Урок 1

Введение в кластеризацию

1.1. Задача кластеризации

1.1.1. Обучение на размеченных данных (напоминание)

В задачах обучения на размеченных данных, которые рассматривались в прошлом курсе, существовала некоторая целевая зависимость:

$$x \mapsto y,$$

где y — значение прогнозируемой величины (в случае задачи регрессии) или метка класса (в случае задачи классификации). По обучающей выборке, которая состоит из множества объектов $x_1, ..., x_\ell$ и ответов на них $y_1, ..., y_\ell$, требовалось спрогнозировать ответы для объектов $x_{l+1}, \ldots, x_{l+u}$ тестовой выборки. Фактически, ставилась задача приблизить целевую зависимость некоторой функцией $a(x)$:

$$a(x) \approx y.$$

1.1.2. Задача кластеризации

В задаче кластеризации обучающая выборка $x_1, ..., x_l$ состоит только из объектов, но не содержит ответы на них, а также одновременно является и тестовой выборкой. Требуется расставить метки $y_1, ..., y_l$ таким образом, чтобы похожие друг на друга объекты имели одинаковую метку, то есть разбить все объекты на некоторое количество групп.

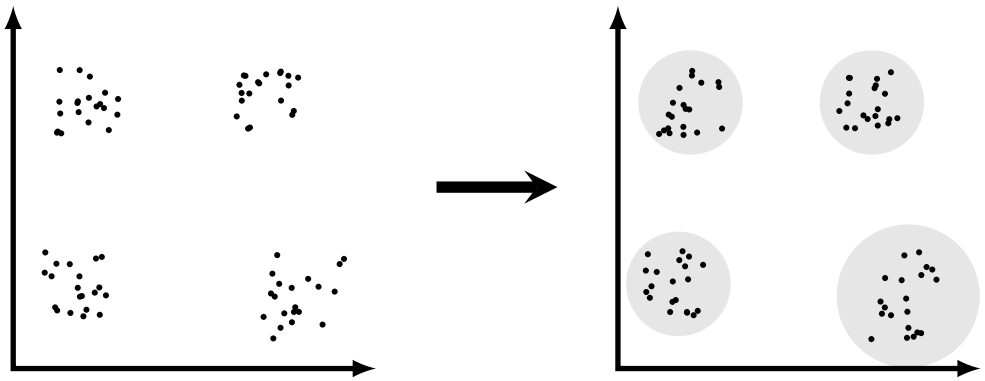


Рис. 1.1: Пример задачи кластеризации

Задачу кластеризации также можно представить как восстановление отображения $x \mapsto y$.

1.1.3. Метрики качества в задаче классификации

Для измерения качества кластеризации требуется ввести так называемые метрики качества. Пусть F_1 — среднее расстояние между объектами в кластерах (должно быть как можно меньше), а F_2 — среднее расстояние

между объектами из разных кластеров (должно быть как можно больше):

$$F_0 = \frac{\sum_{i<j}[y_i = y_j]\rho(x_i, x_j)}{\sum_{i<j}[y_i = y_j]} \quad \to \quad \min, \qquad F_1 = \frac{\sum_{i<j}[y_i \neq y_j]\rho(x_i, x_j)}{\sum_{i<j}[y_i \neq y_j]} \quad \to \quad \max.$$

Очевидно, что, при оптимизации только одной из метрик F_0 или F_1, учтен будет всего лишь один из факторов. Учесть сразу оба можно, например, следующим образом:

$$\frac{F_0}{F_1} \quad \to \quad \min.$$

1.2. Примеры задач кластеризации

В зависимости от специфики задачи и особенностей используемых в задаче данных задачи кластеризации могут сильно отличаться. Может, например, отличаться форма, размер и иерархия кластеров, а также тип задачи (основная или побочная) и тип классификации (жесткая или мягкая).

1.2.1. Различные формы кластеров

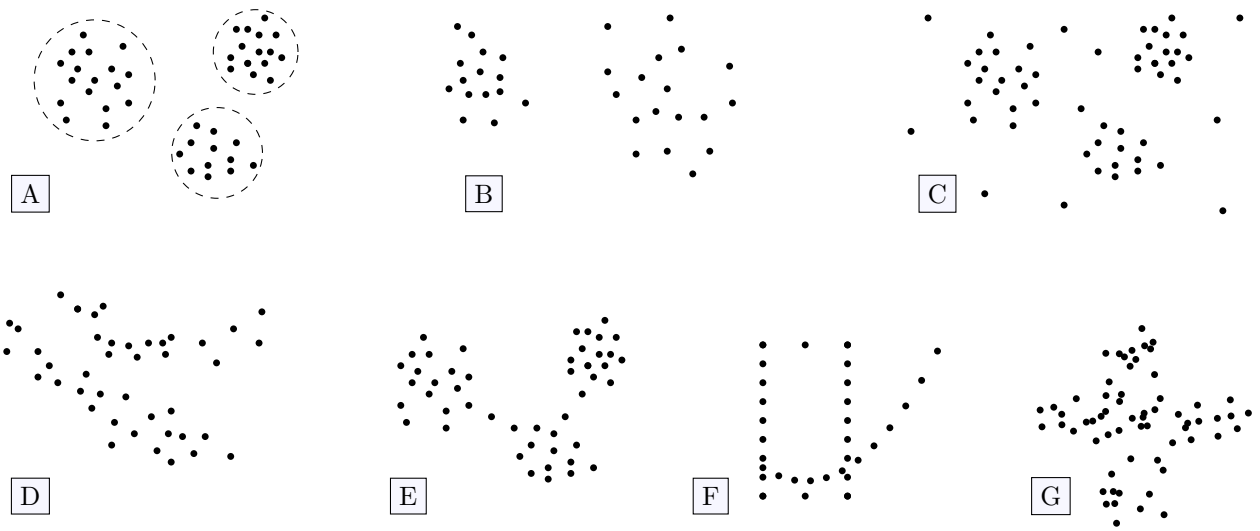


Рис. 1.2: Примеры различных форм кластеров

Кластеры могут иметь совершенно различную форму:

- **A** В простейшем случае кластеры представляют собой «сгустки точек» и могут быть легко выделены окружностью.
- **B** Иногда кластеры принимают более сложную форму, но все также могут быть легко выделены.
- **C** Возможен случай, когда между кластерами есть такие точки, которые сложно отнести к определенному кластеру.
- **D** Кластеры могут представлять собой вытянутые ленты. В этом случае можно найти пару точек из разных кластеров, которые находятся ближе некоторой пары из одного кластера. Такие кластеры можно выделить, добавляя к строящемуся кластеру близжайшую точку.
- **E** Кластеры могут плавно перетекать друг в друга. В этом случае описанная для предыдущего случая стратегия уже не работает.
- **F** Кластеры могут быть образованы по некоторому закону, который все же не известен.
- **G** Кластеры могут пересекаться. В этом случае достаточно сложно определить, к какому кластеру относятся некоторые объекты.

Также кластеров может вовсе не быть.

1.2.2. Иерархия и размер кластеров

В практических задачах бывает нужно, чтобы внутри какого-нибудь большого кластера были кластеры меньшего размера. Например, в задаче кластеризации статей с сайта «habrahabr.ru», кластер «IT» может включать в себя кластер «Алгоритмы», внутри которого могут быть кластеры «Методы машинного обучения» и «Алгоритмы и структуры данных».

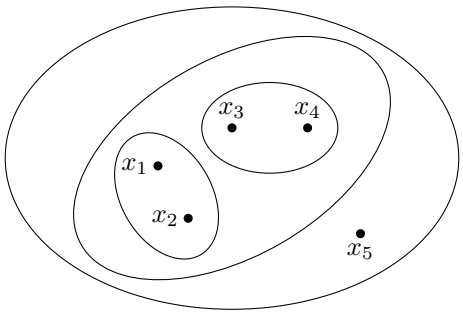


Рис. 1.3: Пример возможной иерархии кластеров.

Также может различаться размер кластеров. Так, при кластеризации новостей по содержанию возможны следующие 3 варианта постановки задачи, которые отличаются размерами выделяемых кластеров:

- В один и тот же кластер попадают новости на одну тему. Например, в первый кластер попадут все спортивные новости и так далее.
- В один и тот же кластер попадают новости об одном и том же большом событии. Например, в один кластер попадут все новости про олимпиаду в городе Сочи.
- В один и тот же кластер попадают тексты об одной и той же конкретной новости, например про открытие олимпиады в сочи.

1.2.3. Основная или вспомогательная задачи

В зависимости от цели задача кластеризации может быть:

- **Основной**, когда полученная кластеризация не используется для решения какой–либо другой задачи.
- **Вспомогательной**, если задача классификации является промежуточной при решении другой задачи.

Рассмотренная выше задача кластеризации новостей является основной задачей.

Задача сегментации целевой аудитории появляется, если для покупателей из разных групп требуется использовать разную маркетинговую кампанию.

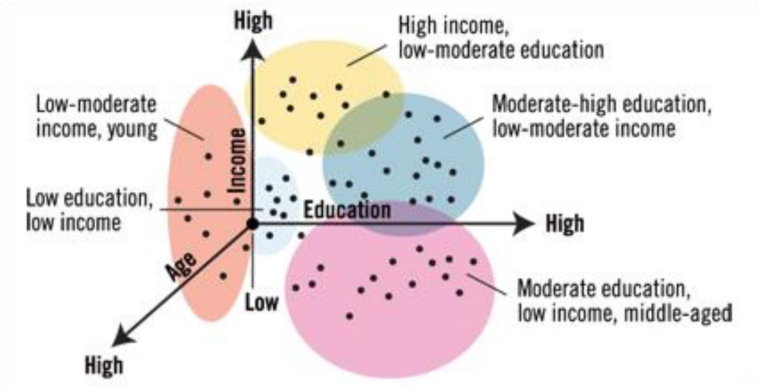


Рис. 1.4: Задача сегментации целевой аудитории

Если для покупателей из разных групп требуется предлагать различный товар, то в этом случае задача кластеризации является вспомогательной по отношению к задаче рекомендации. Если задача сегментации решается для целей аналикити, то она, в свою очередь, будет основной.

Другой пример вспомогательной задачи встречается при распознавании символов. Каждый символ может быть написан множеством различных способов, то есть иметь различные начертания: жирный, курсив и т.д.

Рис. 1.5: Различные начертания символа «5»

В этом случае полезно сначала решить вспомогательную задачу кластеризации символов по их начертанию, а затем уже использовать полученную информацию для распознавания самих символов.

1.2.4. Жёсткая или мягкая кластеризация

В зависимости от того, может ли относиться объект сразу к нескольким кластерам бывает:

- **Жесткая кластеризация**: объект может быть отнесен только к одному из кластеров.
- **Мягкая кластеризация**: объект может быть отнесен к нескольким кластерам сразу (с некоторыми весами).

Например, текст про применение кластеризации в области финансов в случае жесткой кластеризации по темам (финансы, анализ данных, кластеризация) будет отнесен к теме «кластеризация», а в случае мягкой кластеризации — ко всем трем темам с разными весами: к теме «кластеризация» с весом 0.5, к теме «анализ данных» с весом 0.3 и к теме «финансы» с весом 0.2.

1.3. Знакомство с методами кластеризации

Поскольку кластеры могут иметь различный вид в зависимости от специфики задачи, универсального алгоритма кластеризации не существует. При этом использование различных алгоритмов кластеризации в одной и той же задаче может давать совершенно разный ответ.

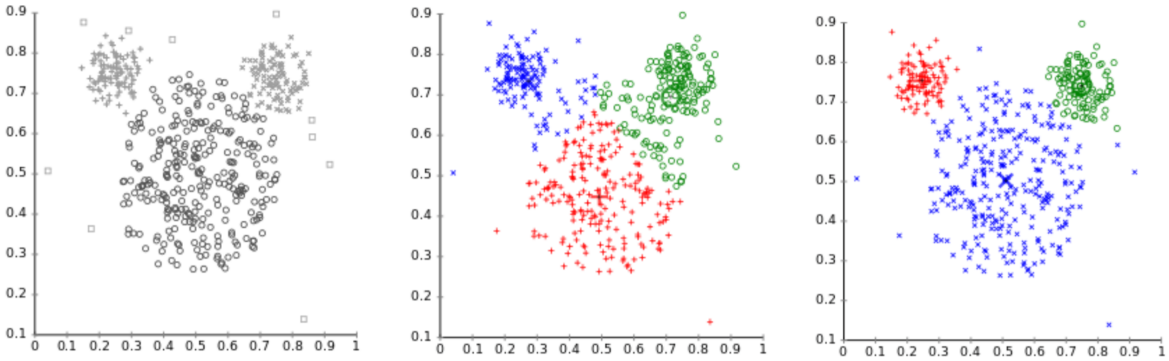


Рис. 1.6: Результаты работы метода k–средних и EM–алгоритма на модельной выборке «mouse dataset»

Поэтому необходимо рассмотреть несколько основных алгоритмов кластеризации.

1.3.1. Метод k–средних

Простейшим методом кластеризации является метод k–средних, где k — число кластеров, которые требуется выделить. Предполагается, что число k известно.

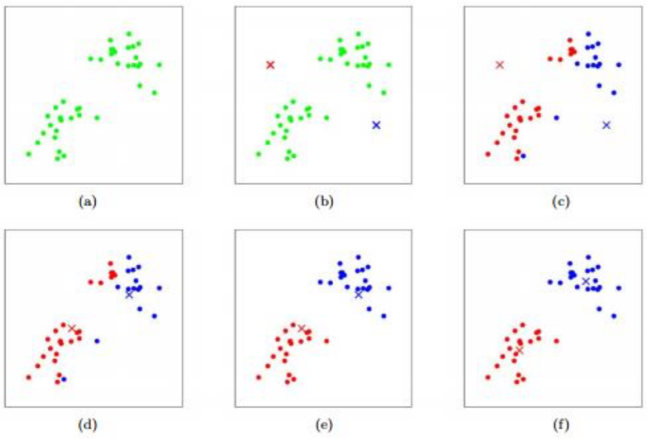


Рис. 1.7: Работа алгоритма k–средних

В начале работы алгоритма выбираются k случайных точек. Это точки играют роль центров кластеров: каждая точка будет отнесена к тому кластеру, расстояние до центра которого минимально. После этого выполняются итерации: на каждом шаге в качестве нового центра кластера выбирается среднее арифметическое всех точек, попавших в этот кластер, и обновляются метки кластеров ля точек в зависимости от близости к новым центрам кластеров. Итерации выполняются, пока не будет получен удовлетворительный результат.

1.3.2. Метод Bisect Means

Если необходимо подобрать число кластеров, можно воспользоваться методом Bisect Means, который является модификацией метода k–средних.

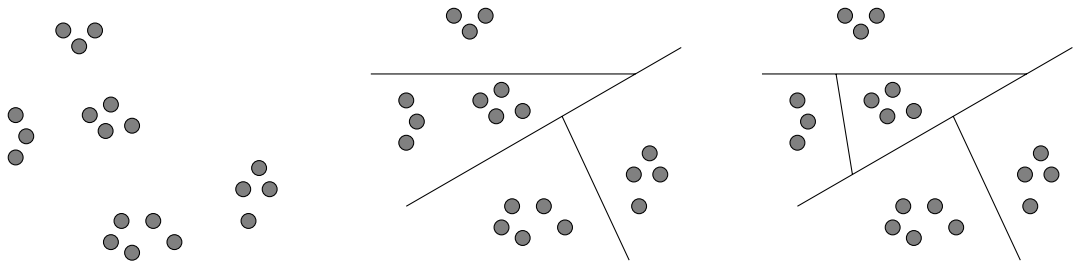


Рис. 1.8: Работа алгоритма Bisect Means

Метод заключается в следующем. На первом шаге применяется метод 2-means, то есть метод k–средних в случае двух кластеров. После этого для точек из каждого из получившихся кластеров, если это необходимо, также запускается 2-means и кластер разбивается на два. Процедура продолжается до тех пор, пока есть такой кластер, который нужно дробить.

1.3.3. EM–алгоритм

EM–алгоритм, другой метод кластеризации, заключается в максимизации правдоподобия. Он основан на том, что плотность вероятности распределения точек $p(x)$ выборки представляет собой взвешенную сумму плотностей вероятности $p_j(x)$ в каждом кластере:

$$p(x) = \sum_{j=1}^{K} w_j p_j(x),$$

где веса $w_j \geq 0$, а также $\sum_{j=1}^{K} w_j = 1$.

При этом все $p_j(x)$ выбираются из некоторого семейства распределений $\varphi(\theta; x)$ с параметром θ:

$$p_j(x) = \varphi(\theta_j; x).$$

В качестве $\varphi(\theta; x)$ часто выбирают семейство нормальных распределений.

Алгоритм заключается в последовательном выполнении двух шагов:

- **E-шаг**: Вычисляются вспомогательные переменные:

$$g_{ji} = p(\theta_j|x_i) = \frac{w_j p_j(x_i)}{p(x_i)}.$$

  Эти параметры фиксируются (так как в этом случае упрощается решение задачи максимизации).

- **M-шаг**: При зафиксированных g_{ji} решение задачи максимизации правдоподобия может быть найдено согласно:

$$w_j = \frac{1}{N}\sum_{i=1}^{N} g_{ji}, \qquad \theta_j = \text{argmax}_\theta \sum_{i=1}^{N} g_{ji} \ln \varphi(\theta; x).$$

Объект относится к кластеру j, для которого максимально значение $p(\theta_j|x_i)$.

EM–алгоритм (замечание)

Действительно, данная задача представляет собой задачу разделения смеси распределений:

$$p(x) = \sum_{j=1}^{K} w_j p_j(x), \qquad p_j(x) = \varphi(\theta_j; x),$$

Рис. 1.9: Задача разделения смеси трех распределений

в которой нужно оценить веса w_j и параметры θ_j отдельных компонентов. Эту задачу можно было бы попытаться решить напрямую:

$$w, \theta = \text{argmax}_{\theta, w} \ln p(x_i).$$

Но найти решение такой задачи крайне сложно. Поэтому в EM–алгоритме дополнительно фиксируются g_{ji}.

EM–алгоритм: смесь гауссовских распределений (пример)

Например, если выборка сгенерирована из распределения:

$$p(x) = \frac{1}{2}\mathcal{N}\left(\begin{pmatrix}4\\0\end{pmatrix}, 1\right) + \frac{1}{2}\mathcal{N}\left(\begin{pmatrix}8\\0\end{pmatrix}, 1\right),$$

где $\mathcal{N}$ — нормальное распределение:

$$\mathcal{N}(\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right).$$

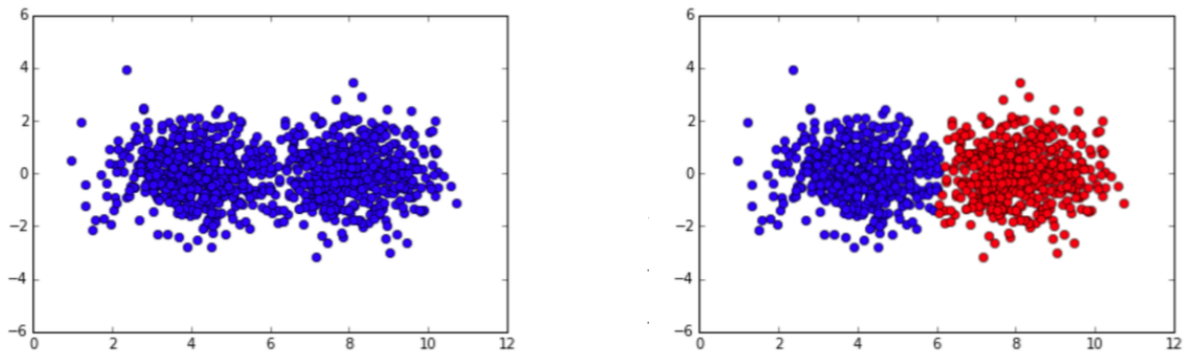


Рис. 1.10: Два получившихся сгустка можно интерпретировать как кластеры

Теперь применим к этой выборке EM-алгоритм. В случае смеси гауссовских распределений его можно выписать более точно:

- **E-шаг**:

$$g_{ji} = p(j|x_i) = \frac{w_j p_j(x_i)}{p(x_i)}$$

- **M-шаг**:

$$w_j = \frac{1}{N}\sum_{i=1}^{N} g_{ji}, \qquad \mu_j = \frac{1}{Nw_j}\sum_{i=1}^{N} g_{ji}x_i, \qquad \Sigma_j = \frac{1}{Nw_j - 1}\sum_{i=1}^{N} g_{ji}(x_i - \mu_j)(x_i - \mu_j)^T$$

Более подробно EM-алгоритм будет рассматриваться позднее.

1.3.4. Методы основанные на плотности точек

Существуют так называемые методы, основанные на плотности точек (density-based), которые используют следующую идею. Для каждой точки выборки рассматривается её окрестность, причем:

- Точка называется основной, если в ее окрестности много других точек (больше, чем некоторое число).
- Точка называется пограничной, если в ее окрестности мало других точек, но среди них есть основная.
- В ином случае точка называется шумовой.

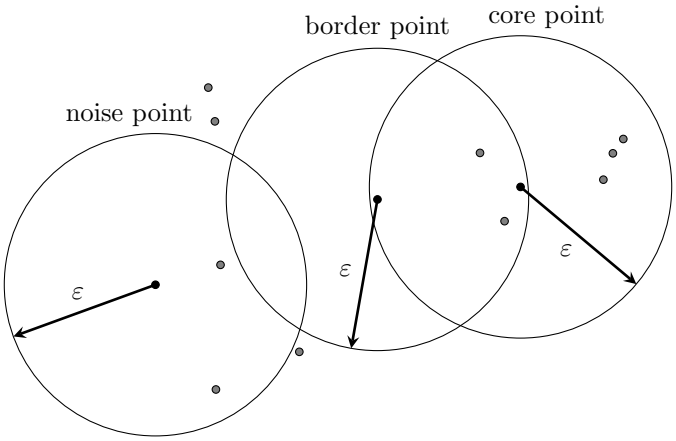


Рис. 1.11: Принцип работы алгоритма DBSCAN: изображены основная, пограничная и шумовая точки

DBSCAN — это один из **density-based** методов, который состоит из следующих шагов:

1. Разделить точки на основные, пограничные и шумовые.
2. Отбросить шумовые точки.
3. Соединить основные точки, которые находятся на расстоянии ε друг от друга.
4. Каждую группу соединенных основных точек объединить в свой кластер.
5. Отнести пограничные точки к соответствующим им кластерам.

1.3.5. Иерархическая кластеризация

Другой подход к кластеризации — иерархическая кластеризация. Он заключается в введении понятия расстояния между кластерами и состоит в следующем.

В начале каждая точка относится к своему собственному кластеру. На каждом шаге алгоритма те кластеры, расстояние между которыми минимально, объединяются в один. Постепенно все кластеры объединяются в один, а процесс их объединения представляет собой дерево. Его можно изобразить с помощью так называемых дендрограмм. По горизонтальной оси дендрограммы отложено расстояние между кластерами в момент слияния, а по вертикальной — точки выборки.

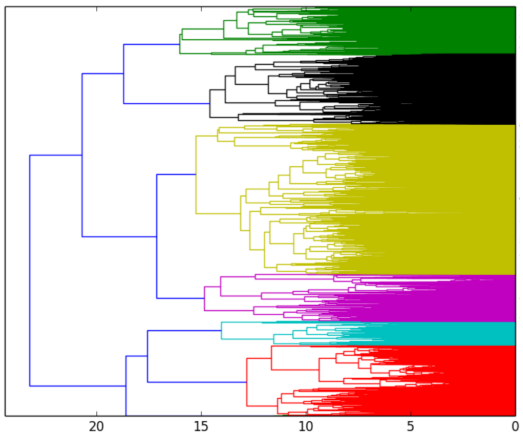


Рис. 1.12: Визуализация работы метода иерархической кластеризации с помощью дендрограмм.

В зависимости от требуемого числа кластеров, это дерево необходимо обрезать на определенной глубине. Важно, что изменение числа кластеров не требует перезапуска алгоритма, как это потребовалось бы в методе k-means.

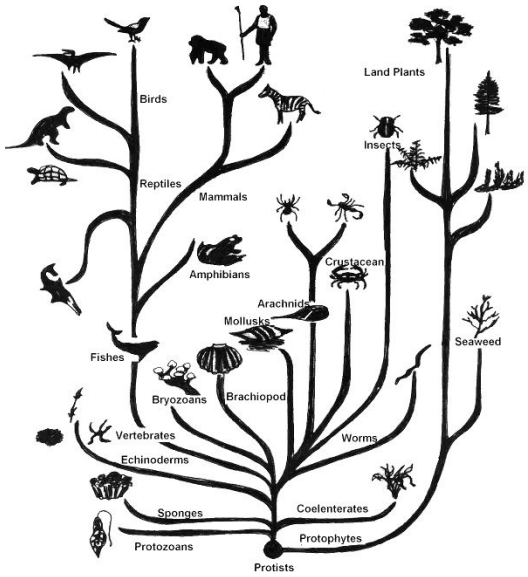


Рис. 1.13: В биологии систематизация видов позволяет глубже понять их происхождение, представляет собой деление видов на определенные группы и этим сильно напоминает рассмотренную выше иерархическую кластеризацию.

Расстояние между кластерами можно вводить несколькими разными способами:

- как среднее расстояние между объектами кластеров (Average linkage).
- как максимальное расстояние между объектами кластеров (Complete linkage).
- как минимальное расстояние между объектами кластеров (single linkage).

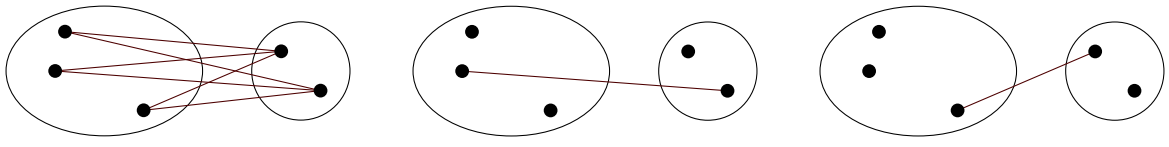


Рис. 1.14: Различные способы ввести расстояние между кластерами

1.3.6. Обзор методов кластеризации

Теперь можно систематизировать основные методы кластеризации. По получающейся структуре кластеров методы делятся на:

- Иерархические, то есть характеризующиеся сложную структуру кластеров. Среди них выделяют:
  - Агломеративные, которые характеризуются последовательным объединением кластеров.
  - Дивизионные, которые заключаются в последовательном делении одного кластера, состоящего из всех объектов, на меньшие.
- Плоские (не иерархические).

Также некоторые методы кластеризации лучше работают на кластерах определенной формы. Читателю предлагается подумать, какие методы лучше использовать в случае кластеров выпуклой формы, кластеров в форме ленты и так далее.

Также некоторые методы кластеризации позволяют сделать и жесткую, и мягкую кластеризацию, а некоторые — только жесткую.

1.4. Пример: Кластеризация текстов

1.4.1. Выборка и признаки

В этом разделе будет показано, как применять методы кластеризации на практике, на примере выборки `20newsgroups` из пакета `sklearn`. Эта выборка включает в себя множество писем на различные темы и доступна с помощью следующего кода:

```
from sklearn.datasets import fetch_20newsgroups

train_all = fetch_20newsgroups(subset='train')
print train_all.target_names
```

В результате его исполнения также был выведен список доступных тем:

```
['alt.atheism', 'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware',
↪  'comp.sys.mac.hardware', 'comp.windows.x', 'misc.forsale', 'rec.autos', 'rec.motorcycles',
↪  'rec.sport.baseball', 'rec.sport.hockey', 'sci.crypt', 'sci.electronics', 'sci.med',
↪  'sci.space', 'soc.religion.christian', 'talk.politics.guns', 'talk.politics.mideast',
↪  'talk.politics.misc', 'talk.religion.misc']
```

Для простоты можно ограничиться несколькими сильно отличающимися темами:

```
simple_dataset = fetch_20newsgroups(subset='train',
  categories=['comp.sys.mac.hardware', 'soc.religion.christian', 'rec.sport.hockey'])
```

Просмотреть письма, содержащиеся в обучающей выборке, можно следующим образом:

```
print simple_dataset.data[0]  # Вывести первое письмо в выборке
print simple_dataset.data[-1] # Вывести последнее письмо
print simple_dataset.data[-2] # Вывести предпоследнее письмо
```

Темы писем содержатся в массиве `target`:

```
simple_dataset.target   # OUTPUT: array([0, 0, 1, ..., 0, 1, 2])
```

Полное количество объектов в выборке:

```
print len(simple_dataset.data)    # OUTPUT: 1777
```

В качестве признаков можно использовать частоты слов (`CountVectorizer`) или взвешенные частоты слов (`TfidfVectorizer`). Следующий код подключает необходимые функции:

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
```

В обоих случаях, чтобы обеспечить небольшое количество признаков, можно установить максимальный и минимальный пороги для документной частоты слов:

```
vectorizer = CountVectorizer(max_df=500, min_df=10)
```

Поскольку слова, которые встречаются почти во всех письмах или, наоборот, очень редко, вряд ли будут полезны, но, отбросив их, можно существенно упростить задачу с вычислительной точки зрения.

После этого можно создать матрицу «объект–признак»:

```
matrix = vectorizer.fit_transform(simple_dataset.data)
```

Размер этой матрицы при данном способе построения признакового описания:

```
print matrix.shape    # OUTPUT: (1777, 3767)
```

То есть в данном случае используются 3767 признаков. Получить список созданных признаков можно следующей командой:

```
vectorizer.get_feature_names()
```

1.4.2. Агломеративная кластеризация (neighbour joining)

В качестве метода кластеризации можно использовать агломеративную кластеризацию, поскольку в нем можно задать желаемую функцию близости. Поскольку решается задача кластеризации текстов, имеет смысл использовать косинусную меру:

```
from sklearn.cluster.hierarchical import AgglomerativeClustering

model = AgglomerativeClustering(n_clusters=3, affinity='cosine', linkage='complete')
preds = model.fit_predict(matrix.toarray())
```

Особое внимание следует обратить на то, что перед выполнением `fit_predict` матрица преобразуется из разреженного к плотному (в котором явно хранятся нулевые элементы) формату, поскольку данная реализация алгоритма не поддерживает работу с разреженными матрицами.

Теперь можно вывести результат работы алгоритма:

```
print list(preds)
```

Читатель, запустив этот алгоритм у себя на компьютере, легко убедится, что практически все письма были отнесены к одному кластеру:

```
  [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, 2, 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, 2, 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, 1, 0, 0,
↪  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2,..., 0, 2, 1]
```

Результаты не впечатляют, даже если использовать взвешенные частоты слов или другие значения порогов отсечения. В данном случае могла бы оказаться полезной фильтрация признаков, но об этом речь пойдет в курсе позднее.

1.4.3. Метод K-средних

Другой метод кластеризации — метод K-средних:

```
from sklearn.cluster import KMeans

model = KMeans(n_clusters=3, random_state=1)
preds = model.fit_predict(matrix.toarray())
```

Здесь `random_state` полностью определяет случайные значения, которые используются в методе K-средних. Это необходимо для обеспечения воспроизводимости результатов.

Результат выполнения алгоритма и истинные значения ответов:

```
print preds                 # Результат кластеризации
print simple_dataset.target # Истинные ответы
```

```
[0 0 2 ..., 0 2 1] # Результат кластеризации
[0 0 1 ..., 0 1 2] # Истинные ответы
```

Прогнозы сделаны правильно, если изменить нумерацию кластеров:

```
mapping = {2 : 1, 1: 2, 0: 0}
mapped_preds = [mapping[pred] for pred in preds]
```

Теперь можно подсчитать долю случаев, когда тема была определена неверно:

```
print float(sum(mapped_preds != simple_dataset.target)) / len(simple_dataset.target)
```

```
0.0483961733258
```

То есть точность составляет порядка 95%.

Для сравнения можно воспользоваться классификатором на тех же данных:

```
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import cross_val_score
clf = LogisticRegression()
print cross_val_score(clf, matrix, simple_dataset.target).mean()
```

```
0.985360318588
```

Качество его работы составляет порядка 98%. То есть на данной выборке метод K–средних работает не сильно хуже классификатора. Ситуация может измениться, если взять выборку посложнее.

Пусть выбраны три похожие темы (все про компьютеры):

```
dataset = fetch_20newsgroups(
    subset='train',
    categories=['comp.sys.mac.hardware', 'comp.os.ms-windows.misc', 'comp.graphics'])
```

И также использован метод K-средних:

```
matrix = vectorizer.fit_transform(dataset.data)
model = KMeans(n_clusters=3, random_state=42)
preds = model.fit_predict(matrix.toarray())
print preds
print dataset.target
```

```
[0 1 2 ..., 0 2 0]
[2 1 1 ..., 2 0 2]
```

Видно, что качество работы алгоритма упало:

```
mapping = {2 : 0, 1: 1, 0: 2}
mapped_preds = [mapping[pred] for pred in preds]
print float(sum(mapped_preds != dataset.target)) / len(dataset.target)
```

```
0.260125499144
```

Для сравнения результат работы классификатора:

```
clf = LogisticRegression()
print cross_val_score(clf, matrix, dataset.target).mean()
```

```
0.917279226713
```

В данном случае классификатор дает качество 91%, а алгоритм кластеризации только 73%, что существенно хуже.

1.4.4. SVD + KMeans

Качество кластеризации можно улучшить, если воспользоваться сингулярным разложением матриц (SVD) для уменьшения числа признаков (до `n_components`):

```
from sklearn.decomposition import TruncatedSVD

model = KMeans(n_clusters=3, random_state=42)
svd = TruncatedSVD(n_components=1000, random_state=123)
features = svd.fit_transform(matrix)
preds = model.fit_predict(features)
print preds
print dataset.target
```

```
[0 2 1 ..., 0 1 0]
[2 1 1 ..., 2 0 2]
```

Тогда:

```
mapping = {0 : 2, 1: 0, 2: 1}
mapped_preds = [mapping[pred] for pred in preds]
print float(sum(mapped_preds != dataset.target)) / len(dataset.target)
```

```
0.206503137479
```

В результате получается всего 20% ошибок. Теперь можно еще сильнее уменьшить число признаков:

```
model = KMeans(n_clusters=3, random_state=42)
svd = TruncatedSVD(n_components=200, random_state=123)
features = svd.fit_transform(matrix)
preds = model.fit_predict(features)
print preds
print dataset.target
```

```
[2 0 1 ..., 2 1 2]
[2 1 1 ..., 2 0 2]
```

Соответствие между метками тем и номерами кластеров могло измениться, поэтому имеет смысл перебрать все возможные соответствия (то есть все возможные перестановки чисел 1,2 и 3):

```
import itertools
def validate_with_mappings(preds, target, dataset):
    permutations = itertools.permutations([0, 1, 2])
    for a, b, c in permutations:
        mapping = {2 : a, 1: b, 0: c}
        mapped_preds = [mapping[pred] for pred in preds]
        print float(sum(mapped_preds != target)) / len(target)

validate_with_mappings(preds, dataset.target, dataset)
```

```
0.900741585853
0.674272675414
0.705647461495
0.893896177981
0.205362236167
0.620079863092
```

В случае лучшей перестановки все равно 20% ошибок.

Может показаться подозрительным, что 200 признаков вместо 1000 дают такой же результат. И действительно: это просто удачное совпадение, поскольку изменив значение `random_state`, качество получается уже другим:

```
model = KMeans(n_clusters=3, random_state=42)
svd = TruncatedSVD(n_components=200, random_state=321)
features = svd.fit_transform(matrix)
preds = model.fit_predict(features)
print preds
print dataset.target
validate_with_mappings(preds, dataset.target, dataset)
```

```
[2 1 0 ..., 2 0 2]
[2 1 1 ..., 2 0 2]
0.713063320023
0.845407872219
0.889332572732
0.70051340559
0.586423274387
0.265259555048
```

При использовании другого значения `random_state` ошибок уже 26%.

1.5. Выбор метода кластеризации

1.5.1. Алгоритмы кластеризации в пакете scikit-learn

В этом разделе урока будет посвящено особое внимание проблеме выбора метода кластеризации в зависимости от специфики конкретной задачи.

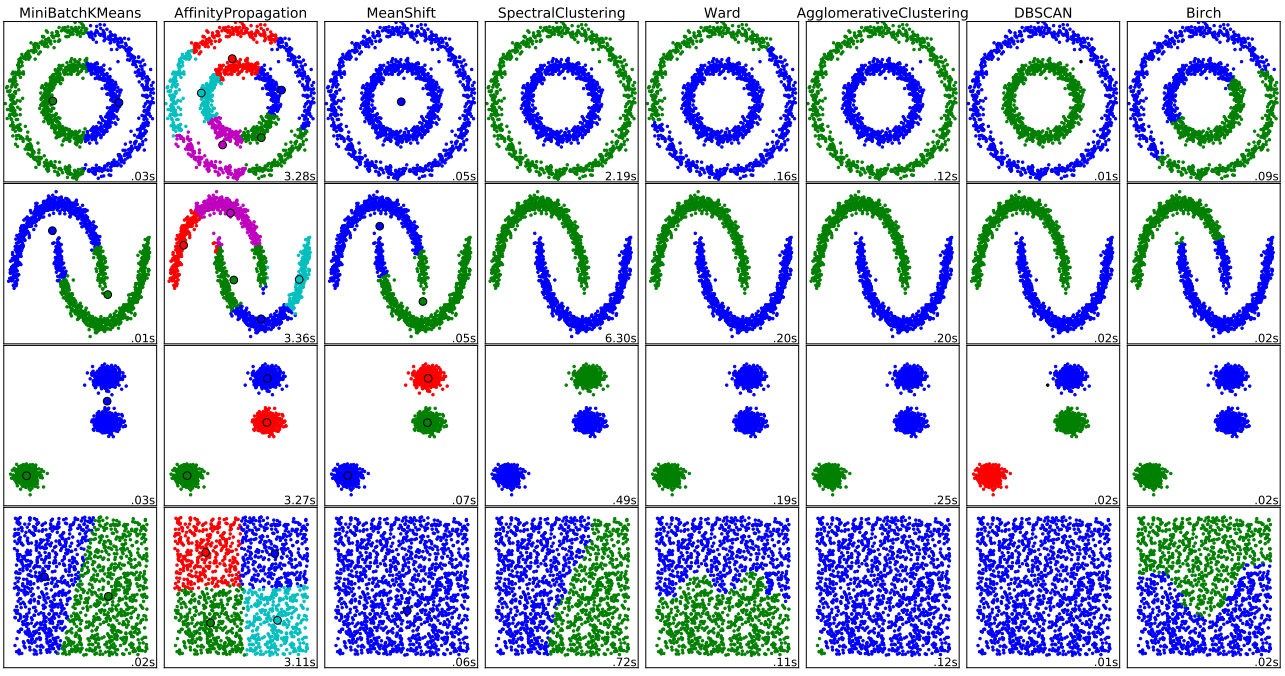
В пакете scikit-learn доступны следующие из рассмотренных алгоритмы:

- **Kmeans** — метод K-средних.
- **MiniBatchKMeans** —- разновидность метода K-средних для случая большой выборки. В этом случае алгоритм работает на случайных подмножествах обучающей выборки.
- **GaussianMixture** — EM-алгоритм
- **AgglomerativeClustering** — агломеративная иерархическая кластеризация
- **Ward** — вид аггломеративной кластеризации, предназначенный для использования с евклидовым расстоянием.
- **DBSCAN** — алгоритм кластеризации, основанный на плотности.

Также в scikit-learn доступны следующие методы, которые расмотрены не были: **MeanShift**, **AffinityPropagation**, **SpectraClustering**, **Birch**.

1.5.2. Демонстрация алгоритмов из пакета scikit-learn

Следующий пример взят из документации пакета scikit-learn.



Он демонстрирует работу различных алгоритмов на нескольких модельных выборках:

- Два концентрических кольца (пример невыпуклых кластеров)
- Кластеры-ленты
- Несколько «сгустков точек», которые легко разделяются
- Равномерное заполнение точками пространства признаков (адекватный алгоритм отнесет все точки к одному кластеру)

Следует отметить, что некоторые алгоритмы имеют в качестве своего параметра количество кластеров, в частности KMeans и агломеративная кластеризация. Этот параметр был установлен равным двум.

Из представленных алгоритмов наиболее адекватные результаты дают агломеративная иерархическая кластеризация и DBSCAN. Они будут подробнее рассмотрены далее. Также будет детально рассмотрен метод K-средних из-за простоты и возможности работы с большими выборками.

1.5.3. Особенности основных алгоритмов кластеризации

Метод K-средних

- **Параметры:** число кластеров.
- **Масштабируемость:** MiniBatch (реализация KMeans) может работать на очень большой выборке.
- **Сценарий использования:** выпуклые кластеры примерно одинакового размера.
- **Метрика:** Евклидова. Строгого запрета на использование другой метрики нет, но это не совсем корректно (будет рассмотрено позже).

EM-алгоритм с нормальным распределением в каждом кластере

- **Параметры:** число компонент.
- **Масштабируемость:** Чтобы восстановить значения параметров и исключить переобучение, нужна большая выборка. Метод практически не масштабируем.
- **Сценарий использования:** задача восстановления плотности. Кластеры предполагаются выпуклыми.
- **Метрика:** обобщение евклидовой метрики (об этом будет рассказано позднее).

Агломеративная кластеризация

- **Параметры:** число кластеров, linkage (метод вычисления расстояния между кластерами) и метрика.
- **Масштабируемость:** Масштабируется на случай большого числа объектов и большого числа кластеров.
- **Сценарий использования:** Случай большого числа кластеров, так как у метода существует тенденция к образованию одного большого кластера.
- **Метрика:** Любая метрика или функция близости.

DBSCAN

- **Параметры:** радиус окрестности и количество соседей, необходимое, чтобы считать точку основной.
- **Масштабируемость:** Алгоритм может работать в случае большого количества объектов и умеренного количества кластеров.
- **Сценарий использования:** неравные невыпуклые кластеры, при необходимости отсеивать выбросы.
- **Метрика:** Евклидова (в реализации sklearn)