру - стоп. В ином случае идём на шаг 2.

Число созданных кластеров K является функцией от параметра t. Если значение t возрастает, меньшее число кластеров создаётся. Расстояние до ближайшего соседа в шаге 2 может быть заменено на среднюю дистанцию между и p ближайшими соседями в кластере m. В таком случае, пользователь может определить ещё один параметр p. Исследователи Лю и Фу в 1978 году использовали этот алгоритм для данных, представленных в виде предложений или строк в приложении синтаксического распознавания.

Учёные Джарвис и Патрик в 1973 году определили меру подобия как число совпадений с списке ближайших соседей двух векторов. Их алгоритм кластеризации может быть подведён следующим образом: разместить векторы и в один и тот же кластер, если и разделяют как минимум ближайших соседей и и входят в k ближайших соседей друг другу. Этот алгоритм неитерактивный и его вычислительная сложность может быть привлекательной, если ближайшие соседи эффективно вычисляются. Пользователь должен определить размер окружения или соседства k, а также границу подобия . Джарвис и Патрик не предоставили никаких инструкций для выбора этих параметров, но предположили, что это значение можно вычислить итерактивно. Можно заметить, что большое значение k помогает алгоритму находить глобальные структуры, в то время как малые значения – локальные. Иерархия может создаваться за счёт вариации значения .

Понятие подобия, основанного на разделении ближайших соседей, было модифицировано в меру «взаимной близости» двух векторов. Если является p-м ближайшим соседом к , а – это q-й близкий сосед к , то значения взаимного соседства (ЗВС) между и определяется как . Чем меньше ЗВС, тем более подобны векторы. Это представляет более строгое понятие подобия, чем количество разделённых ближайших соседей.

**Алгоритм взаимного соседства кластеризации**

1. Определить k ближайших соседей для каждого вектора.
2. Вычислить ЗВС для каждой пары векторов. Если и не являются взаимно ближайшими соседями для заданного значения k, то установить ЗВС() произвольно большим числом.
3. Идентифицировать все пары векторов по ЗВС из шага 2. Соединять каждые пары в кластер, начиная с пары, имеющей наименьшее ЗВС.

Повторять шаг 3 для ЗВС с границами 3, 4 … 2k для генерации иерархии.

Параметр k, который определяет глубину соседства, является ключевым в производительности алгоритма. Малые значения k несколько «сильных» кластеров, большие значения генерируют меньшее число «слабых» кластеров. Фактически, k всегда можно выбрать достаточно большим для того, чтобы алгоритм вернул единственный кластер.

Этот алгоритм может идентифицировать несферические кластеры, линейно не разделимые кластеры, кластеры неравных размеров.

### Нечёткая кластеризация

Алгоритмы, которые были раскрыты ниже, определяли каждый вектор к одному и только одному кластеру. Другими словами, векторы разделяются на непересекающиеся множества. Если кластеры компактны и хорошо разделены, как показано на рисунке ниже, не существует неопределённости или неизвестности, в какой кластер должен быть определён каждый вектор. Легко видеть, что на рисунке ниже существует два кластера, с хорошо определёнными границами.

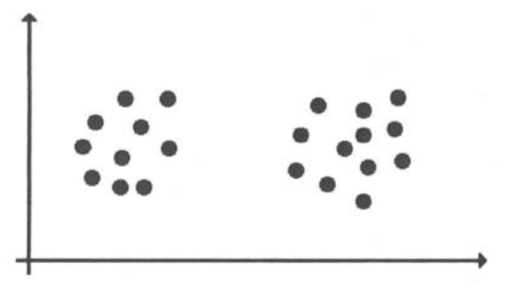


Рисунок 2.14. Хорошо разделённые кластеры

Но часто вопрос состоит в том, как поступать, если кластеры касаются или перекрываются. Рисунок ниже показывает случай, в котором границы кластеров нечёткие и распределение векторов по кластерам затруднено. Векторы и , находящиеся в разных кластерах, могут оказаться в кластере, который включает оба этих вектора. Говорят, что такие объекты данных имеют нечёткую границу.

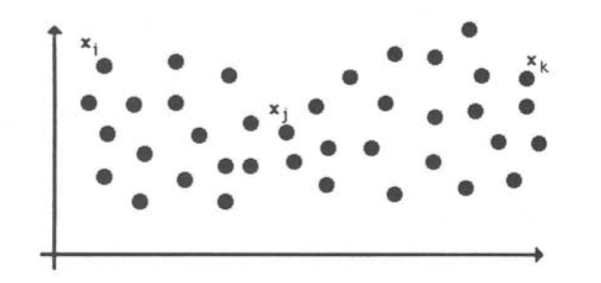


Рисунок 2.15. Плохо разделённые кластеры

Теория нечётких множеств, разработанная учёным Задехом, позволяет объектам иметь степени членства в кластерах. Степень членства принимает значение в интервале . Для обычных кластеров, называемых «чёткими», степень членства для вектора равна единице, если он принадлежит этому кластеру, и ноль, если не принадлежит. В нечёткой кластеризации вектор имеет степень членства или принадлежности для q-ого кластера, где . Чем больше , тем больше уверенность в том, что вектор принадлежит кластеру q. Если , что абсолютно точно, что принадлежит кластеру j. Интерпретация таких значений, как 0.3 менее ясная. Степень принадлежности является субъективной и оно основано на определении, заменяющим меры. Например, один вектор может принадлежать одному кластеру со степенью 0.45, а другому – 0.55.

Степень принадлежности – это не то же самое, что и вероятность того, что вектор принадлежит определённому кластеру, несмотря на то, что оба эти значения принимают значения в интервале [0, 1]. В вероятностной структуре вектор принадлежит только одному кластеру, в зависимости от результата случайного эксперимента. В теории нечётких множеств объект может принадлежать разным группам одновременно. Степени членства определяют, насколько кластеры подходящие.

Другая интерпретация степени принадлежности – это мера совместимости вектора или объекта с описанием нечёткого множества. Иногда это свойство полезно для интерпретации результатов алгоритма. Рассмотрим проблему кластеризации различных компьютеров в диапазоне от микрокомпьютеров до системных блоков, основанной таких атрибутах, как объём памяти, скорость ЦПУ и тип процессора. К какому кластеру отнести компьютер, основанный на процессоре Motorola 68020 – к персональным компьютерам или рабочим станциям, точно неизвестно.

Сторонники нечёткой кластеризации доказывают, что она лучше, чем обычная кластеризация для человеческих понятиях, таких как «большой», «маленький», «высокий» и «низкий». Нечёткая кластеризация подходит в приложениях, с потерянной или недостаточной информацией. Скептики кластеризации такого вида не оспаривают её математическую корректность, но они не уверены, что нечёткая кластеризация имеет только преимущества перед классической кластеризацией.

Ключевой шаг в алгоритме нечёткой кластеризации – это определение функции принадлежности или членства. Баккер в 1978 году показал, как сконструировать функцию принадлежности, основанной на декомпозиции схожести. Установим, что векторов вначале разбиты на кластеры и пусть - это число векторов в кластере . Обозначим через схожесть между вектором x и кластером . Чем больше эта величина, тем ближе эти вектор и кластер. Функция членства для вектора x и кластера – это:

где - относительный размер кластера . Эта функция членства неотрицательна и суммируется к 1 для каждого вектора:

Функция подобия или близости может быть основана на концепции расстояния, концепции соседства или концепции вероятности. Это мера отношения между вектором и кластером в целом или между вектором и одним или более образцами кластера. Выбор этой функции зависит от данных. Баккер и Джейн определили функцию близости, основанную на собственном векторе кластера:

где обозначает расстояние Евклида вектором x и центроидом кластера . Параметр контролирует размер соседства и оказывает воздействие на принадлежность кластера. Производительность нечётких алгоритмов кластеризации критически зависит от выбора функции членства.

Нечёткие алгоритмы создают разбиения, которые минимизируют вынужденные нечёткости, такими же шагами, как и алгоритмы квадратичной ошибки. Вынужденная нечёткость принимает минимальное значение, если мы может обернуть разбиение функцией , или, что эквивалентно, разбиение чёткое. Таким образом, функция критерия должна быть определена для характеристики вынужденных нечёткостей в разбиении. Например следующая функция:

где K – это число кластеров и – это пересечение двух нечётких множеств, определённое следующим образом:

Минимальное значение - 0, которое представляет максимальную нечёткость, максимальное – 1, которое отвечает за чёткое разбиение. Проблема нечёткой кластеризации состоит в нахождении такого разбиения, которое максимизирует . Основные шаги нечёткой кластеризации даны ниже.

**Алгоритм нечёткой кластеризации**

1. Выбрать исходное разбиение .
2. Вычислить функцию членства .
3. Вычислить функцию критерия .
4. Переклассифицировать векторы, чтобы улучшить .
5. Повторять шаги 2 – 4 до стабилизации.

Результат работы алгоритма нечёткой кластеризации включает в себя не только разбиение, но и информацию в форме значений членства. Хотя новая информация, полученная из значений членств более нужна анализу данных. В итоге, нечёткая кластеризация является интересной концепцией, включающая в себя разные методы частичной кластеризации.

### Алгоритмы семейства FOREL

FOREL (Формальный элемент) — алгоритм кластеризации, основанный на идее объединения в один кластер объектов в областях их наибольшего сгущения.

Минимизируемый алгоритмом функционал качества:

где первое суммирование ведётся по всем кластерам выборки, второе суммирование – по всем объектам x, принадлежащим текущему кластеру – центр текущего кластера, – расстояние.

Необходимые условия работы:

* выполнение гипотезы компактности, предполагающей, что близкие друг к другу объекты с большой вероятностью принадлежат к одному кластеру (таксону).
* наличие линейного или метрического пространства кластеризуемых объектов

Входные данные:

1. Кластеризуемая выборка
2. Параметр R – радиус поиска локальных сгущений
3. Опционально k – число кластеров

На выходе получается кластеризация на заранее неизвестное число таксонов.

Принцип работы алгоритма состоит в том, что на каждом шаге мы случайным образом выбираем объект из выборки, строим вокруг него сферу радиуса R, внутри этой сферы выбираем центр тяжести и делаем его центром новой сферы. Таким образом, на каждом шаге сфера двигается в сторону локального сгущения объектов выборки, то есть алгоритм старается захватить как можно больше объектов выборки сферой фиксированного радиуса. После того как центр сферы стабилизируется, все объекты внутри сферы с этим центром помечаются как кластеризованные и выходят из выборки. Этот процесс повторяется до тех пор, пока вся выборка не будет кластеризована.

Пошаговое описание алгоритма:

1. Случайно выбирается текущий объект из выборки.
2. Помечаются объекты, находящиеся на расстоянии менее, чем R от текущего.
3. Вычисляется их центр тяжести, помещается этот центр как новый текущий объект.
4. Повторяются шаги 2-3, пока новый объект не совпадёт с текущим.
5. Помечаются объекты внутри сферы радиуса R вокруг текущего объекта как кластеризованные, они выходят из выборки.
6. Шаги 1-5 повторяются до тех пор, пока не будет кластеризирована вся выборка.

Эвристика выбора центра тяжести:

1. В линейном пространстве – центр масс.
2. В метрическом пространстве – объект, сумма расстояний до которого минимальна, среди всех внутри сферы.
3. Объект, который внутри сферы радиуса R содержит максимальное количество других объектов из всей выборки (медленно).
4. Объект, который внутри сферы маленького радиуса содержит максимальное количество объектов (из сферы радиуса R).

Наблюдения:

1. Доказана сходимость алгоритма за конечное число шагов.
2. В линейном пространстве центром тяжести может выступать произвольная точка пространства, в метрическом – только объект выборки.
3. Чем меньше R, тем больше таксонов (кластеров).
4. В линейном пространстве поиск центра происходит за время О(n), в метрическом O(n²).
5. Наилучших результатов алгоритм достигает на выборках с хорошим выполнением условий компактности.
6. При повторении итераций возможно уменьшение параметра R, для скорейшей сходимости.
7. Кластеризация сильно зависит от начального приближения (выбора объекта на первом шаге).
8. Рекомендуется повторная прогонка алгоритма для исключения ситуации «плохой» кластеризации, по причине неудачного выбора начальных объектов.

Преимущества:

1. Точность минимизации функционала качества (при удачном подборе параметра R).
2. Наглядность визуализации кластеризации.
3. Сходимость алгоритма.
4. Возможность операций над центрами кластеров (они известны в процессе работы алгоритма).
5. Возможность подсчета промежуточных функционалов качества, например, длины цепочки локальных сгущений.
6. Возможность проверки гипотез схожести и компактности в процессе работы алгоритма.

Недостатки:

1. Относительно низкая производительность (решается введение функции пересчета поиска центра при добавлении 1 объекта внутрь сферы).
2. Плохая применимость алгоритма при плохой разделимости выборки на кластеры.
3. Неустойчивость алгоритма (зависимость от выбора начального объекта)
4. Необходимость априорных знаний о ширине (диаметре) кластеров.

# Приложение, реализующее алгоритмы кластеризации

## Краткое описание приложения

Основное приложение, с помощью которого был проведён анализ работы некоторых алгоритмов кластерного анализа, написано в среде обычного браузера с помощью его стандартных технологий, таких как HTML, CSS и JavaScript.

Данные представлены в виде точек на двумерной плоскости. Это сделано по следующим причинам:

* Такое представление данных очень лекго проанализировать визуально и дать оценку правильности (или «естественности») полученной кластеризации.
* Простота такого представления помогает сосредоточиться только на правильности работы алгоритма и его анализе, не вдаваясь в то, каким образом получены данные, насколько они валидны и прочее.
* Количественные характеристики объекта можно представить в относительной шкале, таким образом, такое представление данных часто можно применить на практике.
* Представление выбрано двумерным, а не трёхмерным по причине того, что отрисовка в трёхмерном пространстве является более тяжёлой задачей для браузера, чем в двумерном, так как наиболее ресурсоёмкая компонента браузера – объектная модель документа, о которой будет сказано ниже. В результате чего, сложность и время работы алгоритмов серьёзно увеличатся из-за незначительных для самой кластеризации аспектов.

Приложение позволяет выполнить две основные задачи: узнать и проанализировать время, за которое алгоритм выполнил кластеризацию, и визуально оценить естественность и правильность его работы.

Приложение включает в себя поле, в котором пользователь может настраивать количество точек (объектов или векторов), не прибегая к обновлению страницы.

В нижней части находятся поля ввода для настройки радиуса точек и их количества, а также кнопка обновления поля.

В верхней части программы находится панель управления алгоритмами. Она включает в себя:

* Список доступных алгоритмов (в данном приложении – иерархический алгоритм, метод k-means нечёткой кластеризации и алгоритм семейства FOREL).
* Поля ввода для параметров алгоритма (в частности, пользователь может ввести количество кластеров или их радиус в зависимости от алгоритма).
* Кнопка запуска алгоритма.
* Поле, отображающее время, за которое алгоритм завершил процесс кластеризации (в миллисекундах).

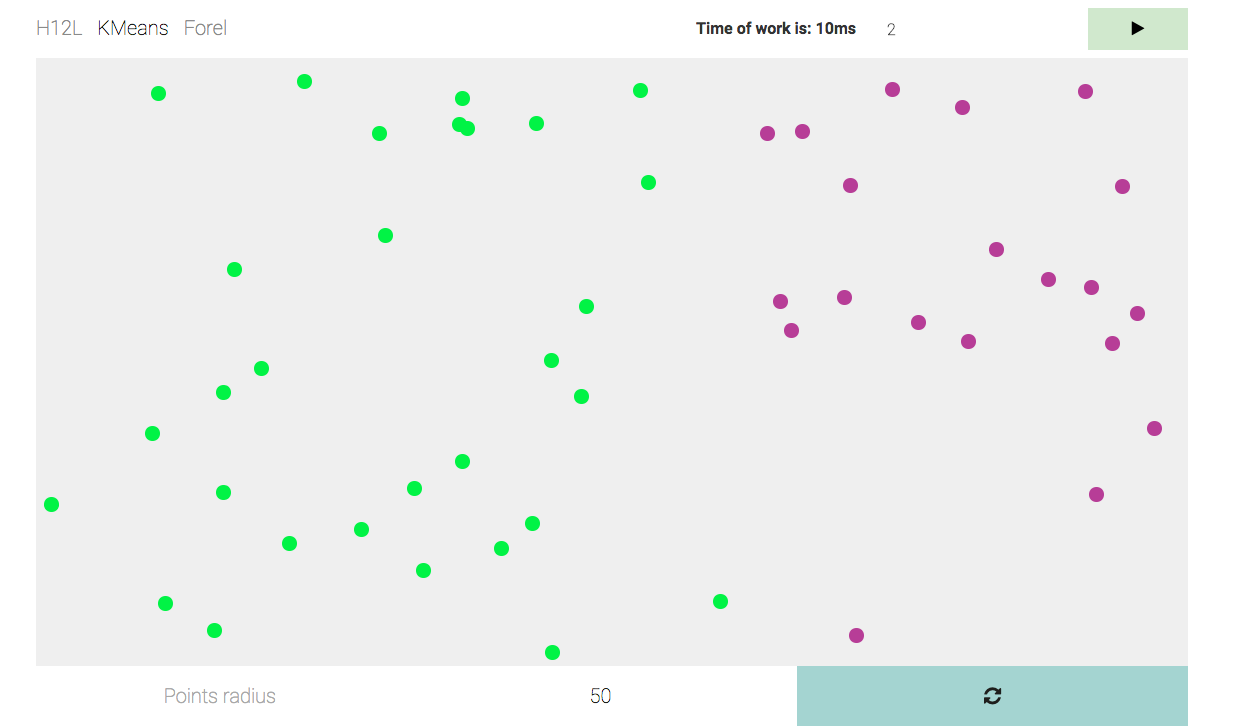


Рисунок 3.1. Приложение, реализующее анализ алгоритмов кластеризации

## Основные технологии, используемые в приложении

### HTML

HTML – стандартный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами, полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Язык HTML является приложением SGML (стандартного обобщённого языка разметки) и соответствует международному стандарту ISO 8879.

Язык XHTML является более строгим вариантом HTML, он следует всем ограничениям XML и, фактически, XHTML можно воспринимать как приложение языка XML к области разметки гипертекста.

Во всемирной паутине HTML-страницы, как правило, передаются браузерам от сервера по протоколам HTTP или HTTPS, в виде простого текста или с использованием шифрования.

Язык HTML был разработан британским учёным Тимом Бернерсом-Ли приблизительно в 1986—1991 годах в стенах ЦЕРНа в Женеве в Швейцарии. HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области вёрстки. HTML успешно справлялся с проблемой сложности SGML путём определения небольшого набора структурных и семантических элементов – дескрипторов. Дескрипторы также часто называют «тегами». С помощью HTML можно легко создать относительно простой, но красиво оформленный документ. Помимо упрощения структуры документа, в HTML внесена поддержка гипертекста. Мультимедийные возможности были добавлены позже.

Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения). В идеале, текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащённостью (цветной экран современного компьютера, монохромный экран органайзера, ограниченный по размерам экран мобильного телефона или устройства и программы голосового воспроизведения текстов). Однако современное применение HTML очень далеко от его изначальной задачи. Например, тег <TABLE> предназначен для создания в документах таблиц, но часто используется и для оформления размещения элементов на странице. С течением времени основная идея платформонезависимости языка HTML была принесена в жертву современным потребностям в мультимедийном и графическом оформлении.

### CSS

CSS – формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

CSS используется создателями веб-страниц для задания цветов, шрифтов, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими шрифт Брайля.

До появления CSS оформление веб-страниц осуществлялось исключительно средствами HTML, непосредственно внутри содержимого документа. Однако с появлением CSS стало возможным принципиальное разделение содержания и представления документа. За счёт этого нововведения стало возможным лёгкое применение единого стиля оформления для массы схожих документов, а также быстрое изменение этого оформления.

Преимущества:

* Несколько дизайнов страницы для разных устройств просмотра. Например, на экране дизайн будет рассчитан на большую ширину, во время печати меню не будет выводиться, а на КПК и сотовом телефоне меню будет следовать за содержимым.
* Уменьшение времени загрузки страниц сайта за счет переноса правил представления данных в отдельный CSS-файл. В этом случае браузер загружает только структуру документа и данные, хранимые на странице, а представление этих данных загружается браузером только один раз и может быть закэшировано.
* Простота последующего изменения дизайна. Не нужно править каждую страницу, а лишь изменить CSS-файл.
* Дополнительные возможности оформления. Например, с помощью CSS-вёрстки можно сделать блок текста, который остальной текст будет обтекать (например, для меню) или сделать так, чтобы меню было всегда видно при прокрутке страницы.

Недостатки:

* Различное отображение вёрстки в различных браузерах (особенно устаревших), которые по-разному интерпретируют одни и те же данные CSS.
* Часто встречающаяся необходимость на практике исправлять не только один CSS-файл, но и теги HTML, которые сложным и ненаглядным способом связаны с селекторами CSS, что иногда сводит на нет простоту применения единых файлов стилей и значительно удлиняет время редактирования и тестирования.

### JavaScript

JavaScript – прототипно-ориентированный сценарный язык программирования, являющийся диалектом языка ECMAScript.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

#### Возможности языка

JavaScript является объектно-ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам — функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания, что придаёт языку дополнительную гибкость.

Несмотря на схожий с Си синтаксис, JavaScript по сравнению с языком Си имеет коренные отличия:

* объекты, с возможностью интроспекции;
* функции как объекты первого класса;
* автоматическое приведение типов;
* автоматическая сборка мусора;
* анонимные функции.

В языке отсутствуют такие полезные вещи, как:

* модульная система: JavaScript не предоставляет возможности управлять зависимостями и изоляцией областей видимости;
* стандартная библиотека: в частности, отсутствует интерфейс программирования приложений по работе с файловой системой, управлению потоками ввода-вывода, базовых типов для бинарных данных;
* стандартные интерфейсы к веб-серверам и базам данных;
* система управления пакетами, которая бы отслеживала зависимости и автоматически устанавливала их.

#### Структура языка

Структурно JavaScript можно представить в виде объединения трёх чётко различимых друг от друга частей:

* ядро (ECMAScript),
* объектная модель браузера (Browser Object Model или BOM),
* объектная модель документа (Document Object Model или DOM).

Если рассматривать JavaScript в отличных от браузера окружениях, то объектная модель браузера и объектная модель документа могут не поддерживаться.

Объектную модель документа иногда рассматривают как отдельную от JavaScript сущность, что согласуется с определением DOM как независимого от языка интерфейса документа. В противоположность этому ряд авторов находят BOM и DOM тесно взаимосвязанными.

#### Объектная модель браузера

Объектная модель браузера – браузер-специфичная часть языка, являющаяся прослойкой между ядром и объектной моделью документа. Основное предназначение объектной модели браузера — управление окнами браузера и обеспечение их взаимодействия. Каждое из окон браузера представляется объектом window, центральным объектом DOM. Объектная модель браузера на данный момент не стандартизирована, однако спецификация находится в разработке WHATWG и W3C.

Помимо управления окнами, в рамках объектной модели браузера, браузерами обычно обеспечивается поддержка следующих сущностей:

* управление фреймами,
* поддержка задержки в исполнении кода и зацикливания с задержкой,
* системные диалоги,
* управление адресом открытой страницы,
* управление информацией о браузере,
* управление информацией о параметрах монитора,
* ограниченное управление историей просмотра страниц,
* поддержка работы с HTTP cookie.

#### Объектная модель документа

Объектная модель документа — интерфейс программирования приложений для HTML и XML-документов. Согласно DOM, документ (например, веб-страница) может быть представлен в виде дерева объектов, обладающих рядом свойств, которые позволяют производить с ним различные манипуляции:

* генерация и добавление узлов,
* получение узлов,
* изменение узлов,
* изменение связей между узлами,
* удаление узлов.

## Архитектура приложения

Программа представляет собой single-page приложение, то есть такой веб-сервис, в котором используется только одна html-страница, а управлением и изменением логики и представления занимается связка JavaScript и CSS.

Входная точка для JavaScript в данном сервисе – main.js, который хранит в себе ссылки на две сущности singleton: Algorithms и Field (singleton – паттерн такого класса или фабрики классов в объектно-ориентированном программировании, который создаёт только один объект этого класса (классов)).

Объект Field отвечает за отображение данных, их обновление и ререндеринг (перерисовка DOM-элементов браузером), а также за контроль этими процессами. Он ссылается на сущность PointsFactory, которая, в свою очередь, ссылается на класс Point, объекты которого создаёт.



Рисунок 3.2. Класс Field

Points – это класс объектов (или векторов) данных, которые включают в себя ссылку на DOM-элемент, имеют координаты (атрибуты или характеристики), по которым проводится кластеризации, и некоторые методы управления. С Point также косвенно, через основной файл приложения, связаны сущности из Algorithms, но они не могут непосредственно изменять их, что сделано для более ясной и понятной архитектуры, где сущности максимально независимы друг от друга для того, чтобы не было затратно что-то в них менять. Реализация класса Point показана на рисунке ниже:



Рисунок 3.3. Класс Point

Объект Alghoritms выполняет основную работу и решает основные задачи данного приложения:

* Конфигурация, хранение и управления алгоритмами приложения.
* Отрисовка и построения DOM-дерева для элементов управления кластеризациями.
* Управление работой приложения, оттображение и вывод результатов.

Когда Algorithms передаёт управление собственно алгоритмам, он, кроме основных параметров, передаёт в качестве параметра ссылку на Field, с помощью которого алгоритмы и анализируют данные.

Классы алгоритмов также включают в себя объекты Cluster. Этот объект группирует несколько объектов Points, вычисляет центроиды, расстояние до других кластеров и точек, если это необходимо. Расстояния он вычисляет с помощью объектов pointsDistaces и clustersDistances, с помощью которых выбираются и конфигурируются необходимые виды дистанций.

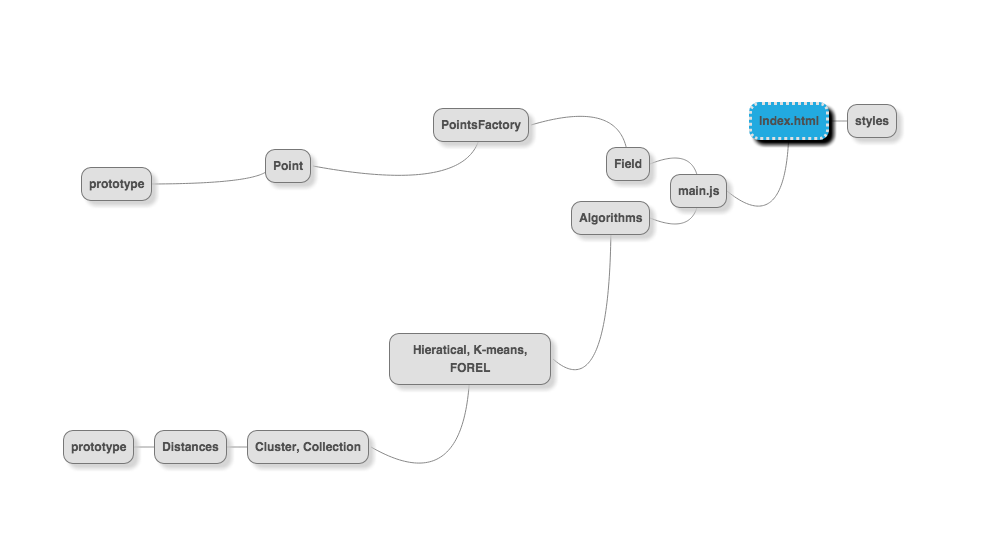


Рисунок 3.4. Архитектура приложения

Все объекты данной программы ссылаются также на методы prototype, которые включают в себя методы, помогающие в небольших задачах.



Рисунок 3.. Класс Cluster



Рисунок 3.. Конфигурация алгоримов

# Описание и анализ результатов работы программы

## Алгоритмы, реализованные в приложении

### Иерархический алгоритм кластеризации

Иерархический алгоритм кластеризации в качестве входного параметра, кроме объекта Field, который принимается всеми алгоритмами для работы с объектами данных, принимает количество групп или кластеров, на которое будет разделены исходные данные. Алгоритм реализует агломеративный метод кластеризации single-link, шаги которого более подробно описаны ниже:

1. Создаём начальную коллекцию кластеров и помещаем туда каждый объект данных по отдельности. В результате получается n кластеров, состоящих из одного элемента, где n – количество векторов.
2. Запускается метод findMinDistance, который находит номера кластеров, расстояние между которыми минимально (мера расстояния конфигурируется заранее), то есть находятся такие i и j матрицы подобия , что минимально среди всех значений этой матрицы. Метод findMinDistance представлен на рисунке ниже:

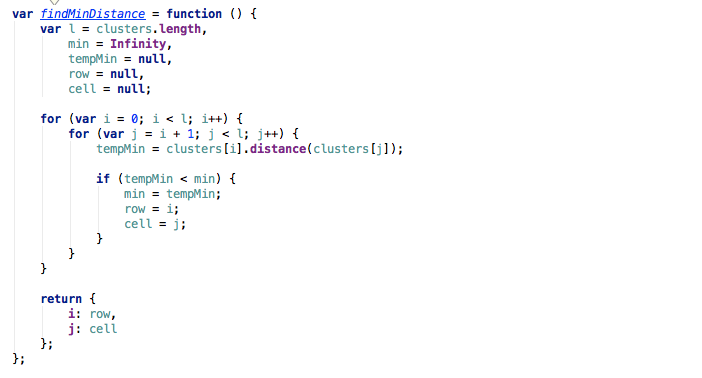


Рисунок .. Метод findMoreDistance

1. Элементы одного из найденных кластеров помещаем вместе с элементами другого. Первый кластер удаляем.



Рисунок .. Слияние двух кластеров

1. Повторяем шаги 2-3 до тех пор, пока число кластеров не больше заданного.

### Алгоритм кластеризации k-means

Так же как иерархический алгоритм кластеризации, метод k-means принимает в качестве аргументов количество кластеров, которое требуется получить в результате работы. Шаги алгоритма описаны ниже:

1. Создаём коллекцию кластеров, коллекцию временных групп, которые, возможно, будут изменяться на некоторой итерации, и коллекцию множество исходных объектов.
2. Выполняем функцию createCentorids. Она проверяет, были ли созданы до этого кластеры. Если нет, создаёт новые k кластеров без элементов и с центрами, которые равны координатам первых k точек. Если да, то создаёт кластеры, элементы которого равны элементам сформированных временных групп и центром, который равен среднему арифметическому координат точек из этих групп. На каждом шаге функция также «сбрасывает» временные группы.

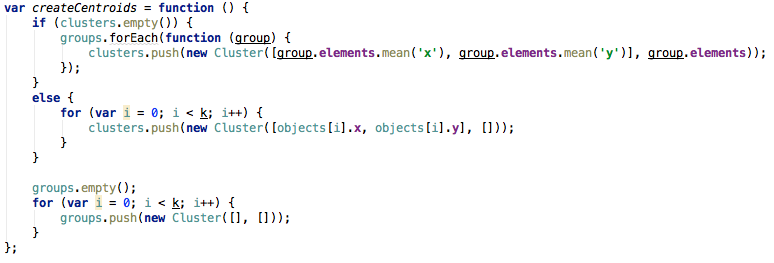


Рисунок 4.. Метод CreateCentroids

1. Выполняем функцию separate. Она распределяет объекты по временным кластерам (группам), основываясь на самом ближнем центроиде текущего разбиения.

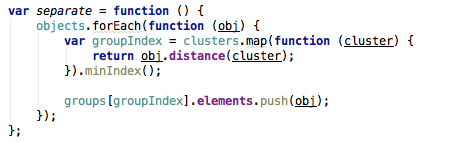
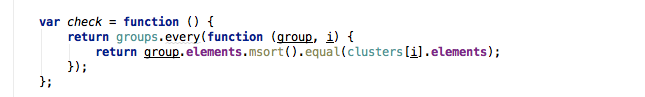


Рисунок .. Метод separate

1. Выполняем функцию check, которая проверяет, эквивалентны ли текущие кластеры временным. Если да, то алгоритм прекращает работу.



1. Выполняет шаги 2 – 5 до остановки алгоритма.



Рисунок .. Итерации алгоритма

### Алгоритм семейства FOREL

Алгоритм в качестве входных параметров принимает радиус, который будут иметь кластеры.

Шаги работы алгоритма:

1. Объявляем пустую коллекцию кластеров и ссылку на объеты данных.
2. Выбираем случайный объект, принимаем его за текущий центр. Находим все точки, на расстоянии меньше радиуса и находим новый центр, на основе их среднего. Повторяем это до тех пор, пока центр не перестанет изменяться.

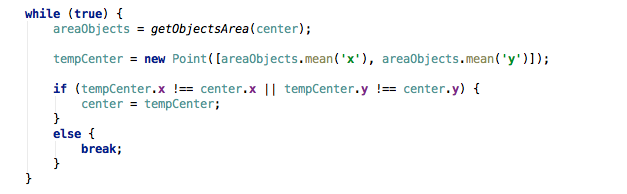


Рисунок .. Стабилизация локального центра

1. После нахождения устойчивого центра помещаем объекты, находящиеся в радиусе от него в кластер и больше их не рассматриваем. Повторяем до тех пор, пока не останется векторов.

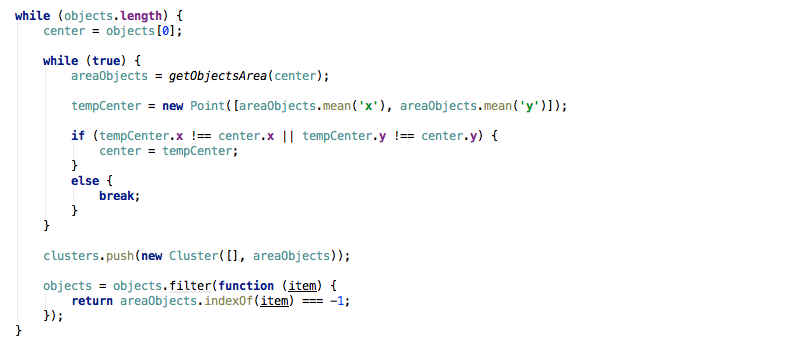


Рисунок .. Итерации алгоритма

## Наблюдения за работой алгоритмов

### Наблюдения за скоростью работы алгоритмов

Наблюдения за работой алгоритмов проводились с разным количеством объектов данных, в частности с 10, 20, 100, 1000 объектами. С каждым алгоритмом наблюдение повторялось 10 раз. Множества, над которыми проводились исследования, являлись эквивалентными для всех методов. Для иерархического алгоритма и для алгоритма k-means в качестве параметра (здесь параметр – количество итоговых кластеров) было передано значение 2. Для алгоритма семейства FOREL аргументом являлось значение 200 (так как для заданного поля иерархический алгоритм и алгоритм k-means разбивали множество на 2 кластера радиусом, который в среднем равен 200).

Ниже представлена таблица наблюдения за работой алгоритмов над 10 объектами.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер наблюдения | Время работы H12L алгоритма (мс) | Время работы K-means алгоритма (мс) | Время работы FOREL алгоритма (мс) |
| 1 | 10 | 5 | 3 |
| 2 | 8 | 2 | 3 |
| 3 | 6 | 3 | 3 |
| 4 | 8 | 3 | 2 |
| 5 | 8 | 2 | 2 |
| 6 | 6 | 3 | 3 |
| 7 | 15 | 3 | 2 |
| 8 | 7 | 2 | 2 |
| 9 | 6 | 3 | 2 |
| 10 | 5 | 2 | 2 |
| **Среднее** | **7.9** | **2.8** | **2.4** |

Таблица 4.1. Результат работы алгоритмов с 10 объектами данных

Результаты наблюдений показывают, что иерархический алгоритм кластеризации работает в среднем в 3 раза медленнее, чем частичные алгоритмы кластеризации. Алгоритмы FOREL и K-means работают примерно одинаково, разницу можно объяснить погрешностью.

Далее представлена таблица наблюдения за работой алгоритмов над 20 объектами.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер наблюдения | Время работы H12L алгоритма (мс) | Время работы K-means алгоритма (мс) | Время работы FOREL алгоритма (мс) |
| 1 | 30 | 5 | 3 |
| 2 | 27 | 5 | 3 |
| 3 | 26 | 5 | 4 |
| 4 | 29 | 4 | 3 |
| 5 | 24 | 4 | 4 |
| 6 | 30 | 4 | 3 |
| 7 | 22 | 5 | 4 |
| 8 | 25 | 4 | 4 |
| 9 | 25 | 4 | 5 |
| 10 | 24 | 4 | 4 |
| **Среднее** | **26.2** | **4.4** | **3.7** |

Таблица 4.2. Результат работы алгоритмов с 50 объектами данных

Алгоритм иерархической кластеризации на таком количестве векторов работает в среднем в 5 раз медленнее, чем алгоритмы нечёткой кластеризации. Алгоритмы нечёткой кластеризации отличаются теперь более существенно.

Далее представлена таблица наблюдения за работой алгоритмов над 100 объектами.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер наблюдения | Время работы H12L алгоритма (мс) | Время работы K-means алгоритма (мс) | Время работы FOREL алгоритма (мс) |
| 1 | 2888 | 19 | 15 |
| 2 | 2947 | 16 | 12 |
| 3 | 3244 | 14 | 13 |
| 4 | 3029 | 14 | 13 |
| 5 | 3551 | 14 | 12 |
| 6 | 3270 | 17 | 12 |
| 7 | 3253 | 15 | 13 |
| 8 | 3515 | 16 | 11 |
| 9 | 3506 | 15 | 11 |
| 10 | 3345 | 14 | 11 |
| **Среднее** | **3255** | **15.4** | **12.3** |

Таблица 4.3. Результат работы алгоритмов с 100 объектами данных

Время работы алгоритма иерархической кластеризации в среднем в 200 раз больше, чем у других алгоритмов.

При работе с количеством точек равном 1000, время работы иерархического алгоритма стало чрезмерно большим, то есть не достаточно малым для удобной и корректной работы данной программы. Поэтому ниже представлены результаты только для алгоритмов нечёткой кластеризации.

|  |  |  |
| --- | --- | --- |
| Номер наблюдения | Время работы K-means алгоритма (мс) | Время работы FOREL алгоритма (мс) |
| 1 | 200 | 254 |
| 2 | 194 | 150 |
| 3 | 158 | 169 |
| 4 | 179 | 162 |
| 5 | 169 | 187 |
| 6 | 200 | 171 |
| 7 | 175 | 175 |
| 8 | 177 | 185 |
| 9 | 187 | 175 |
| 10 | 191 | 188 |
| **Среднее** | **183** | **181** |

Таблица 4.4. Результат работы алгоритмов с 1000 объектами данных

Как видно, время работы алгоритмов примерно равно.

В результате исследований обнаружилось, что иерархический алгоритм кластеризации колоссально проигрывает в производительности алгоритмам нечёткой кластеризации. Эта разница экспоненциально растёт с увеличением числа исследуемых точек. Более того, алгоритм иерархической кластеризации основан на методе single-link, который требует меньшего числа процедур для построения новой кластеризации. Также стоит отметить, что этот алгоритм не строит граничный граф, а просто находит два ближайших кластера и объединяет их в один. Таким образом, если бы эти перечисленные факторы имели место быть, разница была бы ещё большей.

Алгоритмы нечёткой кластеризации напротив, работают весьма быстро, более того, при работе с большим количеством объектов, более большой процент объёма работы занимает работа с отрисовкой объектов браузером. Разница между скоростями работой алгоритмов немного отличается на малом количестве точек, и почти не отличается при большом их количестве. Это связано с немного разными методами, используемыми в работе алгоритмов, а также с погрешностью вычислительного окружения.

### Визуальное наблюдение за работой алгоритмов

В этой части работа алгоритмов будет оцениваться визуально, и это оценка будет нести субъективный характер, так как зависит от некоторых внутренних качеств исследователя. Оцениваться будет «правильность» и «естественность» полученного разбиения.

На каждом шаге наблюдения, алгоритмы будут работать на одинаковых множествах, каждое из которых имеет свои индивидуальные качественные и количественные характеристики.

Первый шаг исследования – оценить работу алгоритма на множестве из 10 точек, которое хорошо разделено. Это множество показано ниже:

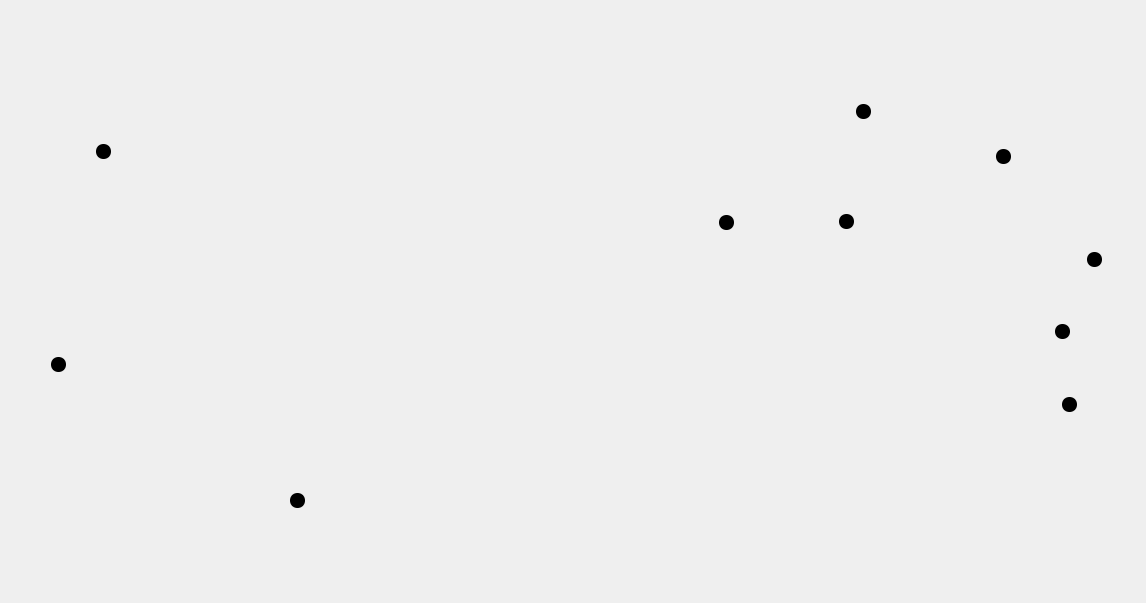


Рисунок 4.8. Исходное множество из 10 объектов

Результат работы иерархического алгоритма с параметром 2:



Рисунок 4.9. Результат работы иерархического алгоритма

Результат работы алгоритма k-means с параметром 2:



Рисунок 4.10. Результат работы K-means

Результат работы FOREL с параметром 300 (радиус в 300 пикселей выбран в виду того, что оба кластера примерно такого радиуса в данном масштабе):

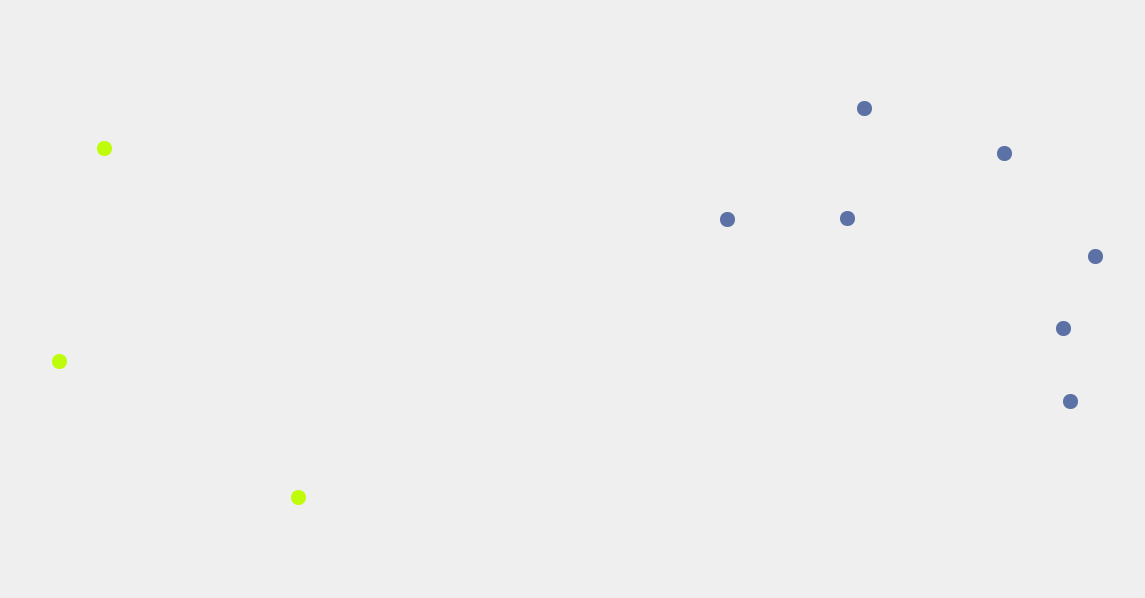


Рисунок 4.11. Результат работы FOREL

Следующий этап – проверить работу алгоритмов на 50 векторах, в которых кластеры «соприкасаются». Исходное множество имеет следующий вид:

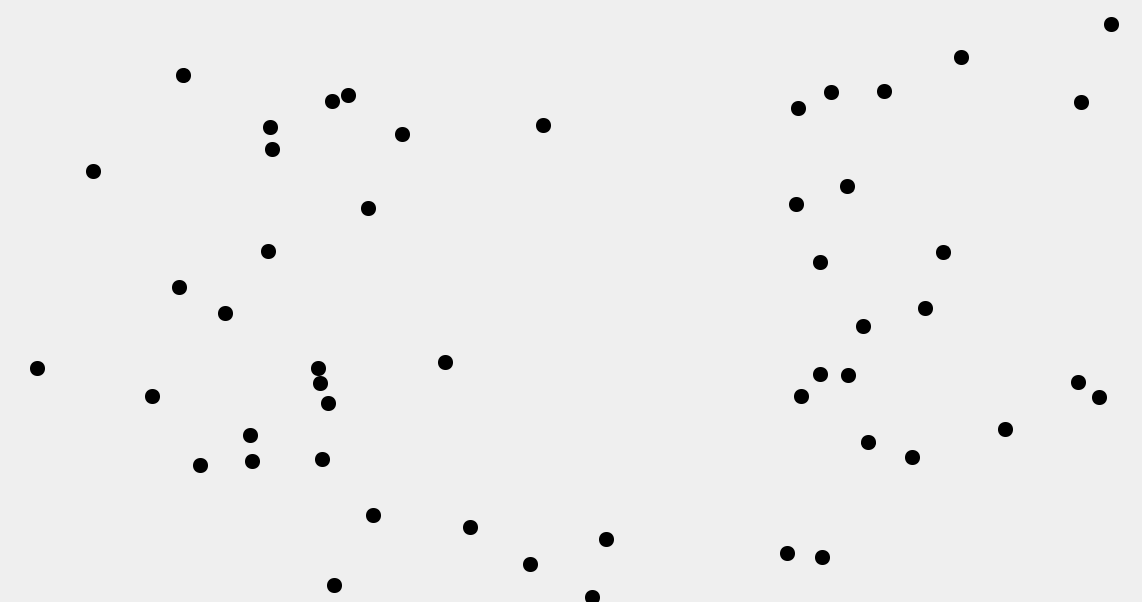


Рисунок 4.12. Исходное множество из 50 векторов

Результат работы иерархического алгоритма с параметром 2:

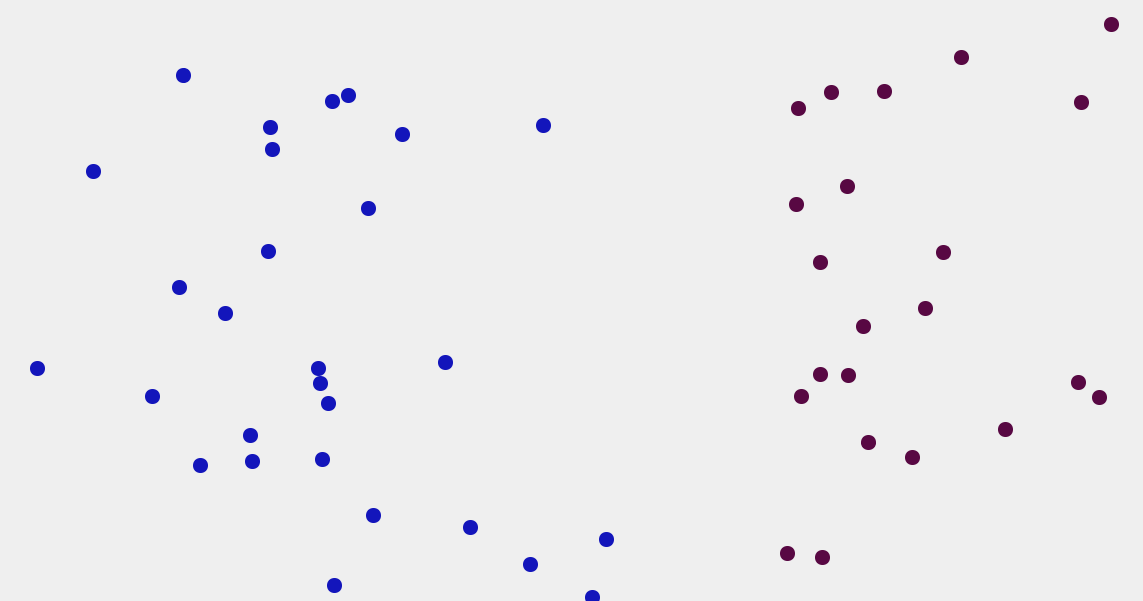


Рисунок 4.13. Результат работы иерархического алгоритма

Результат работы алгоритма k-means с параметром 2:

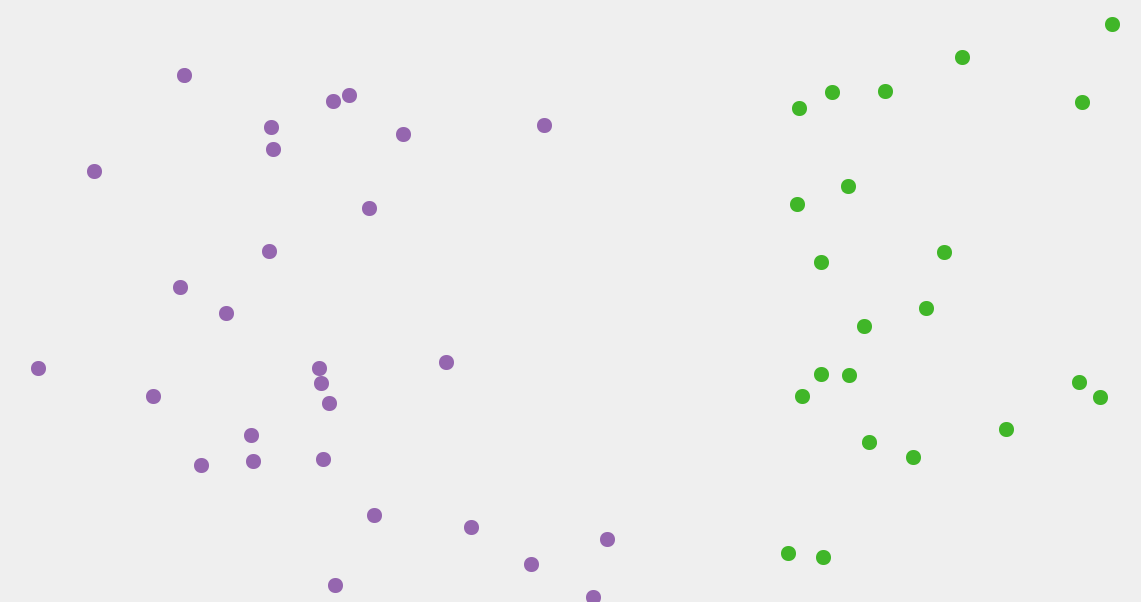


Рисунок 4.14. Результат работы k-means

В этом множестве хуже всего себя проявил алгоритм FOREL, который составлял кластеризации, подобные представленной ниже. Это показывает, что данный алгоритм лучше всего применять для хорошо разделённого множества.



Рисунок 4.15. Результат работы FOREL

В результате дальнейших наблюдений выяснилось, что иерархический алгоритм кластеризации ввиду природы метода single-link чаще, чем алгоритм частичной кластеризации строил «неправильные» или «неестественные» кластеризации из-за изолированных объектов или соприкасаемости кластеров, хотя изолированные точки сильно влияли также и на работу метода k-means.

В результате наблюдения выяснилось, что алгоритм FOREL наиболее подвержен ошибкам при примерно равной плотности точек на всей поверхности. Таким образом, если есть хоть малая возможность того, что точки не будут хорошо разделены, этот алгоритм следует исключить при выборе оптимального подхода. Для иерархического алгоритма лучше использовать метод complete-link, так как он создаёт кластеры, проверяя все отношения между точками. Наиболее предсказуемым был результат у алгоритма k-means, хотя этот метод создавал неправильные разделения, если имелась изолированная точка.

# Заключение

В этой дипломной работе были рассмотрены и проанализированы основные алгоритмы, методы, аспекты и проблемы кластеризации. Всё это было сделано на основе широкого спектра теоретических материалов, взятых как у зарубежных, так и у отечественных источников. На основе этой информации была разработана унифицированная теоретическая система по основам кластерного анализа.

Теоретически также были выявлены некоторые проблемы, с которыми встречается кластерный анализ: вычислительная сложность, невозможность математически правильно и точно определить понятие кластера, влияние некоторых побочных факторов, например, существование изолированной точки или потеря некоторых данных, на правильность работы алгоритма кластеризации. Было определено несколько общих методик и практик для устранения такого рода проблем, произведено их подробное описание и анализ.

Также в этой работе были рассмотрены основные типы данных исходной информации, параметров, с которыми работает алгоритм, типы характеристик, шкал, а также некоторые среды, в которых выполняются эти методы.

Кроме этого, были проведены практические наблюдения и анализ некоторых наиболее распространённых алгоритмов кластеризации в среде Интернет-технологий. К примеру, было экспериментально выяснено, что наиболее подходящий алгоритм среди рассматриваемых для решения задач такого рода – алгоритм k-means нечёткой кластеризации. Также было подтверждено, что алгоритм семейства FOREL кластеризации работает не очень корректно на плохо разделённых данных.

Приложение, которое было написано в ходе дипломной работы, можно расширить и применить для некоторых практических задач, как, например, кластеризация географических объектов, а также для более комплексных и сложных задач.

# Литература

1. Algorithms for Clustering Data / Anil K. Jain, Richard C. Dubes – New Jersey, USA: Prentice-Hall, 1988.
2. Data Clustering. Algorithms and Applications / Charu C. Aggarwal, Chandan K. Reddy – New York, USA: CRC Press, 2014.
3. Data Clustering and Learning / Joachim M. Buhmann – Bonn, Germany: Bonn University, 2002.
4. Кластеризация данных / Александр Котов, Николай Красильников [Электронный ресурс], 2006.
5. Методы и модели анализа данных: OLAP и Data Mining / А.А. Барсегян, М.С. Куприянов, В.В. Степаненко, И.И. Холод – Спб.: БХВ-Петербург, 2007.
6. Data Clustering: A Review / A. K. Jain, M. N. Murty, P. J. Flynn – Michigan, Michigan State University, 2000.
7. Clustering Algorithms / Jure Leskovec and Anand Rajar – Stanford, USA: Stanford University, 2010.
8. Partitioning Data into Clusters – Wolfram Documentation [Электронный ресурс] – Режим доступа:

https://reference.wolfram.com/language/tutorial/PartitioningDataIntoClusters.html.

1. Сайт Wikipedia [Электронный ресурс] – Режим доступа: <http://www.wikipedia.org>.
2. Статьи сайта «ХабраХабр» по кластерному анализу [Электронный ресурс] – Режим доступа: <http://habrahabr.ru>.
3. Javascript – сильные стороны / Д. Крокфорд – СПб.: Питер, 2012.
4. Javascript – подробное руководство / Дю Флэнаган – СПб.: Символ-Плюс, 2008.