# Методы кластеризации

Кластеризация – задача разбиения множества на классы, называемые кластерами. Внутри каждого кластера должны оказаться похожие объекты, а объекты из разных кластеров должны быть как можно более разными. Кластеризация отличается от классификации тем, что нам заранее не известны классы, требуется их найти. В связи с этим задача неоднозначна и не имеет конкретного «правильного» решения. Не существует наилучшего критерия качества алгоритма, количество кластеров заранее обычно не известно, а выбор метрики («критерий схожести объектов») субъективен.

### Возможные цели кластеризации:

* Понять структуру множества объектов, увидеть зависимости.
* Подготовка для последующего анализа: разбить объекты на кластеры и работать с каждым кластером в дальнейшем по отдельности.
* Уменьшить количество объектов в выборке для дальнейшего анализа: оставить некоторое количество объектов из каждого кластера.

### Кластерные структуры:

* Расстояния между объектами в одном кластере меньше, чем между объектами из разных.



* Кластеры могут быть в виде лент, тогда предположение о межкластерных и внутрикластерных расстояниях не работает.



* Кластеры могут соединяться перемычками или быть наложены на шумовой фон из объектов, далеких от всех кластеров.



* Кластеры могут пересекаться.



* Кластеры могут иметь совершенно другие закономерности.



* Кластеров вообще может не быть.



### EM алгоритм

Метод основан на предположении, что в каждом кластере данные подчиняются нормальному закону распределения. А значит, все множество описывается линейной комбинацией многомерных распределений. Задача сводится к поиску математического ожидания и дисперсии для каждого распределения, поскольку нормальное распределение однозначно задается этими параметрами. А также необходимо найти коэффициенты плотностей в линейной комбинации.

Мерой качества модели является функция правдоподобия, максимизируя ее, находим оптимальные параметры распределений. Алгоритм подразумевает, что все объекты принадлежат всем кластерам, но с разными вероятностями. После нахождения оптимальных параметров, объект относится к кластеру, для которого вероятность принадлежности выше.

#### Особенности алгоритма:

* Количество кластеров должно быть известно заранее.
* Алгоритм устойчив к шумам и пропускам.
* Алгоритм быстро сходится при удачной инициализации параметров.
* Алгоритм может «застревать» в локальных оптимумах.
* Иногда предположение о нормальности распределения данных может быть ошибочным.
* Алгоритм неплохо работает на «перекрывающихся» кластерах.

#### Реализация на Python

* Реализация алгоритма c подробной документацией в библиотеке skicit-learn лежит здесь <http://scikit-learn.org/stable/modules/generated/sklearn.mixture.GMM.html>
* Импортируем алгоритм EM, данные про цветки ирис.
* Обозначим функцию, за число кластеров возьмем 3, как и настоящих классов.
* Обучим модель и предскажем значения кластеров для данных функцией fit\_predict
* Выведем на экран предсказанные значения, информацию о сходимости метода и средние значения кластеров( по каждому признаку).

**import** numpy **as** np

**import** pandas **as** pd

**from** sklearn **import** mixture, datasets

**import** sklearn **as** skl

g = mixture.GMM(n\_components=3)

data = skl.datasets.load\_iris()

aim = data.target

data = data.data

prediction = g.fit\_predict(data)

**print**(prediction)

**print**(g.converged\_)

**print**(g.means\_)

[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2

2 2 0 2 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 0 2 2 2 0 2 2 2 0 2 2 2 0 2

2 2]

True

[[ 5.90845874 2.74067981 4.36295651 1.38571005]

[ 5.006 3.418 1.464 0.244 ]

[ 6.74388131 3.05099112 5.6461753 2.07166896]]

### K-means

Алгоритм является частным случаем EM, однако, проще объяснить его другим способом.

Сначала инициализируются начальные приближения центров кластеров: k объектов становятся центрами кластеров. Остальные объекты относятся к кластеру, центр которого ближе. Далее пересчитываются центры кластеров по формуле центра масс. Объекты снова разбиваются на кластеры, в зависимости от того, какой центр ближе теперь. И так происходит пока центры не перестанут меняться.

#### Особенности алгоритма:

* Агоритм требует выбора количества кластеров, от этого часто зависит качество кластеризации, поэтому лучше пробовать несколько вариантов с разным количеством кластеров.
* Алгоритм чувствителен к начальному приближению: одним из оптимальных вариантов является выбор k наиболее удаленных друг от друга точек.

#### Реализация в Python:

* Реализация метода в пакете scikit-learn с подробной документацией лежит тут [http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html%23sklearn.cluster.KMeans)
* Загрузим необходимые пакеты, а также данные про цветки ириса.
* Обозначим кластеризатор, в качестве поиска начального приближения выберем функцю k-means++. Подробнее про нее можно прочитать здесь <https://ru.wikipedia.org/wiki/K-means%2B%2B>
* Обучим на данных и предскажем значения кластеров, выведем на экран предсказания, а также центры кластеров.

**import** numpy **as** np

**from** sklearn **import** cluster, datasets

**import** sklearn **as** skl

g = cluster.KMeans(n\_clusters=3, init='k-means++')

data = skl.datasets.load\_iris()

aim = data.target

data = data.data

prediction = g.fit\_predict(data)

**print**(prediction)

**print**(g.cluster\_centers\_)

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2

2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2 1 2

2 1]

[[ 5.006 3.418 1.464 0.244 ]

[ 5.9016129 2.7483871 4.39354839 1.43387097]

[ 6.85 3.07368421 5.74210526 2.07105263]]

### Спектральная кластеризация

Алгоритм основан на понижении размерности исходного пространства с помощью собственных векторов матрицы расстояний.

Работа алгоритма:

1. Строится матрица расстояний между объектами. (n объектов)
2. Матрица нормализуется и приводится к виду матрицы лапласиана (немного измененная матрица, но суть такая же).
3. Находятся собственные векторы полученной матрицы (n-мерные)
4. Теперь каждый собственный вектор является новым признаком. Если взять не все собственные вектора, то получим пространство меньшей размерности, чем исходное.
5. Далее в новом пространстве применяется стандартный метод вроде K-means.

Особенности алгоритма:

* Алгоритм не делает предположений о форме кластеров, поэтому хорошо работает даже если кластеры нестандартной формы.

#### Реализация на Python:

* Реализация метода в пакете scikit-learn с подробной документацией лежит тут [http://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html#sklearn.cluster.SpectralClustering](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html%23sklearn.cluster.SpectralClustering)
* Загрузим необходимые пакеты и данные про цветки ириса.
* Обозначим алгоритм и обучим его на данных.
* Выведем на экран предсказания классов для данных.

**import** numpy **as** np

**from** sklearn **import** cluster, datasets

**import** sklearn **as** skl

g = skl.cluster.SpectralClustering(n\_clusters=3)

data = skl.datasets.load\_iris()

aim = data.target

data = data.data

prediction = g.fit\_predict(data)

**print**(prediction)

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2

2 2 1 2 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2 1 2

2 1]

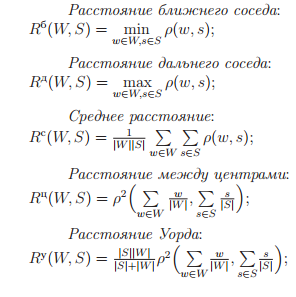
### Иерархическая кластеризация

Алгоритм строит системы вложенных разбиенией, которую легко визуализировать с помощью дендрограммы. Существуют дивизимные (нисходящие) алгоритмы, однако более популярными являются агломеративные. Сложность агломеративных методов *O*(*n3*) и даже *O*(*n2*), дивизивных *O*(*2т*).

Работает алгоритм следующим образом: сначала все объекты являются отдельными кластерами. Затем последовательно 2 ближайших кластера соединяются в один. Таким образом на последнем шаге мы получаем один кластер, состоящий из всех объектов.

Возникает вопрос: как понять какие кластеры близки, а какие нет?

Для этого существует несколько распространенных метрик:



Желательные свойства расстояния:

1. Монотонность.

Функция расстояния монотонна, если на каждом шаге расстояние между объединяемыми кластерами только увеличивается.

Это свойство позволяет нарисовать дендрограмму без самопересечений

Расстояние между центрами свойством монотонности НЕ обладает, в отличие от всех остальных.

1. Своство растяжения или сжатия.

Расстояние обладает свойством растяжения, если по мере роста кластера расстояния от него до других увеличивается. Свойство растяжения позволяет сделать более четкую кластеризацию, в то время как свойство сжатия делает кластеризацию более размытой.

Расстояния Уорда и дальнего соседа являются растягивающими, а расстояние ближнего соседа – сжимающим.

Среднее расстояние и расстояние между центрами ни одним, ни другим свойством не обладают.

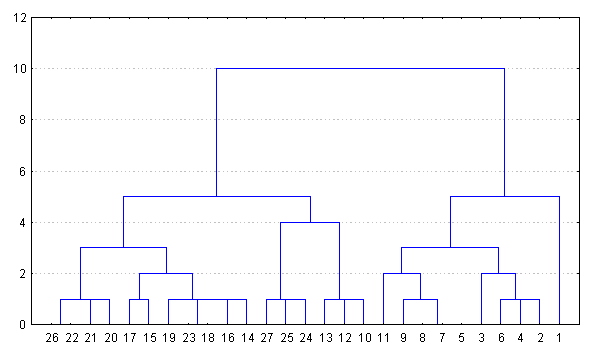
1. Редуктивность.

Расстояние R называется редуктивным, если для любого и любых близких кластеров и объединение окрестностей и содержит в себе окрестность кластера .

Расстояние между центрами свойством редуктовности НЕ обладает, в отличие от всех остальных.

Каждое из расстояний, перечисленных выше, имеет свои недостатки и подходит не для всех задач. Метод ближнего соседа обладает цепочечным эффектом, когда независимо от формы кластера к нему присоединяются ближайшие к границе объекты. В некоторых случаях это приводит к тому, что кластеры «отращивают щупальца». В зависимости от задачи это свойство может быть как полезным, так и мешающим. Метод ближнего соседа хорошо подходит для выделения кластеров ленточной формы. Метод дальнего соседа цепочечного эффекта не имеет, но на раннем этапе может объединять довольно несхожие группы. Расстояние между центрами масс не монотонно и не редуктивно, поэтому редко используется на практике, хотя интуитивно кажется «золотой серединой». Метод Уорда оказался наилучшим по результатам экспериментального сравнения. Он чаще других методов восстанавливает интуитивно наилучшую кластеризацию.

Выбор числа кластеров проще всего производить отсечением нескольких последних объединений на дендрограмме. По общему виду дендрограммы можно определить оптимальное количество кластеров. Например на картинке ниже кажется логичным удалить 2 последних объединения и получить 3 кластера.



#### Реализация на Python:

Код с использованием пакета scikit-learn

* Реализация метода в пакете scikit-learn с подробной документацией лежит тут <http://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html>
* Загрузим пакеты, данные про цветки ириса, обозначим кластеризатор.
* Выведем на экран предсказания для объектов.

**import** numpy **as** np

**from** sklearn **import** cluster, datasets

**import** sklearn **as** skl

g = skl.cluster.AgglomerativeClustering(n\_clusters=3, linkage='ward')

data = skl.datasets.load\_iris()

aim = data.target

data = data.data

prediction = g.fit\_predict(data)

prediction

array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 0,

2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2,

0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0], dtype=int64)

График дендрограммы с использованием пакета scipy

* Загрузим пакеты.
* Функцией linkage выполним кластеризацию. На выходе получаем матрицу «связей»
* Передаем эту матрицу функции dendrogram, которая строит дендрограмму.
* Строим график дендрограммы.

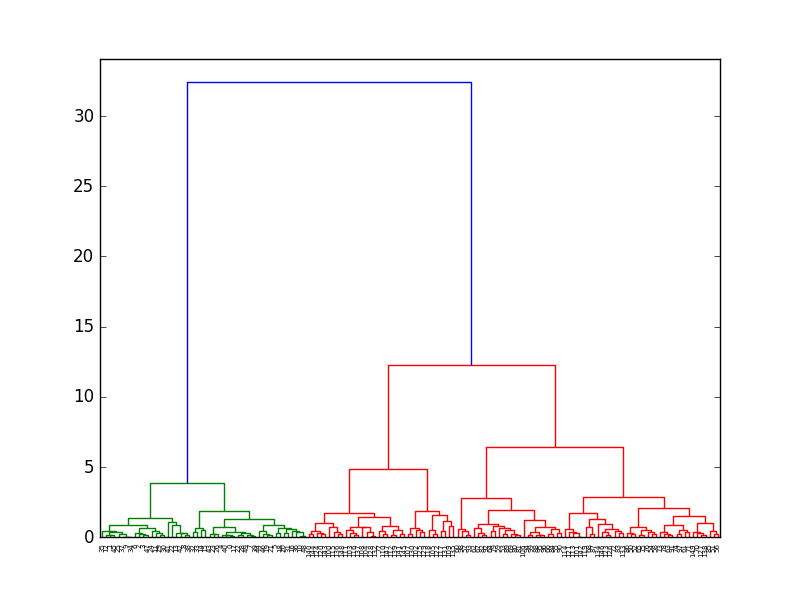
**import** scipy

**from** matplotlib **import** pyplot **as** plt

linkage = scipy.cluster.hierarchy.linkage(data, method='ward', metric='euclidean')

dendrogram = scipy.cluster.hierarchy.dendrogram(linkage)

plt.show(dendrogram)



### Графовые методы кластеризации.

Графавые методы кластеризации основаны на представлении выборки в виде графа, где вершинами являются объекты, а ребрами попарные расстояния между объектами.

### Метод связных компонент.

Метод заключается в удалении из графа всех ребер длины >R. То есть задача состоит в подборе такого числа R, чтобы исходный граф развалился на несколько связных компонент. Полученные подграфы и есть искомые кластеры.

### Метод кратчайшего незамкнутого пути (минимального дерева)

Метод заключается в построении минимального покрывающего дерева: свзяного графа с минимальной суммарной длиной ребер. Затем, последовательно удаляя из него самые длинные ребра, получаем несколько связных компонент, которые и будут кластерами.

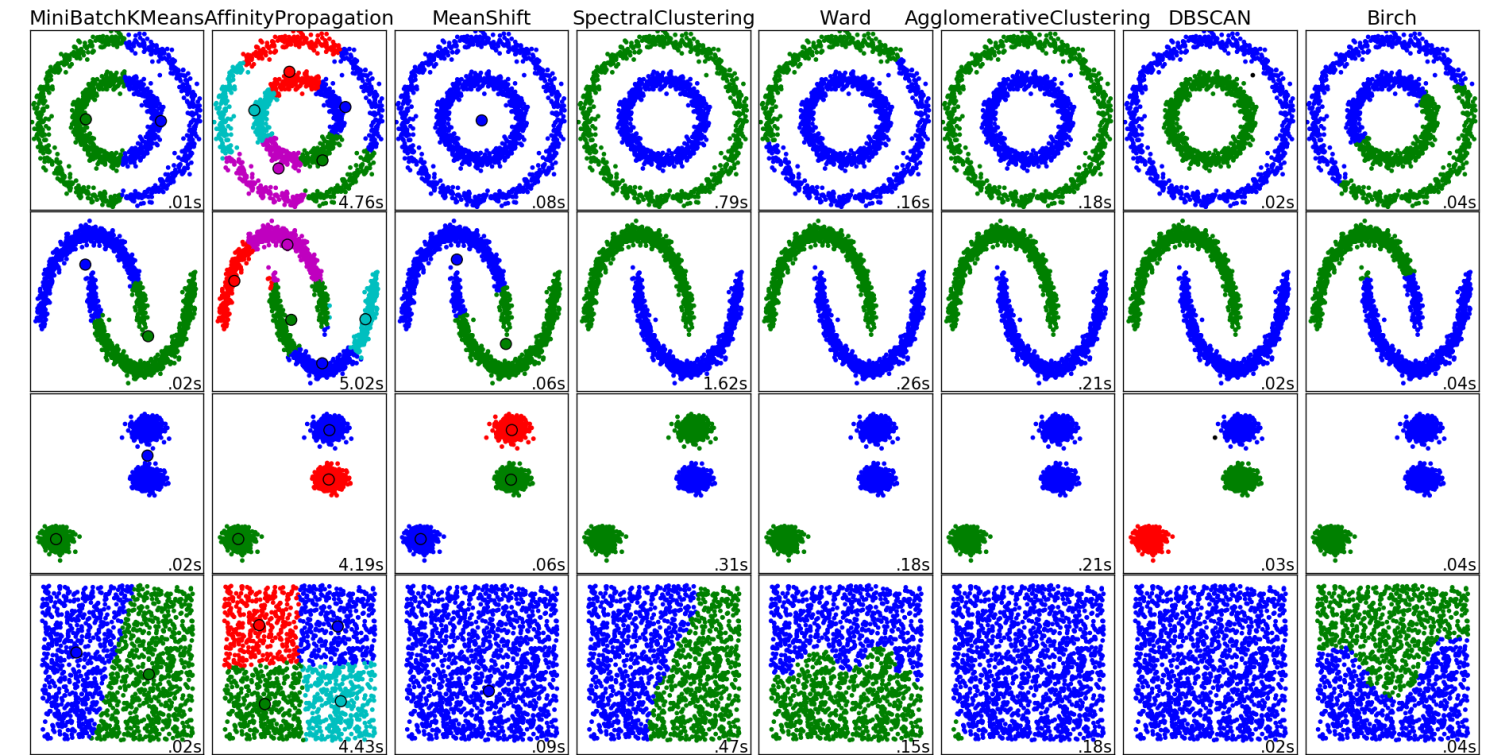
Ниже приведена картинка минимального покрывающего дерева для 9 объектов. Удалив ребро CD получили 2 кластера. Далее, удалив ребро EF можем получить 3 кластера. Простой, но медленный алгоритм *O*(*n3*), также чувствительный к шуму.



### Методы оценивания качества кластеризации.

Оптимальных критериев качества кластеризации не существует. Поскольку нет оптимального примера разбиения на кластеры, как в задачах классификации, задача кластеризации сводится к некоторой импровизации. Однако, данные можно визуализировать: например применить метод понижения размерности до 2-3 измерений и построить в этих измерениях график. Оценить визуально насколько оптимально алгоритм отделил близкие точки. Второй способ – попробовать найти закономерность в выделенных алгоритмом кластерах и оценить ее логичность.

#### Визуализация работы алгоритмов из scikit-learn.



#### Материал был составлен на основе статей, рекомендуемых к прочтению:

<http://www.ccas.ru/voron/download/Clustering.pdf>

<https://habrahabr.ru/post/101338/>

### Метод DBSCAN Density-Based Spatial Clustering of Applications with Noise

Может справляться с ленточными классами любой формы

Сложность *O*(*n*\*log*n*)

Устойчив к шуму.

Требует подбора *ε-*окрестности и числа объектов *M* в кластере.

### Советы

Иерархическая кластеризация – когда нужна таксономия (знать отношения вложенности)

k-means, когда есть оценка числа кластеров, кластеры сферические. Проводится ряд испытаний с разным кол-ом кластеров.

DBSCAN – кластеры ленточные или линейно неразделимы или есть сильный шум.