# Глава 1. Аналитический обзор

Задача кластеризации – частный случай задачи машинного обучения, в частности, обучения без учителя, которая сводится к разбиению имеющегося множества объектов данных на подмножества таким образом, что элементы одного подмножества существенно отличались по некоторому набору свойств от элементов всех других подмножеств. Объект данных обычно рассматривается как точка в многомерном метрическом пространстве, каждому измерению которого соответствует некоторое свойство (атрибут) объекта, а метрика – есть функция от значений данных свойств. От типов измерений этого пространства, которые могут быть как числовыми, так и категориальными, зависит выбор алгоритма кластеризации данных и используемая метрика. Этот выбор продиктован различиями в природе разных типов атрибутов.

* 1. Цели кластеризации
* Понимание данных путём выявления кластерной структуры. Разбиение выборки на группы схожих объектов позволяет упростить дальнейшую обработку данных и принятия решений, применяя к каждому кластеру свой метод анализа (стратегия «разделяй и властвуй»).
* Сжатие данных. Если исходная выборка избыточно большая, то можно сократить её, оставив по одному наиболее типичному представителю от каждого кластера.
* Обнаружение новизны. Выделяются нетипичные объекты, которые не удаётся присоединить ни к одному из кластеров.

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

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

Результатом таксономии является древообразная иерархическая структура. При этом каждый объект характеризуется перечислением всех кластеров, которым он принадлежит, обычно от крупного к мелкому.

Классическим примером таксономии на основе сходства является биноминальная номенклатура живых существ, предложенная Карлом Линнеем в середине XVIII века. Аналогичные систематизации строятся во многих областях знания, чтобы упорядочить информацию о большом количестве объектов.

* 1. Применение

### Биология и биоинформатика

* В области экологии кластеризация используется для выделения пространственных и временных сообществ организмов в однородных условиях;
* Кластерный анализ используется для группировки схожих геномных последовательностей в семейство генов, которые являются консервативными структурами для многих организмов и могут выполнять схожие функции;
* Кластеризация помогает автоматически определять генотипы по различным частям хромосом;
* Алгоритмы применяются для выделения небольшого числа групп генетических вариации человеческого генома.

### Медицина

* Используется в позитронно-эмиссионной томографии для автоматического выделения различных типов тканей на трехмерном изображении;
* Применяется для выявления шаблонов устойчивости к антибиотикам; для классификации антибиотиков по типу антибактериальной активности.

### Маркетинг

* Кластеризация широко используется при изучении рынка для обработки данных, полученных из различных опросов. Может применяться для выделения типичных групп покупателей, разделения рынка для создания персонализированных предложений, разработки новых линий продукции.

### Интернет

* Выделение групп людей на основе графа связей в социальных сетях;
* Повышение релевантности ответов на поисковые запросы путем группировки веб-сайтов по смысловым значениям поискового запроса.

### Компьютерные науки

* Кластеризация используется в сегментации изображений для определения границ и распознавания объектов;
* Кластерный анализ применяется для определения образовавшихся популяционных ниш в ходе работы эволюционных алгоритмов для улучшения параметров эволюции;
* Подбор рекомендаций для пользователя на основе предпочтений других пользователей в данном кластере;
* Определение аномалий путем построения кластеров и выявления неклассифицированных объектов.

## Алгоритмы классификации

### Алгоритмы иерархической кластеризации;

Среди алгоритмов иерархической кластеризации выделяются два основных типа: восходящие и нисходящие алгоритмы. Нисходящие алгоритмы работают по принципу «сверху-вниз»: в начале все объекты помещаются в один кластер, который затем разбивается на все более мелкие кластеры. Более распространены восходящие алгоритмы, которые в начале работы помещают каждый объект в отдельный кластер, а затем объединяют кластеры во все более крупные, пока все объекты выборки не будут содержаться в одном кластере. Таким образом строится система вложенных разбиений. Результаты таких алгоритмов обычно представляют в виде дерева – дендрограммы. Классический пример такого дерева – классификация животных и растений.

Для вычисления расстояний между кластерами чаще все пользуются двумя расстояниями: одиночной связью или полной связью (см. обзор мер расстояний между кластерами).

К недостатку иерархических алгоритмов можно отнести систему полных разбиений, которая может являться излишней в контексте решаемой задачи.

### Алгоритмы квадратичной ошибки

Задачу кластеризации можно рассматривать как построение оптимального разбиения объектов на группы. При этом оптимальность может быть определена как требование минимизации среднеквадратической ошибки разбиения:

где *cj* — «центр масс» кластера *j* (точка со средними значениями характеристик для данного кластера).

Алгоритмы квадратичной ошибки относятся к типу плоских алгоритмов. Самым распространенным алгоритмом этой категории является метод k-средних. Этот алгоритм строит заданное число кластеров, расположенных как можно дальше друг от друга. Работа алгоритма делится на несколько этапов:

1. Случайно выбрать *k* точек, являющихся начальными «центрами масс» кластеров.
2. Отнести каждый объект к кластеру с ближайшим «центром масс».
3. Пересчитать «центры масс» кластеров согласно их текущему составу.
4. Если критерий остановки алгоритма не удовлетворен, вернуться к п. 2.

В качестве критерия остановки работы алгоритма обычно выбирают минимальное изменение среднеквадратической ошибки. Так же возможно останавливать работу алгоритма, если на шаге 2 не было объектов, переместившихся из кластера в кластер.

К недостаткам данного алгоритма можно отнести необходимость задавать количество кластеров для разбиения.

### Нечеткие алгоритмы

Наиболее популярным алгоритмом нечеткой кластеризации является алгоритм c-средних (c-means). Он представляет собой модификацию метода k-средних. Шаги работы алгоритма:

1. Выбрать начальное нечеткое разбиение *n* объектов на *k* кластеров путем выбора матрицы принадлежности *U* размера *n x k*.
2. Используя матрицу U, найти значение критерия нечеткой ошибки:  
   https://habrastorage.org/getpro/habr/post_images/21c/c5d/72b/21cc5d72b0c9bee8ad0476798eb4093c.jpg,  
   где *ck* — «центр масс» нечеткого кластера *k*:  
   https://habrastorage.org/getpro/habr/post_images/5fb/811/641/5fb811641b4fa3c84ba2490811ff53ea.jpg.
3. Перегруппировать объекты с целью уменьшения этого значения критерия нечеткой ошибки.
4. Возвращаться в п. 2 до тех пор, пока изменения матрицы *U* не станут незначительными.

Этот алгоритм может не подойти, если заранее неизвестно число кластеров, либо необходимо однозначно отнести каждый объект к одному кластеру.

### Алгоритмы, основанные на теории графов

Суть таких алгоритмов заключается в том, что выборка объектов представляется в виде графа G=(V, E), вершинам которого соответствуют объекты, а ребра имеют вес, равный «расстоянию» между объектами. Достоинством графовых алгоритмов кластеризации являются наглядность, относительная простота реализации и возможность вносения различных усовершенствований, основанные на геометрических соображениях. Основными алгоритмам являются алгоритм выделения связных компонент, алгоритм построения минимального покрывающего (остовного) дерева и алгоритм послойной кластеризации.

### Алгоритм выделения связных компонент

В алгоритме выделения связных компонент задается входной параметр R и в графе удаляются все ребра, для которых «расстояния» больше R. Соединенными остаются только наиболее близкие пары объектов. Смысл алгоритма заключается в том, чтобы подобрать такое значение R, лежащее в диапазон всех «расстояний», при котором граф «развалится» на несколько связных компонент. Полученные компоненты и есть кластеры.

Для подбора параметра R обычно строится гистограмма распределений попарных расстояний. В задачах с хорошо выраженной кластерной структурой данных на гистограмме будет два пика – один соответствует внутрикластерным расстояниям, второй – межкластерным расстояния. Параметр R подбирается из зоны минимума между этими пиками. При этом управлять количеством кластеров при помощи порога расстояния довольно затруднительно.

### Алгоритм минимального покрывающего дерева

Алгоритм минимального покрывающего дерева сначала строит на графе минимальное покрывающее дерево, а затем последовательно удаляет ребра с наибольшим весом. На рисунке изображено минимальное покрывающее дерево, полученное для девяти объектов.



Путём удаления связи, помеченной CD, с длиной равной 6 единицам (ребро с максимальным расстоянием), получаем два кластера: {A, B, C} и {D, E, F, G, H, I}. Второй кластер в дальнейшем может быть разделён ещё на два кластера путём удаления ребра EF, которое имеет длину, равную 4,5 единицам.

### Послойная кластеризация

Алгоритм послойной кластеризации основан на выделении связных компонент графа на некотором уровне расстояний между объектами (вершинами). Уровень расстояния задается порогом расстояния *c*. Например, если расстояние между объектами https://habrastorage.org/getpro/habr/post_images/f67/e6a/707/f67e6a707dcac300ff90658e523b135f.jpg, то https://habrastorage.org/getpro/habr/post_images/e91/fe1/65a/e91fe165a15d6fa1981cbb51f0e7fff3.jpg.

Алгоритм послойной кластеризации формирует последовательность подграфов графа *G*, которые отражают иерархические связи между кластерами:

https://habrastorage.org/getpro/habr/post_images/5ec/cd0/7cc/5eccd07cc4e56b227092cbe66501a67f.jpg,  
  
где *Gt = (V, Et)* — граф на уровне *сt*,  
https://habrastorage.org/getpro/habr/post_images/35f/249/c62/35f249c626b8d0e118efc7f472beee48.jpg,  
*сt* – t-ый порог расстояния,  
m – количество уровней иерархии,  
*G0 = (V, o)*, o – пустое множество ребер графа, получаемое при *t0* = 1,  
*Gm = G*, то есть граф объектов без ограничений на расстояние (длину ребер графа), поскольку *tm* = 1.  
  
Посредством изменения порогов расстояния {*с0, …, сm*},   
где 0 = *с0* < *с1* < …< *сm* = 1, возможно контролировать глубину иерархии получаемых кластеров. Таким образом, алгоритм послойной кластеризации способен создавать как плоское разбиение данных, так и иерархическое.

## Сравнение алгоритмов

### Вычислительная сложность алгоритмов

| **Алгоритм кластеризации** | **Вычислительная сложность** |
| --- | --- |
| Иерархический | O(n2) |
| k-средних c-средних | O(nkl),  где k – число кластеров, l – число итераций |
| Выделение связных компонент | зависит от алгоритма |
| O(n2 log n) | Минимальное покрывающее дерево |
| Послойная кластеризация | O(max(n, m)), где m |

### Сравнительная таблица алгоритмов

| **Алгоритм кластеризации** | **Форма кластеров** | **Входные данные** | **Результаты** |
| --- | --- | --- | --- |
| Иерархический | Произвольная | Число кластеров или порог расстояния для усечения иерархии | Бинарное дерево кластеров |
| k-средних | Гиперсфера | Число кластеров | Центры кластеров |
| c-средних | Гиперсфера | Число кластеров, степень нечеткости | Центры кластеров, матрица принадлежности |
| Выделение связных компонент | Произвольная | Порог расстояния R | Древовидная структура кластеров |
| Минимальное покрывающее дерево | Произвольная | Число кластеров или порог расстояния для удаления ребер | Древовидная структура кластеров |
| Послойная кластеризация | Произвольная | Последовательность порогов расстояния | Древовидная структура кластеров с разными уровнями иерархии |

## Мера расстояний

Итак, как же определять «похожесть» объектов? Для начала нужно составить вектор характеристик для каждого объекта — как правило, это набор числовых значений, например, рост-вес человека. Однако существуют также алгоритмы, работающие с качественными (т.н. категорийными) характеристиками.

После того, как мы определили вектор характеристик, можно провести нормализацию, чтобы все компоненты давали одинаковый вклад при расчете «расстояния». В процессе нормализации все значения приводятся к некоторому диапазону, например, [-1, -1] или [0, 1].

Наконец, для каждой пары объектов измеряется «расстояние» между ними — степень похожести.

### Евклидово расстояние

Наиболее распространенная функция расстояния. Представляет собой геометрическим расстоянием в многомерном пространстве:

### Квадрат евклидова расстояния

Применяется для придания большего веса более отдаленным друг от друга объектам. Это расстояние вычисляется следующим образом:

### Расстояние городских кварталов

Это расстояние является средним разностей по координатам. В большинстве случаев эта мера расстояния приводит к таким же результатам, как и для обычного расстояния Евклида. Однако для этой меры влияние отдельных больших разностей (выбросов) уменьшается (т.к. они не возводятся в квадрат). Формула для расчета манхэттенского расстояния:

### Расстояние Чебышева

Это расстояние может оказаться полезным, когда нужно определить два объекта как «различные», если они различаются по какой-либо одной координате. Расстояние Чебышева вычисляется по формуле:

### Степенное расстояние

Применяется в случае, когда необходимо увеличить или уменьшить вес, относящийся к размерности, для которой соответствующие объекты сильно отличаются. Степенное расстояние вычисляется по следующей формуле:

где *r* и *p* – параметры, определяемые пользователем. Параметр *p* ответственен за постепенное взвешивание разностей по отдельным координатам, параметр *r* ответственен за прогрессивное взвешивание больших расстояний между объектами. Если оба параметра – *r* и *p* — равны двум, то это расстояние совпадает с расстоянием Евклида.

## Существующие программные средства

На данный момент существует не одно программное средство, реализующее различные алгоритмы кластеризации. Примером такого проекта является Apache Mahout. Apache Mahout ― это открытый проект Apache Software Foundation (ASF), основной целью которого является создание масштабируемых алгоритмов машинного обучения, которые предлагаются для бесплатного использования по лицензии Apache. Mahout содержит реализации кластеризации, категоризации, CF и эволюционного программирования. Более того, там, где это разумно, он использует библиотеку Apache Hadoop, что позволяет Mahout эффективно масштабироваться в облаке.

## Заключение

В данной главе были описаны алгоритмы кластеризации и показано их сравнение.

Из рассмотренных в первой главе алгоритмов был выбран алгоритм минимального покрывающего дерева, так как он имеет произвольную форму кластера, а в качестве критерия схожести используется мера TF-IDF, так как этот алгоритм был предложен в задании к выпускной работе.

В третьей главе будет более подробно описан алгоритм кластеризации с помощью минимального покрывающего дерева.