

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, \dots, x_ℓ и ответов на них y_1, \dots, y_ℓ , требовалось спрогнозировать ответы для объектов x_{l+1}, \dots, 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, \dots, x_l состоит только из объектов, но не содержит ответы на них, а также одновременно является и тестовой выборкой. Требуется расставить метки y_1, \dots, y_l таким образом, чтобы похожие друг на друга объекты имели одинаковую метку, то есть разбить все объекты на некоторое количество групп.

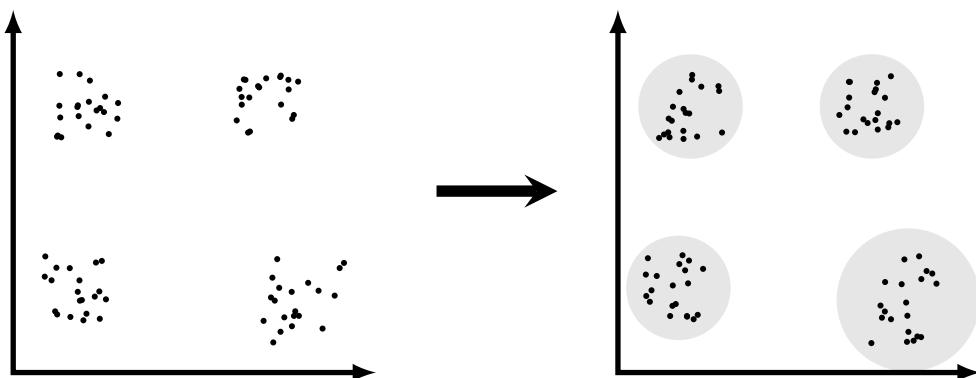


Рис. 1.1: Пример задачи кластеризации

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):

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

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:

$$\frac{F_0}{F_1} \rightarrow \min.$$

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

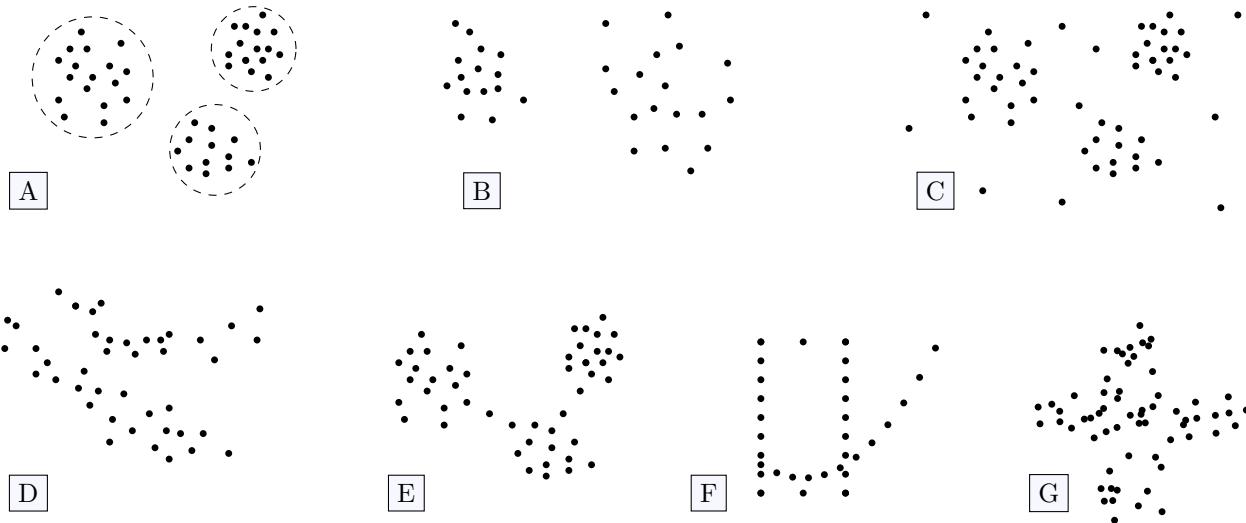


Рис. 1.2: Примеры различных форм кластеров

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».

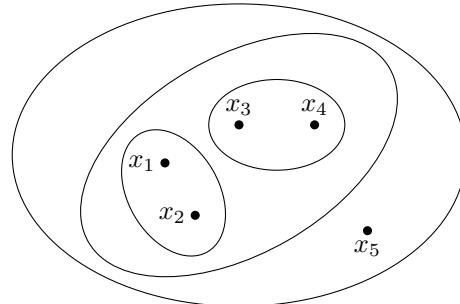


Рис. 1.3: Пример возможной иерархии кластеров.

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.

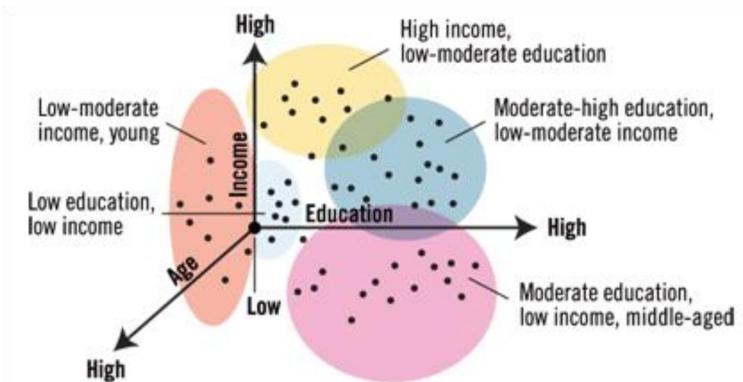


Рис. 1.4: Задача сегментации целевой аудитории

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 аналитики, 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:

- **Жёсткая 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.

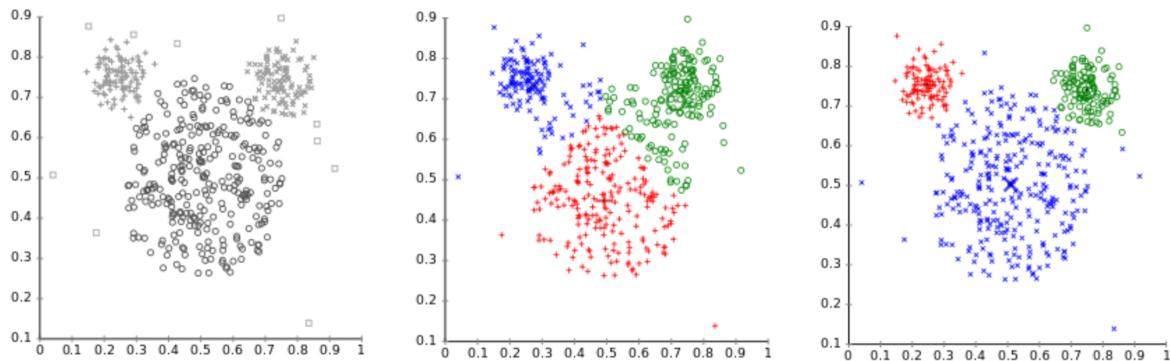


Рис. 1.6: Результаты работы метода k -средних и ЕМ-алгоритма на модельной выборке «mouse dataset»

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.

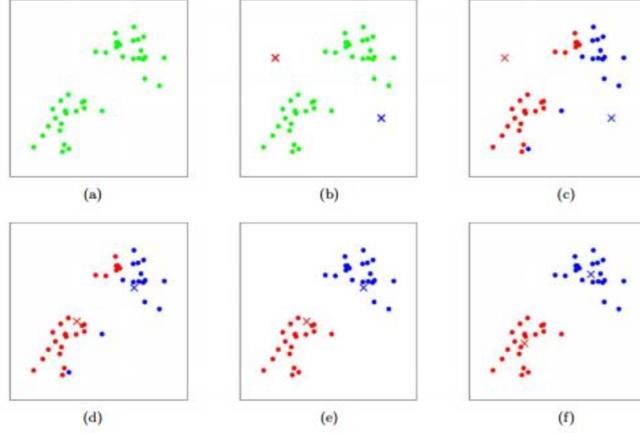


Рис. 1.7: Работа алгоритма k -средних

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 для 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.

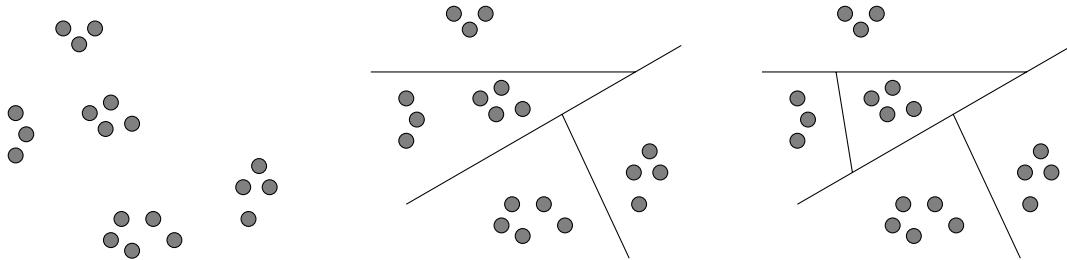


Рис. 1.8: Работа алгоритма Bisect Means

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:

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

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:

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

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:

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

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:

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

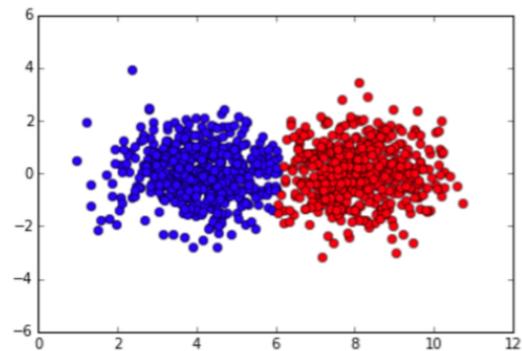
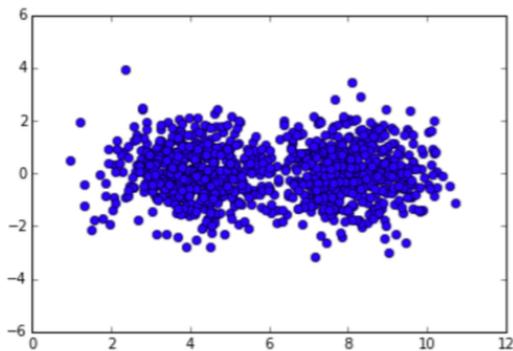


Рис. 1.10: Два получившихся сгустка можно интерпретировать как кластеры

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:**

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

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.

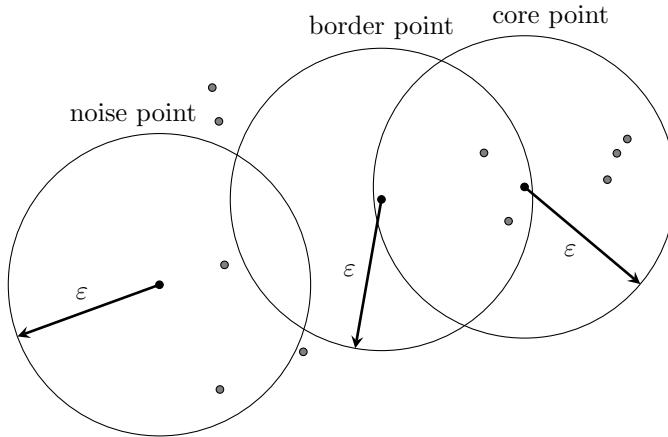


Рис. 1.11: Принцип работы алгоритма DBSCAN: изображены основная, пограничная и шумовая точки

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.

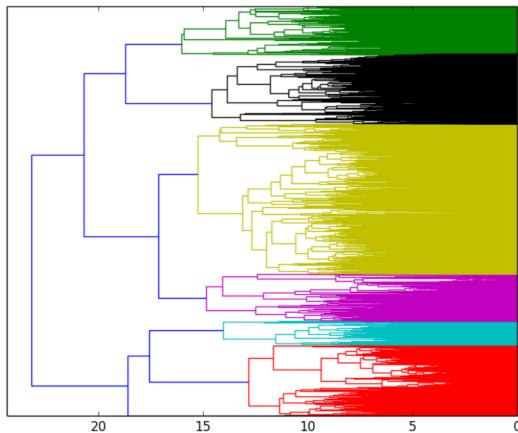


Рис. 1.12: Визуализация работы метода иерархической кластеризации с помощью дендрограмм.

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.

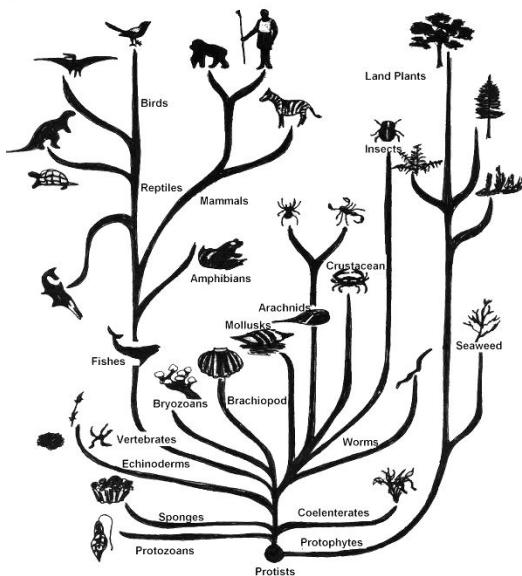


Рис. 1.13: В биологии систематизация видов позволяет глубже понять их происхождение, представляет собой деление видов на определенные группы и этим сильно напоминает рассмотренную выше иерархическую кластеризацию.

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).

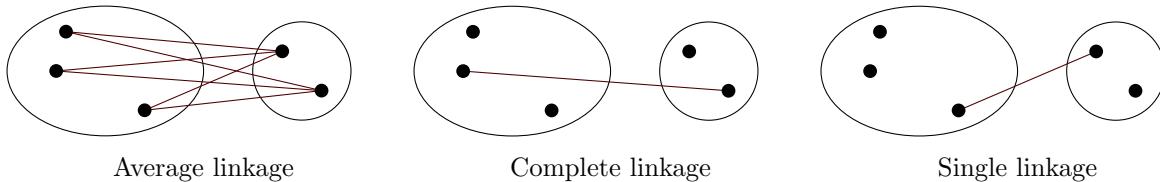


Рис. 1.14: Различные способы ввести расстояние между кластерами

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:

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 К-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 классификации  
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.

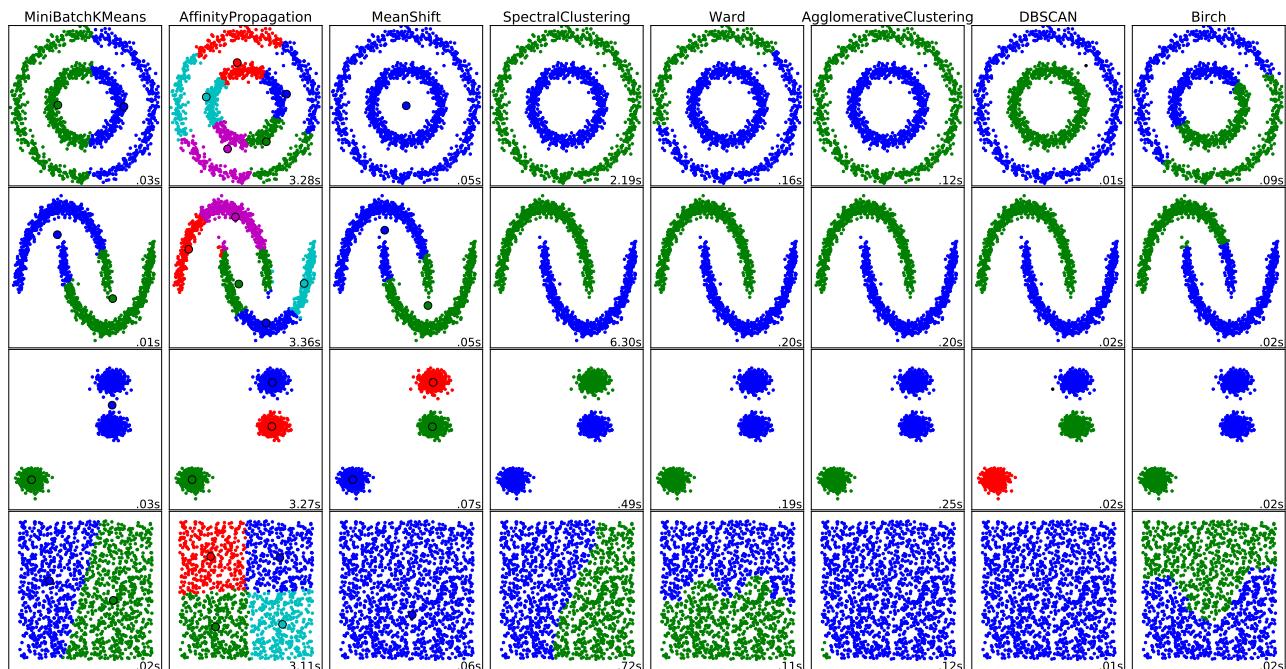
В пакете scikit-learn доступны следующие из рассмотренных алгоритмы:

- **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 агломеративной кластеризации, предназначенный для использования с евклидовым расстоянием.
- **DBSCAN** — алгоритм кластеризации, основанный на плотности.

Также в scikit-learn доступны следующие методы, которые рассмотрены не были: **MeanShift**, **AffinityPropagation**, **SpectralClustering**, **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)

Урок 2

Подробнее о методах кластеризации

2.1. Метод К средних (K-Means)

2.1.1. Как работает K-Means

Алгоритм K-means представляет собой итеративный процесс. Пусть в начале произвольным образом выбраны центры классов. Объект относится к тому кластеру, расстояние до центра которого меньше. На каждой итерации сначала пересчитываются положения центра каждого класса как среднее арифметическое отнесенных к нему точек, а после этого объекты перераспределяются на основании новых положений центров.

2.1.2. Варианты метода K-Means

Описанный выше вариант метода K-means называется K-Means'ом Болла Холла. Существует также версия Мак Кина, в которой при каждом переходе объекта из кластера в кластер центры кластеров пересчитываются.

В ситуации, когда количество объектов очень велико, чтобы не вычислять среднее арифметическое по всем объектам, можно на каждой итерации считать положения центров классов по некоторой случайной выборке объектов. Такой вариант метода называется Mini Batch K-Means и позволяет использовать метод K-Means на огромных выборках.

Если размерность пространства признаков d велика, то операция вычисление расстояния между объектами будет вычислительно сложной (требовать порядка d элементарных операций). Решить проблему можно уменьшением числа признаков: использовать отбор признаков, метод главных компонент (PCA) или сингулярное разложение (SVD).

Сходимость метода K-means сильно зависит от выбора начальных значений центров. В случае двух кластеров можно в качестве центров выбрать две самые удаленные друг от друга точки. Для произвольного числа кластеров существует алгоритм k-means++. Это вариация метода K-means, в которой начальные приближения выбираются не случайно, а некоторым более оптимальным образом. Положение первого центра класса выбирается случайно из равномерного распределения на выборке. На каждом следующем шаге сначала вводится такое вероятностное распределение на выборке, что вероятность выбора точки пропорциональна квадрату расстояния до ближайшего до нее центра, а положение нового центра выбирается из этого распределения.

2.1.3. Пример: квантизация изображений

Допустим, необходимо понизить количество цветов в изображении. Изначально изображение содержит около 96 тысяч цветов из пространства цветов RGB.

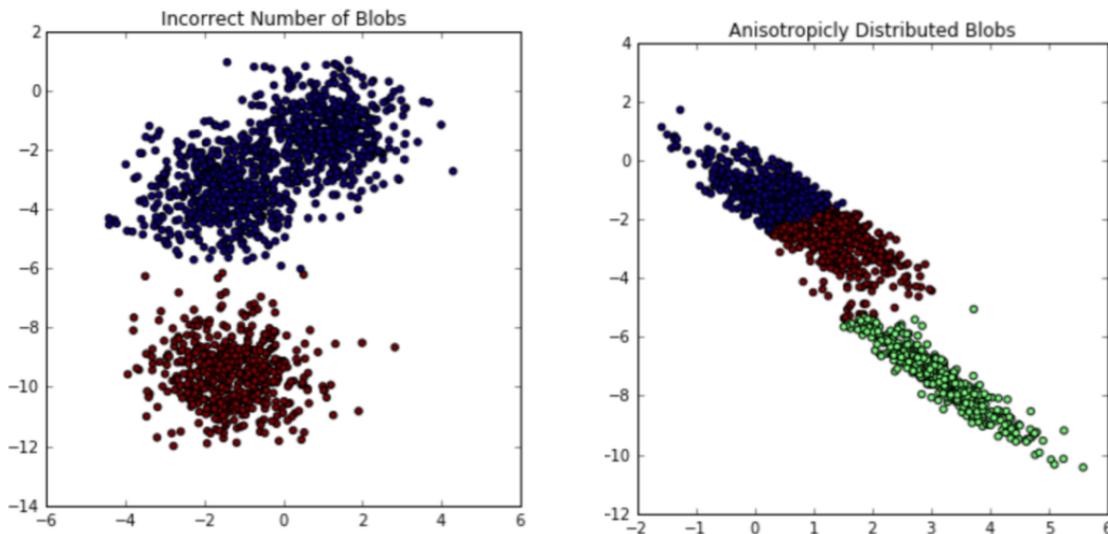
Можно уменьшить количество цветов, выбрав 64 случайных цвета, а остальные цвета на изображении заменить на ближайший в пространстве RGB цвет из них. С другой стороны, точки изображения можно рассматривать как объекты в пространстве признаков и кластеризовать их, например, с помощью K Means. Такой способ дает очень хороший результат.



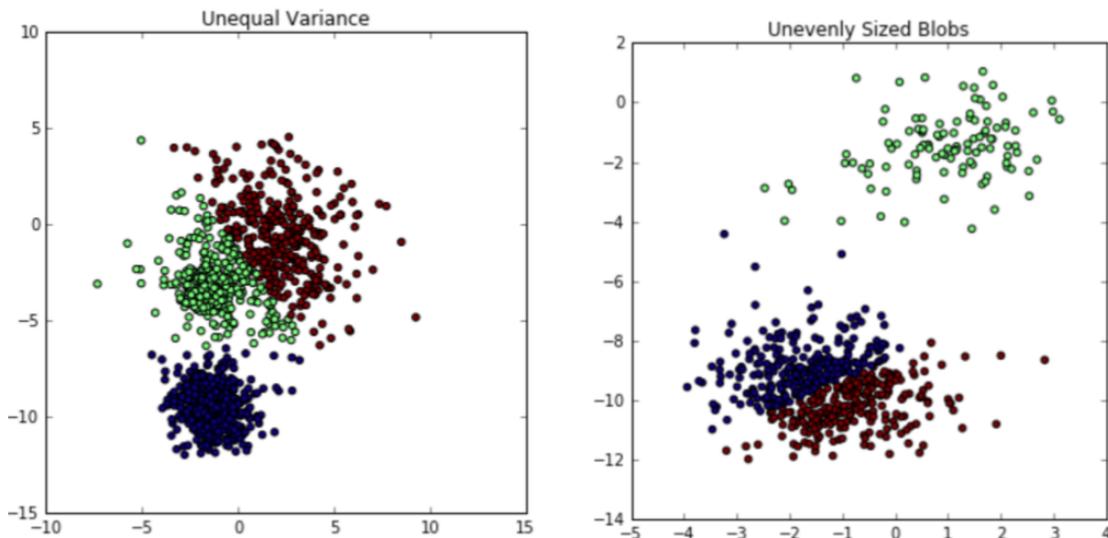
Кластеризация, таким образом, представляет собой естественный способ огрубления признаков и позволяет свести возможные варианты векторов признаков к какому то небольшому набору.

2.1.4. K-means и форма кластеров

Метод K-means может совершенно по-разному работать в зависимости от формы кластеров и от выбранного количества кластеров. Например, алгоритм может решить, что два вытянутых кластера — это одно скопление.

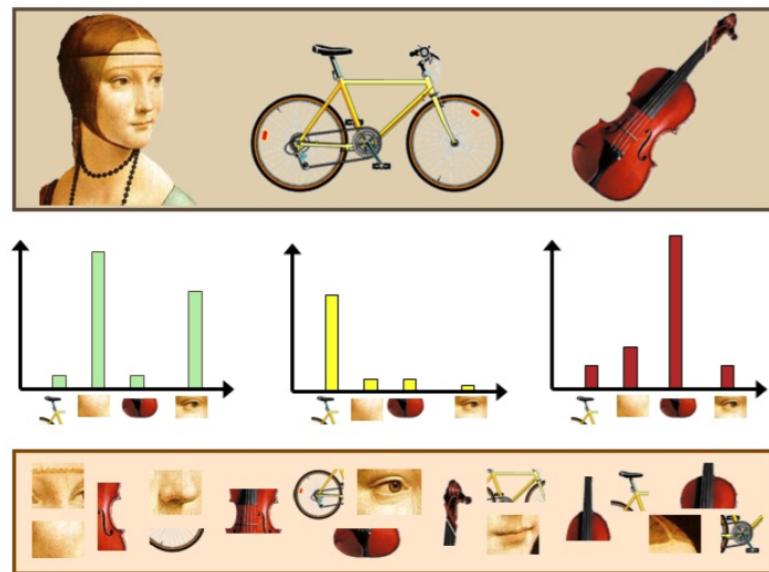


Также, в случае неодинакового разброса у кластеров, алгоритм может не очень оптимально поделить два рядом находящихся друг с другом кластера.

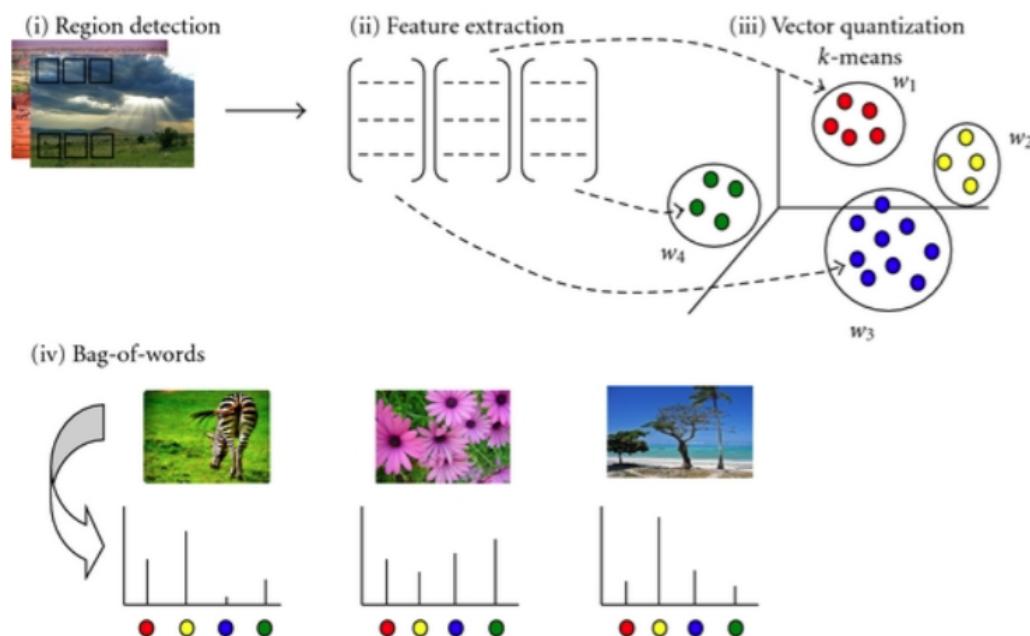


2.1.5. Пример: мешок визуальных слов

Другой пример использования K Means — это мешок визуальных слов. До того как в анализе изображений начали активно использовать свёрточные нейросети, применяли следующий подход для классификации: нарезали изображения на множество кусочков и по аналогии с тем, как классифицируются тексты по частотам разных слов в тексте, в качестве признаков изображения выбирали частоты тех или иных фрагментов изображения.



Проблема заключалась в том, что точное совпадение фрагментов двух изображений практически невозможно. Поэтому фрагменты изображений помещались в признаковое пространство и кластеризовались с помощью K-means. Каждый из получившихся кластеров интерпретировался как «визуальное слово».



2.1.6. Функционал, который минимизирует K-means

Среднее внутрекластерное расстояние имеет следующий вид:

$$F_0 = \frac{\sum_{i < j} [y_i = y_j] \rho(x_i, x_j)}{\sum_{i < j} [y_i = y_j]} \rightarrow \min.$$

Если известны положения центров кластеров, можно записать аналогичную метрику, которая представляет собой среднее расстояние до центра кластера:

$$\Phi_0 = \sum_{y \in Y} \frac{1}{|K_y|} \sum_{i: y_i = y} \rho^2(x_i, \mu_y) \rightarrow \min.$$

Оказывается, что K-Means в варианте Мак Кина находит локальный минимум функционала Φ_0 . Это интуитивно понятно из того факта, что в K-Means итеративно минимизируется средние внутрикластерные расстояния: объект присваивается к тому кластеру, центр которого ближе, а центр кластера перемещается в среднее арифметическое векторов признаков объектов. Легко убедиться, что в одномерном пространстве выбор точки μ минимизирует среднеквадратичное отклонение от других точек кластера:

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i = \operatorname{argmin}_{\mu} \frac{1}{N} \sum_{i=1}^N (\mu - x_i)^2.$$

Действительно:

$$\frac{d}{d\mu} \frac{1}{N} \sum_{i=1}^N (\mu - x_i)^2 = \frac{2}{N} \sum_{i=1}^N (\mu - x_i) = 0 \implies \mu = \frac{1}{N} \sum_{i=1}^N x_i.$$

2.2. Expectation Maximization (EM-алгоритм)

2.2.1. Постановка задачи

EM-алгоритм, как и k-means, является итеративным методом: сначала задается начальное приближение, а затем на каждом шаге форма кластеров уточняется.

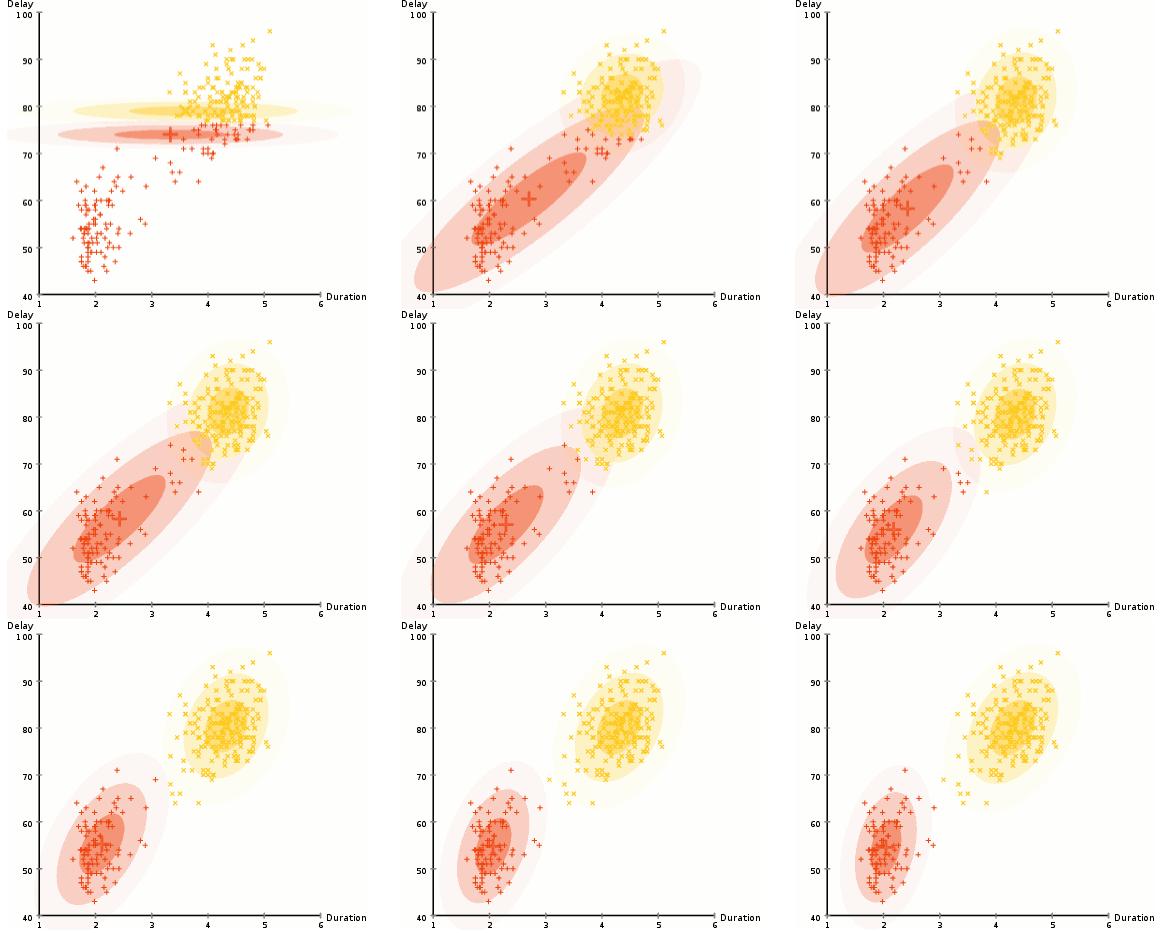


Рис. 2.1: Пример работы EM-алгоритма

2.2.2. Постановка задачи разделения смеси распределений

Пусть w_1, \dots, w_K — априорные вероятности кластеров, $p_1(x), \dots, p_K(x)$ — плотности распределения кластеров, тогда плотность распределения вектора признаков x сразу по всем кластерам равна:

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

Необходимо на основе выборки оценить параметры модели w_1, \dots, w_K , $p_1(x), \dots, p_K(x)$. Это позволит оценивать вероятность принадлежности к кластеру и, таким образом, решить задачу кластеризации. Такая задача называется задачей разделения смеси распределений:

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

где θ_j — параметры распределения $p_j(x)$.

Обычно все распределения отдельных компонент берутся из одного семейства. Например, все компоненты могут иметь нормальное распределение.

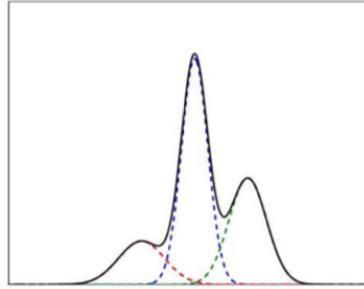


Рис. 2.2: Смесь трех нормальных распределений.

Согласно принципу максимизации правдоподобия:

$$\theta, w = \underset{\theta, w}{\operatorname{argmax}} \sum_{i=1}^K \ln p(x_i) = \underset{\theta, w}{\operatorname{argmax}} \sum_{i=1}^K \ln \left(\sum_{j=1}^K w_j p_j(x_i) \right).$$

Таким образом, имеет место задача максимизации суммы логарифмов сумм, решение которой представляет большую трудность. В таком случае полезным оказывается итеративный метод решения — EM-алгоритм.

2.2.3. EM-алгоритм

EM-алгоритм заключается в последовательном выполнении двух шагов:

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

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

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

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

Итерации происходят до сходимости.

2.2.4. Пример: 2 кластера с гауссовской плотностью

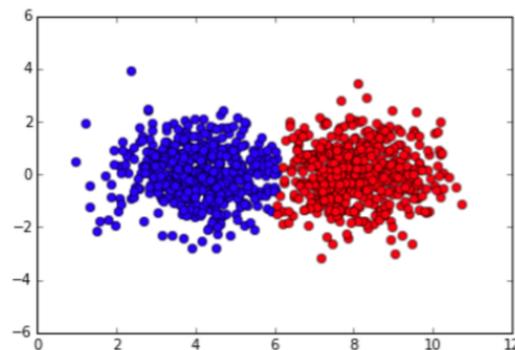


Рис. 2.3: Два получившихся сгустка можно интерпретировать как кластеры

В случае смеси гауссовых распределений:

$$p(x) = w_1 p_1(x) + w_2 p_2(x),$$

можно выписать более подробные выражения для М-шага:

- **E-шаг:**

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

- **M-шаг:**

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

Объект относится к кластеру j , для которого максимально значение $p(j|x_i) = g_{ij}$.

2.2.5. Простое и классическое объяснения ЕМ-алгоритма

Простое объяснение ЕМ-алгоритма заключается в том, что для построения итерационного процесса нужно ввести скрытые переменные таким образом, чтобы вычисления были как можно более простые. Оказывается, что в качестве таких скрытых переменных удобно рассмотреть (здесь сразу применена формула Байеса):

$$g_{ji} = P(j|x_i) = \frac{w_j p_j(x_i)}{\sum_{k=1}^K w_k p_k(x_i)}.$$

На втором шаге (М-шаге) решается задача максимизации правдоподобия с зафиксированными $P(j|x_i)$. Если приравнять производные по параметрам к нулю, получаются выражения:

$$w_J = \frac{1}{N} \sum_{i=1}^N g_{ji}, \quad \theta_j = \operatorname{argmax}_{\theta} \sum_{i=1}^N g_{ji} \ln \phi(\theta; x).$$

Классическое объяснение ЕМ-алгоритма (которое объясняет его название Expectation-Maximization) заключается в следующем. Пусть $L(\theta; X, Z) = p(X, Z|\theta)$ — функция правдоподобия. Тогда в ЕМ-алгоритме на Е-шаге происходит построение функции ожидаемого значения логарифма правдоподобия, зависящей от параметров θ :

$$Q(\theta|\theta^{(t)}) = E_{Z|X,\theta^{(t)}} \log L(\theta; X, Z),$$

а на М-шаге вычисляются значения параметров $\theta^{(t+1)}$, которые максимизируют $Q(\theta|\theta^{(t)})$:

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} Q(\theta|\theta^{(t)}).$$

2.2.6. Приложения ЕМ-алгоритма

ЕМ-алгоритм имеет следующие приложения:

- Оценка параметров в других вероятностных моделях (не только в смеси распределений).
- Восстановление плотности распределения.
- Классификация.

2.3. Агломеративная иерархическая кластеризация

2.3.1. Иерархическая кластеризация

Иерархическая кластеризация — кластеризация, в которой кластеры получаются вложенными друг в друга. Выделяют два способа это сделать:

- Агломеративный подход: каждый объект помещается в свой собственный кластер, которые постепенно объединяются.
- Дивизивный (англ. divisive) подход: сначала все объекты помещаются в один кластер, который затем разбивается на более мелкие кластеры.

Более распространен агломеративный подход, поэтому, когда говорят «иерархическая кластеризация», часто имеют в виду именно его. Что именно имеется в виду, к сожалению, приходится понимать из контекста.

2.3.2. Агломеративная кластеризация

На данном примере показан ход выполнения агломеративной иерархической кластеризации. Считается, что выбран некоторый способ вычисления расстояния между кластерами.

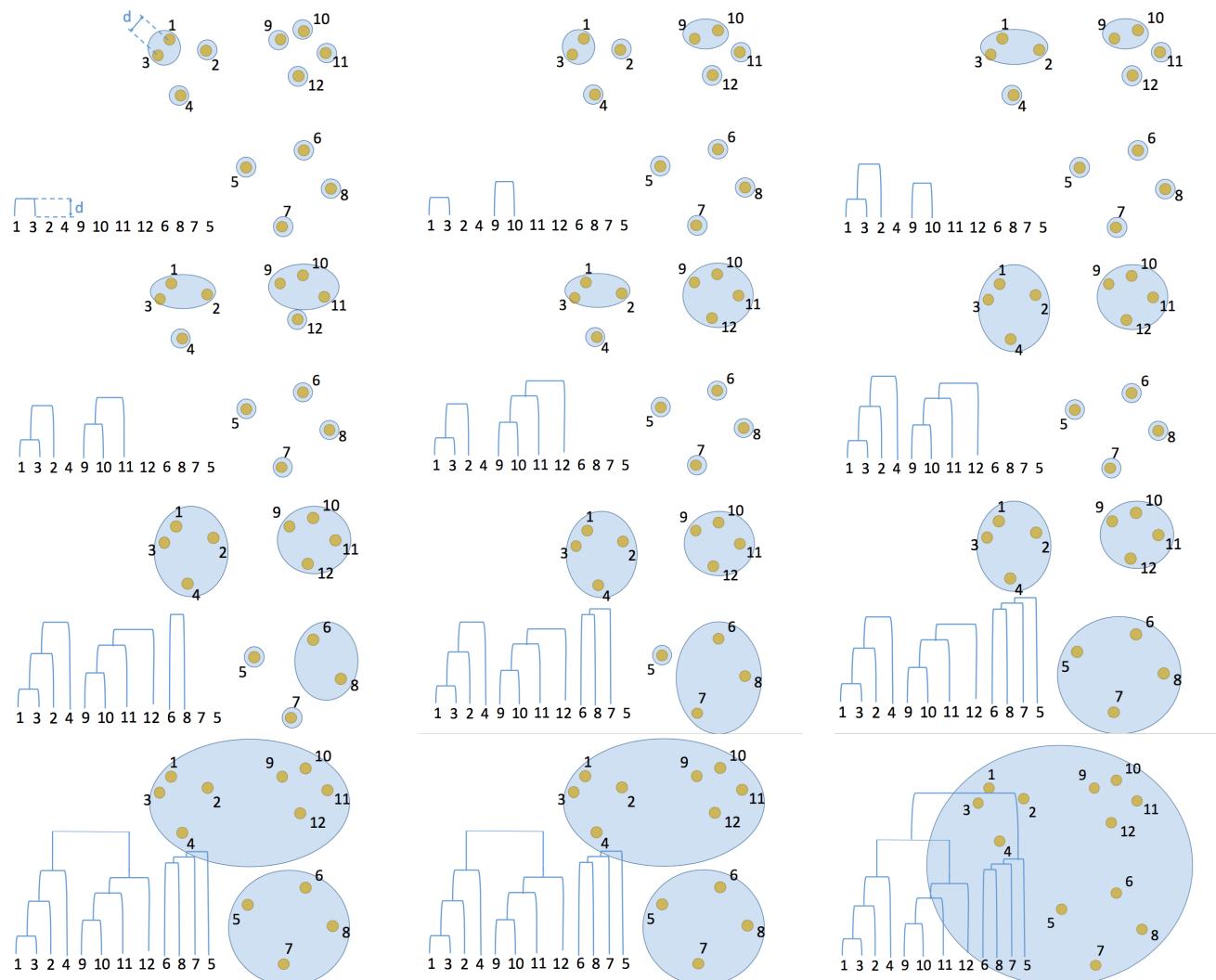


Рис. 2.4: Процесс агломеративной иерархической кластеризации (сразу представлено построение так называемой дендрограммы, о которой пойдет речь несколько позже)

Главная особенность метода агломеративной иерархической кластеризации состоит в том, что для изменения желаемого числа кластеров запускать заново алгоритм не нужно. Для этого достаточно рассмотреть дерево объединения кластеров и обрезать его на желаемом шаге.

2.3.3. Расстояние между кластерами

Расстояние между кластерами можно ввести несколькими разными способами.

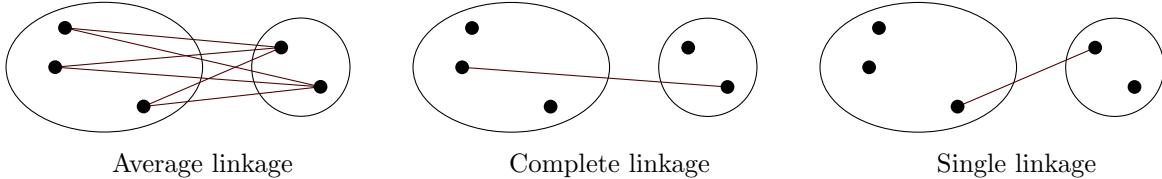


Рис. 2.5: Различные способы ввести расстояние между кластерами

С помощью формулы Ланса-Уильямса можно обобщить множество разных способов ввести расстояния между кластерами:

$$R(U \cup V, S) = \alpha_U R(U, S) + \alpha_V R(V, S) + \beta R(U, V) + \gamma |R(U, S) - R(V, S)|.$$

Эта формула выражает расстояние между кластером, которым получается в результате слияния двух кластеров U и V , и каким-то третьим кластером S . Формула позволяет рекурсивно получить расстояние между двумя сложными кластерами, если известно расстояние между простыми. Разные коэффициенты в этой формуле приводят к разным способам вычислять расстояние между кластерами, в том числе:

- Расстояние ближайшего соседа (при $\alpha_U = \alpha_V = \frac{1}{2}$, $\beta = 0$, $\gamma = -\frac{1}{2}$):

$$R^B(W, S) = \min_{w \in W, s \in S} \rho(w, s).$$

- Расстояние дальнего соседа (при $\alpha_U = \alpha_V = \frac{1}{2}$, $\beta = 0$, $\gamma = \frac{1}{2}$):

$$R^D(W, S) = \max_{w \in W, s \in S} \rho(w, s).$$

- Среднее расстояние (при $\alpha_U = \frac{|U|}{|W|}$, $\alpha_V = \frac{|V|}{|W|}$, $\beta = \gamma = 0$):

$$R^C(W, S) = \frac{1}{|W||S|} \sum_{w \in W} \sum_{s \in S} \rho(w, s).$$

- Расстояние между центрами кластеров (при $\alpha_U = \frac{|U|}{|W|}$, $\alpha_V = \frac{|V|}{|W|}$, $\beta = -\alpha_U \alpha_V$, $\gamma = 0$):

$$R^{II}(W, S) = \rho^2 \left(\sum_{w \in W} \frac{w}{|W|}, \sum_{s \in S} \frac{s}{|S|} \right).$$

- Расстояние Уорда (при $\alpha_U = \frac{|S|+|U|}{|S|+|W|}$, $\alpha_V = \frac{|S|+|V|}{|S|+|W|}$, $\beta = \frac{-|S|}{|S|+|W|}$, $\gamma = 0$):

$$R^{II}(W, S) = \frac{|S||W|}{|S| + |W|} \rho^2 \left(\sum_{w \in W} \frac{w}{|W|}, \sum_{s \in S} \frac{s}{|S|} \right).$$

2.3.4. Дендрограмма

Построение дендрограммы — очень удобный способ визуализировать иерархическую кластеризацию. Расстояние между кластерами на дендрограмме изображаются как высота дуги, которой соединяются метки кластеров (см. на рисунке выше). Также наглядно будет построить график зависимости расстояния между сливаемыми кластерами от номера итерации.

2.3.5. Пример: дендрограмма в задаче кластеризации писем по теме

Далее представлена дендрограмма построенная на реальных данных в задаче кластеризации писем по теме.

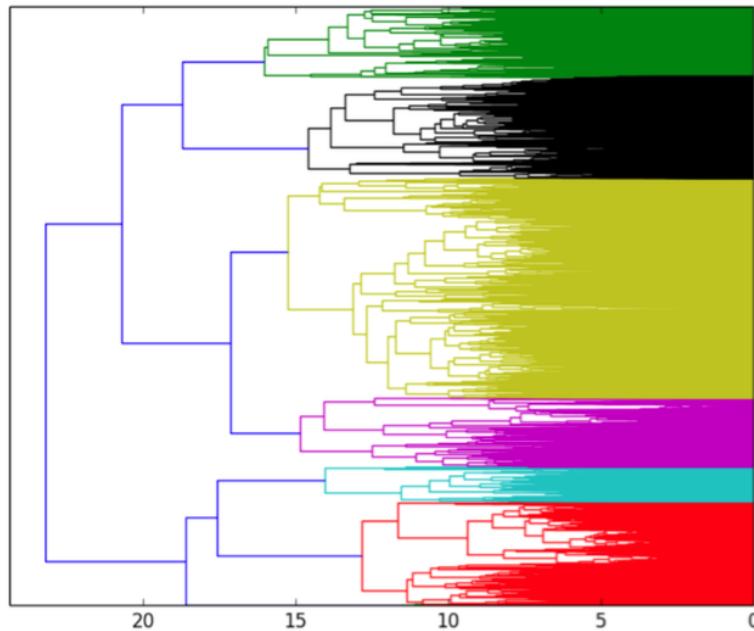


Рис. 2.6: Дендрограмма в задаче кластеризации писем по теме

Если построить дендрограмму для небольшой выборки (100 писем), то она будет иметь вид:

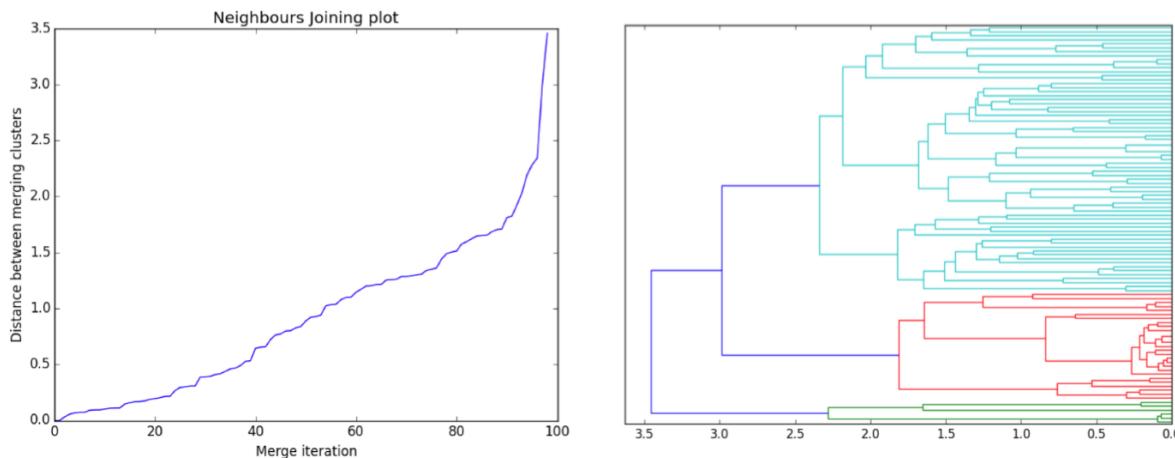


Рис. 2.7: Дендрограмма в задаче кластеризации писем по теме (на подвыборке из 100 писем) и график зависимости расстояния между сливаемыми кластерами от номера итерации

На представленном графике ближе к последним итерациям расстояние между кластерами быстро взмывает вверх, то есть существует некоторое разумное количество кластеров, которые друг от друга более-менее удалены.

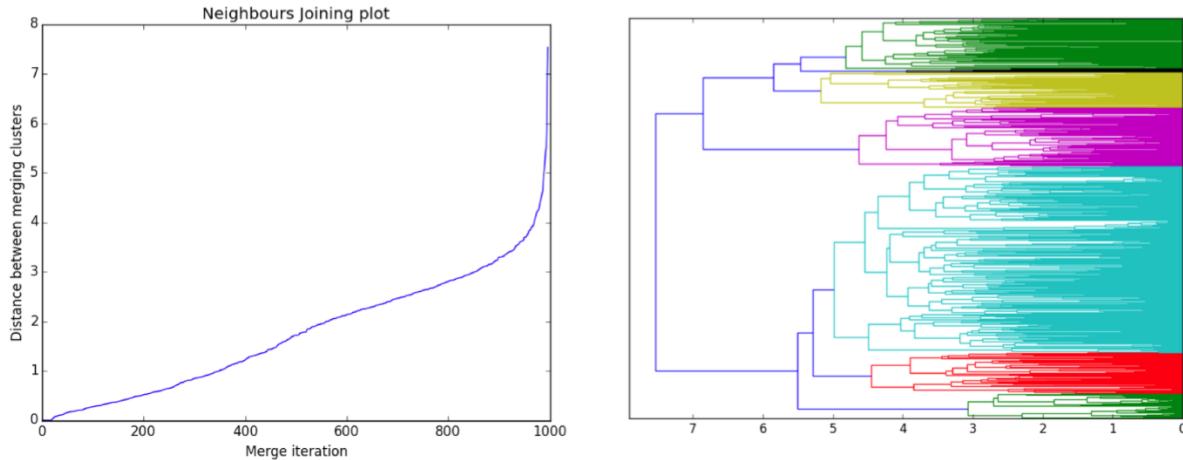


Рис. 2.8: Дендрограмма в задаче кластеризации писем по теме (на подвыборке из 1000 писем) и график зависимости расстояния между сливаемыми кластерами от номера итерации

На большей подвыборке из 1000 писем график стал более гладким, поскольку итерации стало больше, а дендрограмма получилась значительно сложнее.

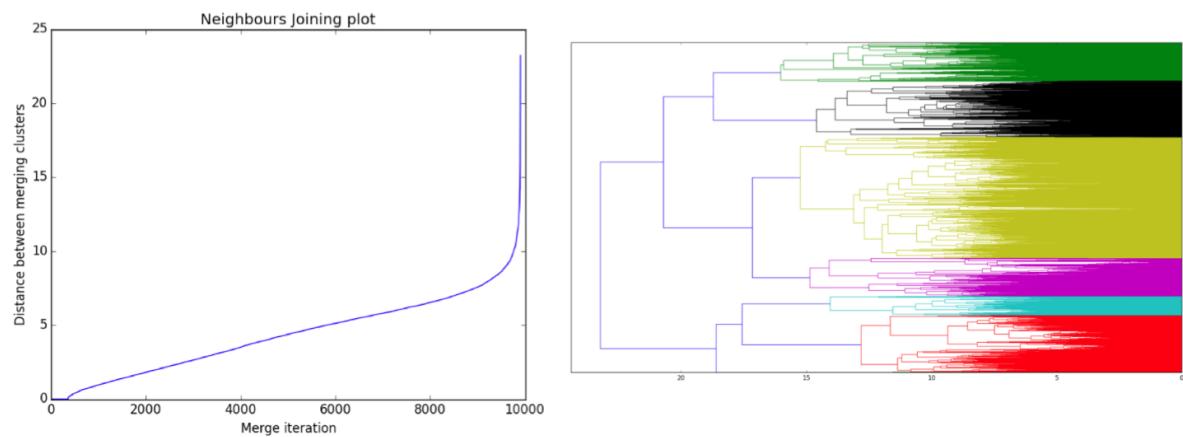


Рис. 2.9: Дендрограмма в задаче кластеризации писем по теме (на подвыборке из 10000 писем) и график зависимости расстояния между сливаемыми кластерами от номера итерации

На самой большой выборке из 10000 писем на графике есть область особенно быстрого роста. Любопытно, что изгиб на графике, за которым происходит особенно быстрый рост, возникал при разном расстоянии между сливаемыми кластерами на всех трех представленных графиках. Это связано с тем, что при исследовании разных подвыборок получается разный набор признаков, а следовательно сравнивать такие расстояния в экспериментах на разных подвыборках некорректно.

2.3.6. Перекос в размерах кластеров

Другая интересная особенность иерархической кластеризации состоит в том, что часто возникает один большой кластер и несколько небольших. Во многих задачах желательно получать кластеры более-менее похожие по размеру.

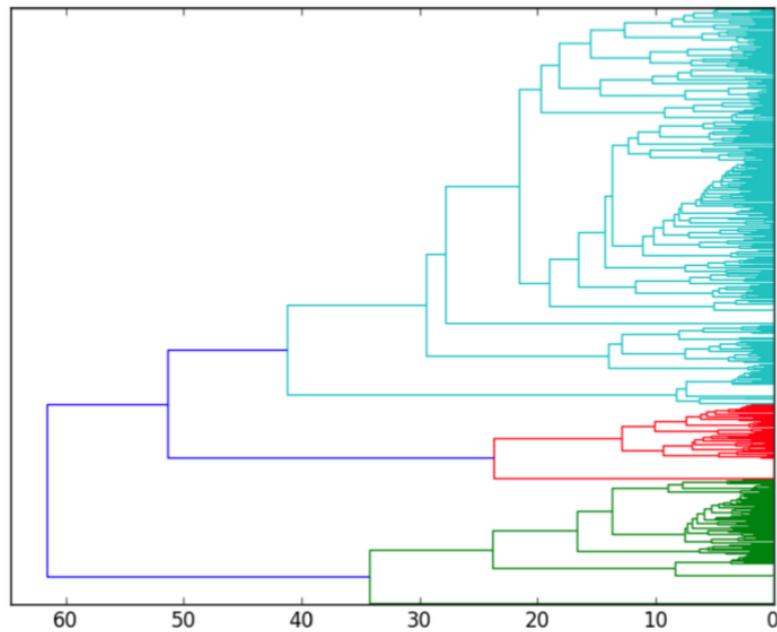


Рис. 2.10: Дендрограмма в задаче кластеризации текстов.

Это может быть связано с тем, что в признаках содержится слишком много шума. В этом случае можно попробовать понизить размерность пространства признаков, например с помощью SVD.

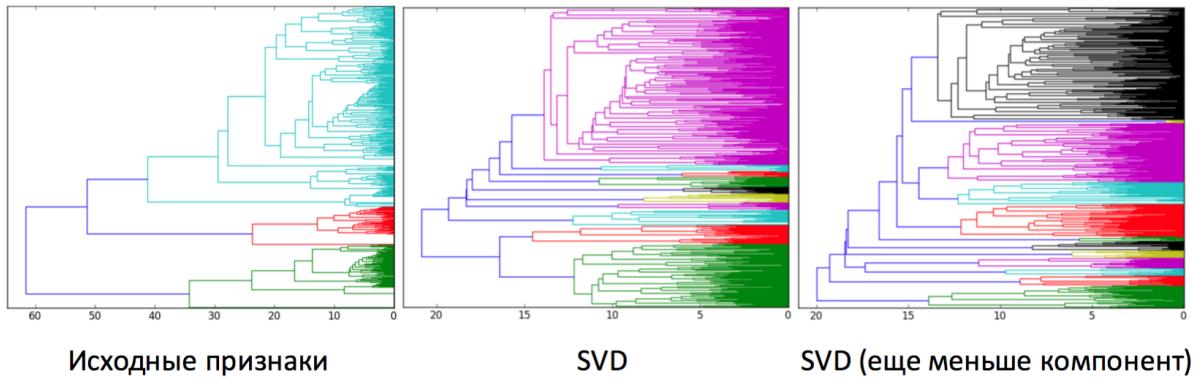


Рис. 2.11: Дендрограммы после понижения размерности признаков с помощью SVD

Если уменьшить размерность слишком сильно, то перегиб на графике зависимости расстояния между сливающимися кластерами от номера итерации пропадает совсем.

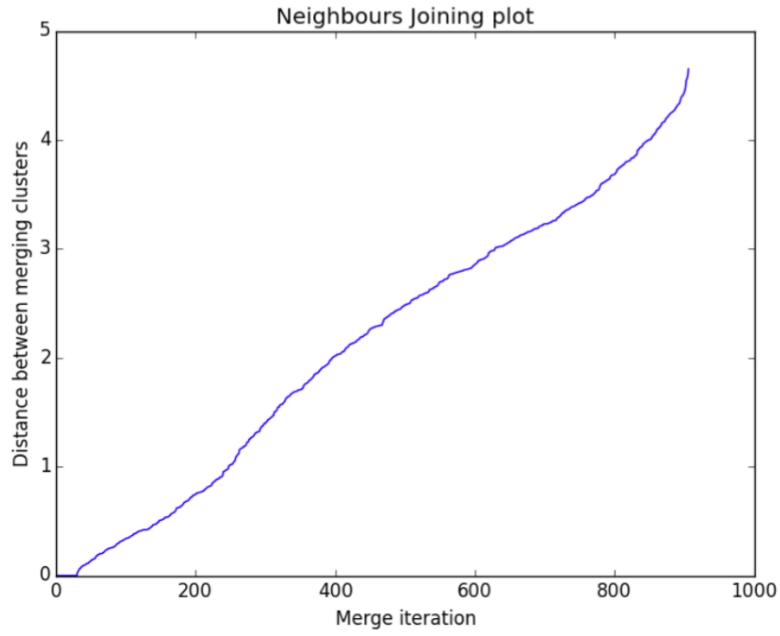


Рис. 2.12: График зависимости расстояния между сливаемыми кластерами от номера итерации

Это значит, что явно выделить кластеры уже нельзя — слишком сильно была понижена размерность пространства признаков.

2.4. Графовые методы кластеризации

2.4.1. Выделение связных компонент

Связность — это свойство, заключающееся в том, что из любой вершины графа можно попасть в любую другую вершину графа по ребрам. Связные компоненты — это подграфы в графе, которые обладают свойством связности, и в то же время никакие вершины из графа нельзя добавить в этот подграф, сохранив свойства связности.

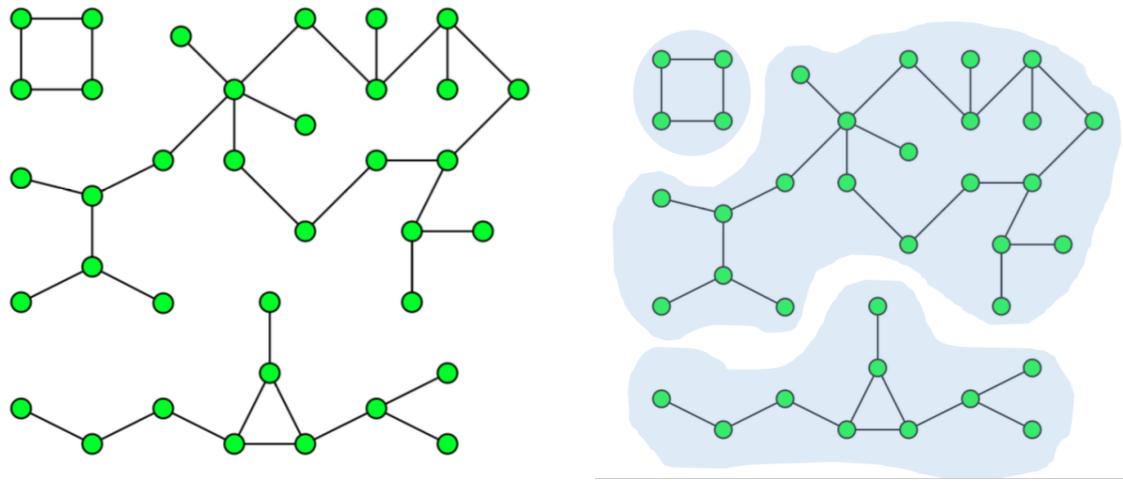


Рис. 2.13: Выделенные связные компоненты графа.

Граф, таким образом, может иметь одну или несколько связных компонент.

2.4.2. Кластеризация по компонентам связности

Кластеризация по компонентам связности происходит следующим образом: соединяются ребрами те объекты, расстояние между которыми меньше R , а затем в получившемся графе выделяются компоненты связности. Если граф получился связный (то есть компонента связности единственна), следует взять меньшее значение R . Однако непонятно, какое значение R нужно выбрать, чтобы получить конкретное значение числа кластеров K . Решить эту проблему позволяет другой простой графовый подход.

2.4.3. Минимальное оствовное дерево

Остовным деревом называется такой связный граф без циклов (дерево), в который входя все вершины исходного графа.

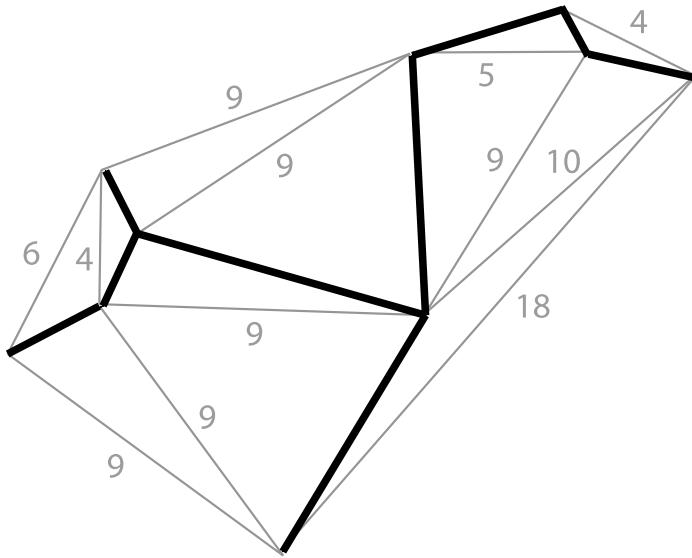


Рис. 2.14: Минимальное оствовное дерево

Минимальное оствовное дерево в связанным взвешенном неориентированном графе — это оствовное дерево этого графа, имеющее минимальный возможный вес, где под весом дерева понимается сумма весов входящих в него рёбер.

Минимальное оствовное дерево можно построить с помощью алгоритма Крускала (Kruskal):

1. Вначале текущее множество рёбер устанавливается пустым.
2. Пока это возможно, проводится следующая операция: из всех рёбер, добавление которых к уже имеющемуся множеству не вызовет появление в нём цикла, выбирается ребро минимального веса и добавляется к уже имеющемуся множеству.
3. Когда таких рёбер больше нет, алгоритм завершён.

Доказательство корректности алгоритма в данном курсе не приводится.

Чтобы решить задачу кластеризация с помощью оствовного дерева, нужно построить взвешенный граф, в котором вершины — это объекты, а веса ребер — это расстояния между объектами. Если необходимо построить K классов, то необходимо удалить $K - 1$ ребро с максимальными весами. Получившийся граф будет состоять из K компонент связности, каждую из которых можно интерпретировать как кластер.

2.5. Методы, основанные на плотности

2.5.1. Идея density-based методов

Идея density-based методов заключается в том, чтобы рассматривать плотность точек в окрестности каждого объекта выборки. Если в окрестности радиуса R с центром в некоторой точке выборки находится N или более других точек выборки, то такая точка считается основной. Здесь R и N — параметры алгоритма. Если точек меньше, чем N , но в окрестности рассматриваемой точки содержится основная точка, то такая точка называется граничной. В ином случае точка считается шумовой.

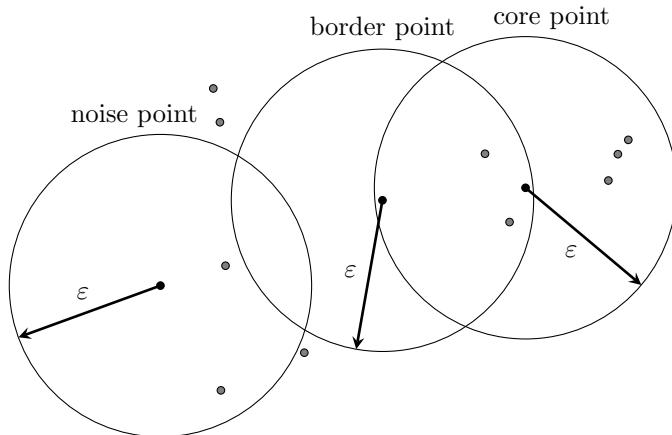


Рис. 2.15: Принцип работы алгоритма DBSCAN: изображены основная, пограничная и шумовая точки

2.5.2. Алгоритм DBSCAN

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. В результате получается граф.
4. Каждую группу соединенных основных точек объединить в свой кластер (то есть выделить связные компоненты в получившемся графе).
5. Отнести пограничные точки к соответствующим им кластерам.

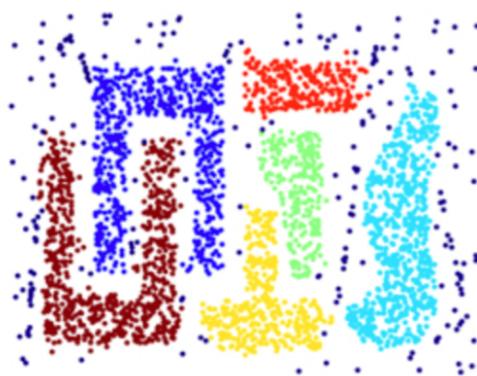


Рис. 2.16: Пример работы алгоритма DBSCAN

Как видно на последнем примере, DBSCAN хорошо справляется с нетривиальными формами кластеров и успешно отделяет шумовые точки, которые могли бы сильно испортить работу других алгоритмов кластеризации. Но DBSCAN часто неправильно определяет количество кластеров, особенно когда несколько кластеров расположены слишком близко друг к другу.

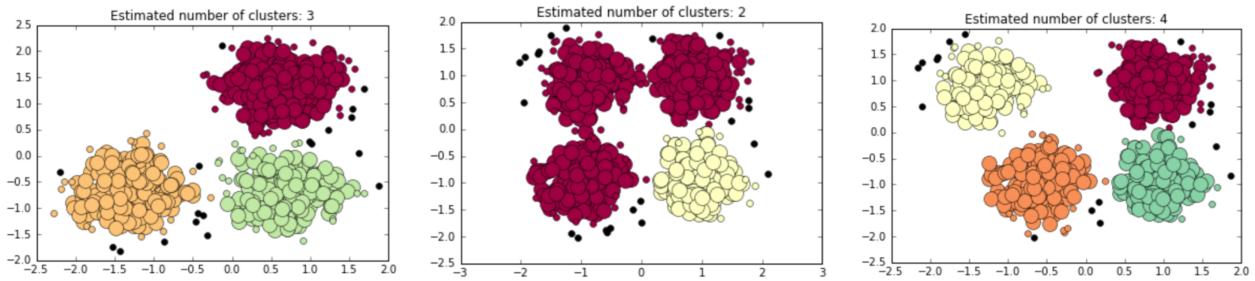
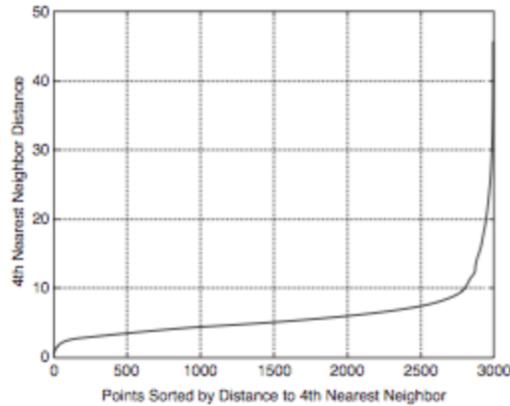


Рис. 2.17: Алгоритм DBSCAN правильно определил число кластеров (рис. слева). Ситуация, когда 3 из 4 кластеров слились воедино (рис. по центру). Эту ситуацию можно было бы избежать, сдвинув один из кластеров влево (рис. справа).

2.5.3. Подбор параметров в DBSCAN

Чтобы подобрать параметры в методе DBSCAN, имеет смысл построить график, по оси y у которого отложено расстояние до k -го соседа, а по оси x — количество точек, расстояние до k -го соседа которых меньше.



Начиная с некоторого номера на этом графике происходит резкий рост расстояния, а значит можно естественным способом выбрать расстояние, выбираемое в качестве радиуса R окрестности. Число k в данном случае есть пороговое количество точек в окрестности некоторой точки, необходимое, чтобы считать ее основной.

Таким образом, подбор параметров в методе DBSCAN заключается в рассмотрении графиков для разных значений k и определении оптимального значения R для каждого из них. Выбрать следует такую пару параметров k и R , чтобы количество шумовых точек было минимальным.

2.6. Оценка качества и рекомендации по решению задачи кластеризации

2.6.1. Внутрикластерное и межкластерное расстояния

Среднее внутрикластерное расстояние имеет следующий вид:

$$F_0 = \frac{\sum_{i < j} [y_i = y_j] \rho(x_i, x_j)}{\sum_{i < j} [y_i = y_j]} \rightarrow \min.$$

Если известны положения центров кластеров, можно записать аналогичную метрику, которая представляет собой среднее расстояние до центра кластера:

$$\Phi_0 = \sum_{y \in Y} \frac{1}{|K_y|} \sum_{i:y_i=y} \rho^2(x_i, \mu_y) \rightarrow \min.$$

Абсолютно аналогично вводится среднее межкластерное расстояние (теперь объекты берутся из разных кластеров):

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

Если известны центры кластеров, в качестве более простого варианта этой метрики можно взять (где μ — среднее арифметическое центров кластеров):

$$\Phi_1 = \sum_{y \in Y} \rho^2(\mu_y, \mu) \rightarrow \max.$$

Учесть значения обоих функционалов в одной метрике можно, если рассмотреть частное:

$$F_0/F_1 \rightarrow \min, \quad \text{или} \quad \Phi_0/\Phi_1 \rightarrow \min.$$

Но так или иначе эти функционалы имеют ряд недостатков. Например, с помощью таких функционалов нельзя подобрать количество кластеров. Действительно, при использовании таких метрик лучшим вариантом будет отнести каждую точку к своему собственному кластеру, так как в этом случае среднее внутрикластерное расстояние будет равно нулю.

2.6.2. Коэффициент силуэта для объекта

Коэффициент силуэта — метрика качества, которая позволяет выбрать количество кластеров. Коэффициент силуэта для некоторого фиксированного объекта определяется следующим образом:

$$s = \frac{b - a}{\max(a, b)},$$

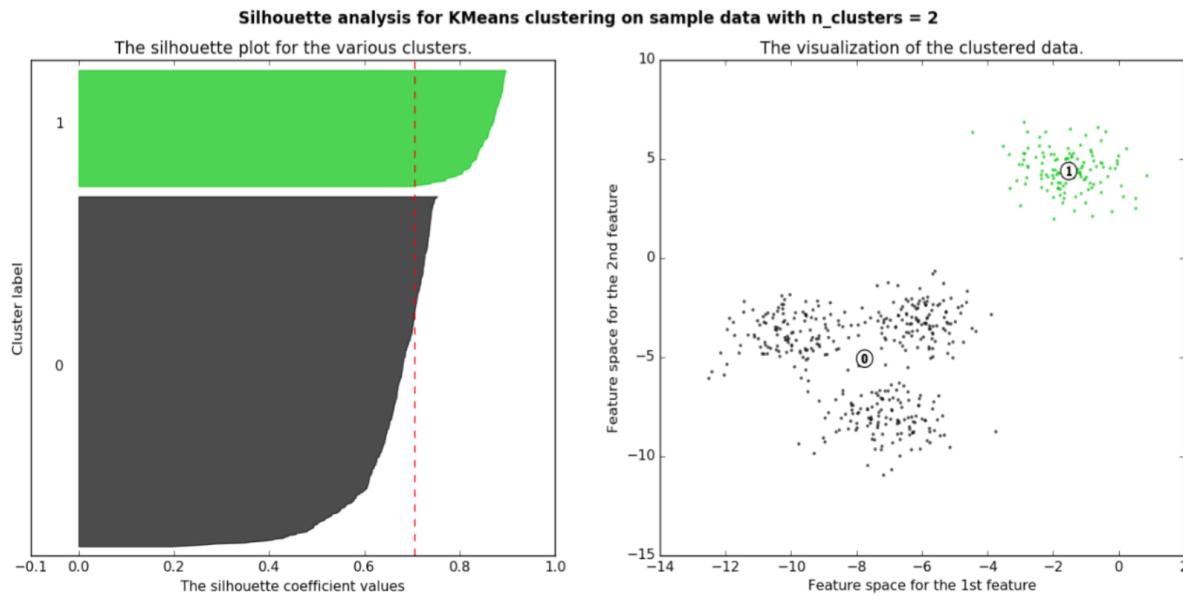
где a — среднее расстояние от данного объекта до других объектов из того же кластера, b — среднее расстояние от данного объекта до объектов из ближайшего другого кластера.

Обычно коэффициент силуэта положителен, но, вообще говоря, может меняться в пределах от -1 до 1 . Если объект находится вблизи границы кластера, и близко к нему расположен другой небольшой кластер, коэффициент силуэта для этого объекта получится очень маленьким.

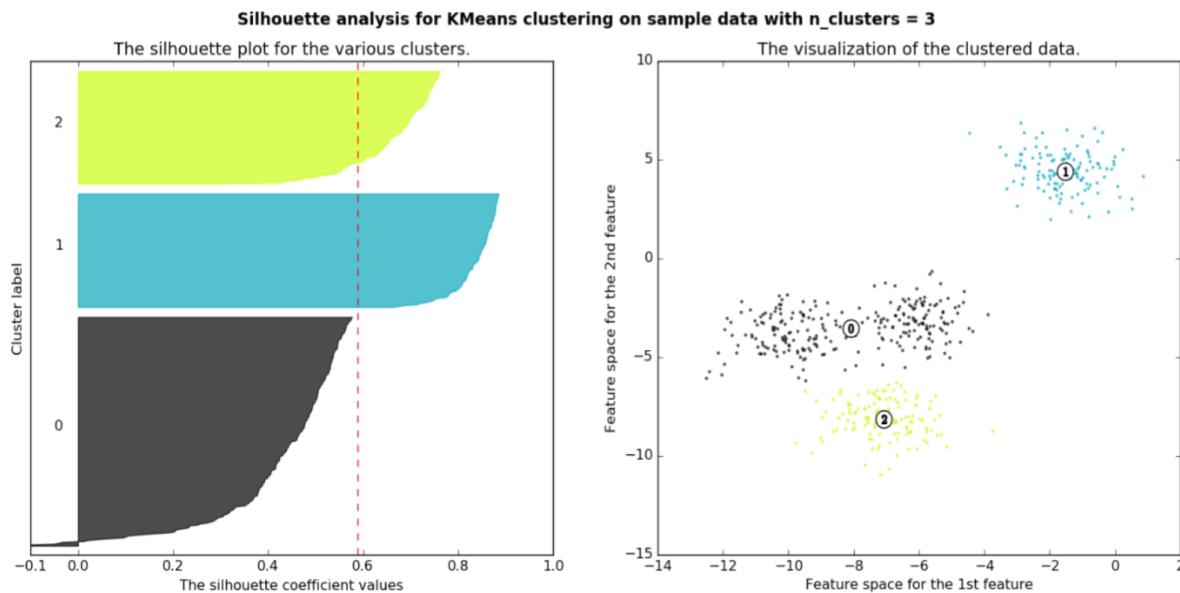
2.6.3. Использование коэффициента силуэта для оценки числа кластеров

Обычно вычисляют среднее значение коэффициента силуэта, а также для каждого кластера строят график, показывающий количество точек с различными значениями коэффициента силуэта. Среднее значение на таких графиках отмечается пунктирной линией. Для наглядности выборка будет изображаться справа от графика.

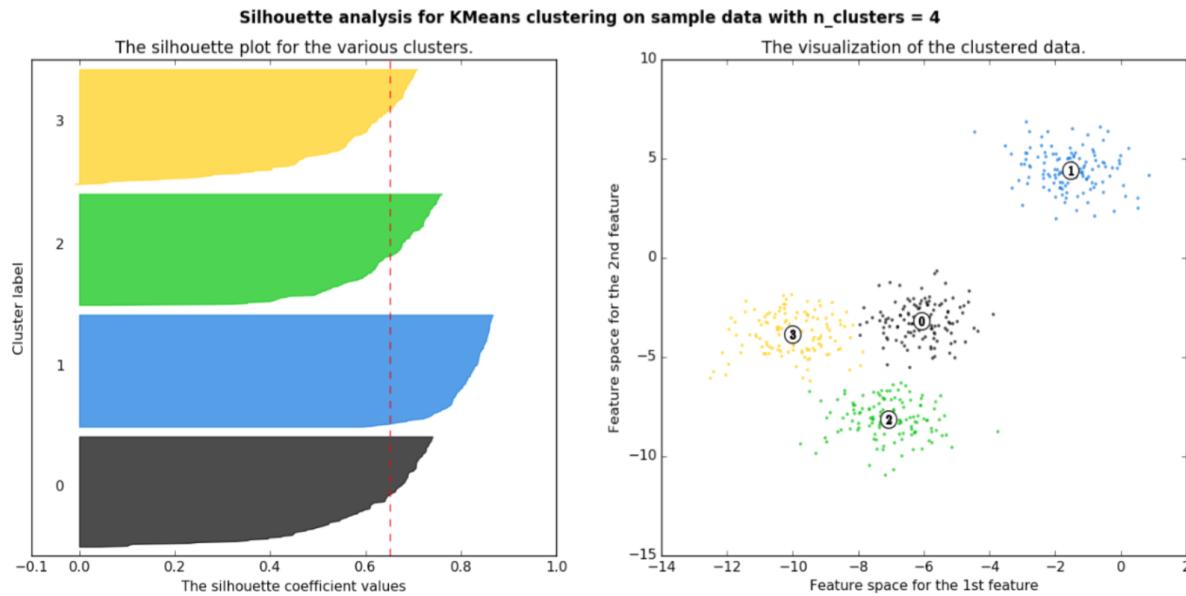
В случае двух кластеров в обоих из них есть точки, коэффициент силуэта для которых больше его среднего значения по всей выборке. Так же разброс значений по кластерам не очень большой.



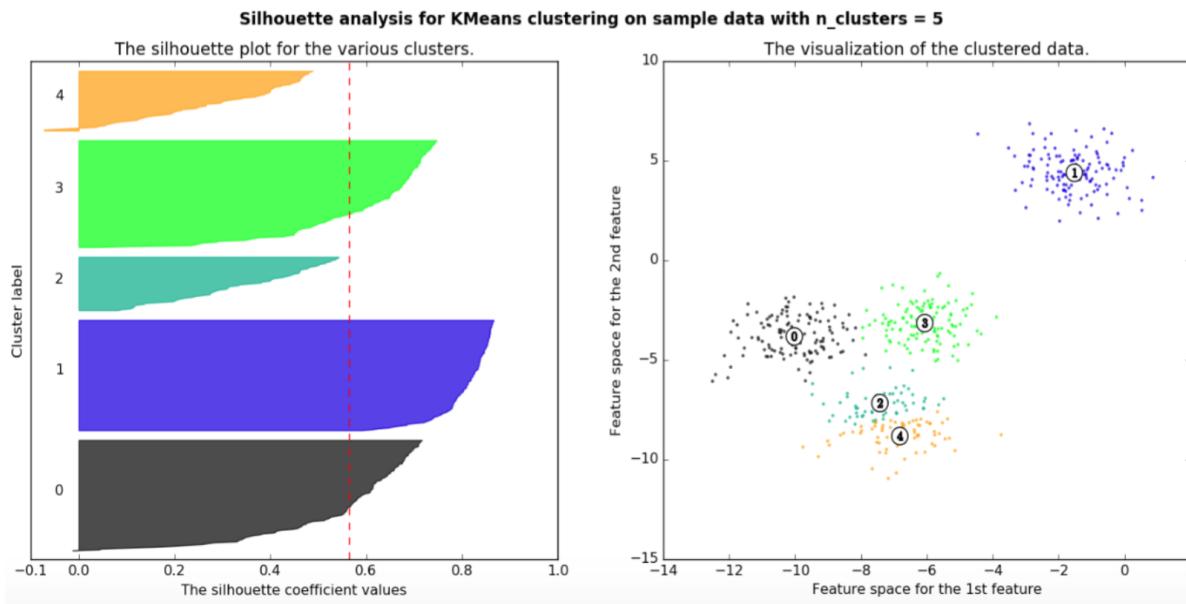
В случае трех кластеров разброс значений коэффициента силуэта в различных кластерах уже большой. При этом в третьем кластере все значения оказались меньше среднего значения по всей выборке. Такая кластеризация выглядит не очень убедительно и такое число кластеров лучше не выбирать.



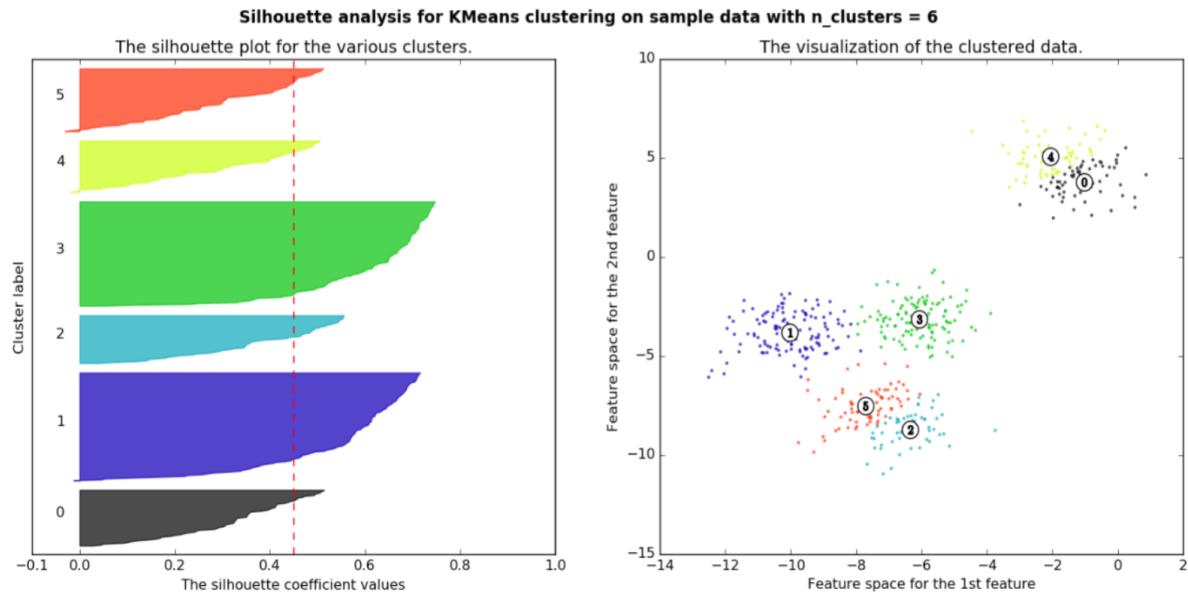
В случае четырех кластеров разброс значений коэффициента силуэта в различных кластерах небольшой, а также во всех кластерах есть точки, значения коэффициентов силуэта в которых больше среднего значения по выборке. Кластеризация с таким числом кластеров выглядит очень хорошо, в чем можно убедиться по визуализации на выборке. В реальных задачах часто признаков очень много и так просто визуализировать на плоскости не получается.



Если попробовать взять число кластеров равным пяти, то разброс по кластерам становится еще большие и опять же есть кластеры, у которых коэффициент меньше среднего значения по всей выборке.

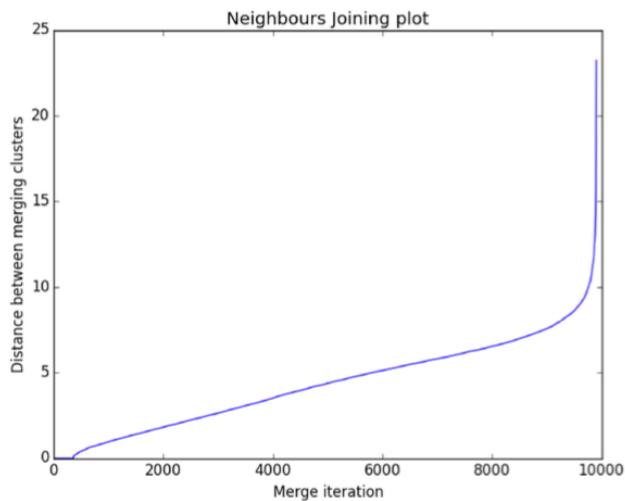


И в случае шести кластеров опять же получаем довольно большой разброс.



2.6.4. Проверка наличия кластерной структуры

Также полезно уметь проверять наличие кластерной структуры. Один из способов это проверить уже был озвучен — необходимо построить график расстояния, на котором происходит слияние, от номера итерации при агломеративной иерархической кластеризации.



В произвольном случае поступают следующим образом: генерируют p случайных точек из равномерного распределения и p случайных точек из обучающей выборки. После этого вычисляется так называемая статистика Хопкинса:

$$H = \frac{\sum_{i=1}^p w_i}{\sum_{i=1}^p u_i + \sum_{i=1}^p w_i},$$

w_i — расстояние от i -ой случайной точки до ближайшей случайной, u_i — расстояние от i -ой точки из выборки до другой ближайшей точки из выборки. Если статистика получается близкой к $1/2$, это значит, что выборка более-менее равномерно заполняет пространство признаков. Если же статистика получается вблизи нуля, это означает, что точки как-то группируются.

2.6.5. Выбор признаков

Кроме того, нужно уметь выбирать хорошие признаки для задачи кластеризации. Все метрики, которые обсуждались ранее, используют расстояние, которое зависит от выбора признаков. Поэтому использовать такие метрики для выбора признаков не получается. Вообще говоря, хотелось бы иметь возможность сравнивать качество кластеризации в зависимости от выбора признаков.

Пусть известна разметка, то есть к каким классам можно было бы отнести объекты выборки, причем этой разметки недостаточно для обучения классификатора. Тогда ее можно использовать для оценки качества кластеризации, например использовать метрику точности (accuracy). Другой подход состоит в использовании однородности, полноты и V-меры:

$$h = 1 - \frac{H(C|K)}{H(C)}, \quad c = 1 - \frac{H(K|C)}{H(K)}, \quad v = 2 \cdot \frac{h \cdot c}{h + c},$$

где $H(C)$ — энтропия класса, $H(K)$ — энтропия кластера, $H(C|K)$ — энтропия класса при условии кластера, $H(K|C)$ — энтропия кластера при условии класса. V-мера, как это видно из выражения, представляет собой среднее гармоническое однородности и полноты.

Однородность будет максимальной, если кластер состоит только из объектов одного класса, а полнота — если все объекты из класса принадлежат к одному кластеру.

Энтропия — мера неопределенности, мера незнания о том, какая будет конкретная реализация случайной величины:

$$H = - \sum_i p_i \log p_i.$$

Энтропия для класса вычисляется следующим образом:

$$H(C) = - \sum_{c=1}^{|C|} \frac{n_c}{n} \log \left(\frac{n_c}{n} \right), \quad P(c) = \frac{n_c}{n}.$$

Энтропия для класса при условии кластера вычисляется следующим образом:

$$H(C|K) = - \sum_{c=1}^{|C|} \sum_{k=1}^{|K|} \frac{n_{c,k}}{n} \log \left(\frac{n_{c,k}}{n_k} \right), \quad P(c|k) = \frac{n_{c,k}}{n}.$$

Если классы идеально совпадают с кластерами, то соответствующие условные энтропии $H(K|C)$ и $H(C|K)$ будут равняться нулю, а следовательно $h = c = 1$.

2.6.6. Привлечение ассессоров для оценки качества

Если разметки, о которой шла речь выше, нет, то можно:

- Использовать метрики без разметки
- Создать разметку с помощью ассессоров и использовать ее
- Предложить ассессорам отвечать на вопросы вида «допустимо ли эти объекты относить в один/разные кластеры». И, используя их ответы на уже готовой кластеризации, оценить ее качество.

Урок 3

Понижение размерности и отбор признаков

3.1. Зачем отбирать признаки?

3.1.1. Задача отбора признаков

Эта неделя посвящена отбору признаков и понижению размерности. В первую очередь необходимо разобраться, зачем работать с признаками и отбирать их.

Во-первых, признаков может быть слишком много, больше чем нужно. Это может возникнуть в ситуациях, когда используется вся имеющаяся на данный момент информация, потому что неизвестно, какая её часть может понадобиться, а какая — нет. В таких случаях можно повысить качество решения задачи, выбирая только действительно важные признаки. Существует другой подход: можно сформировать новые признаки на основе старых, таким образом признаков станет меньше, но их информативность сохранится.

Во-вторых, существуют признаки, из-за которых при решении задачи возникает много проблем. Это шумовые признаки — признаки, которые не связаны с целевой переменной и никак не относятся к решаемой задаче. К сожалению, не всегда можно понять по обучающей выборке, что в ней присутствуют такие признаки.

Полезно рассмотреть несколько примеров присутствия шумовых признаков в данных. Пусть в выборку добавляют 1000 признаков. Значения каждого признака генерируются из стандартного нормального распределения. Понятно, что эти признаки бесполезны, они никак не помогут решить задачу. Но, поскольку их много, может так оказаться (из соображений теории вероятностей), что один из них коррелирует с целевой переменной. При этом он будет коррелировать только на обучающей выборке, а на контрольной выборке корреляции наблюдаться не будет, поскольку признак абсолютно случайный. Однако внутри модели этот признак может быть учтён как важный и иметь какой-то вес. Получается, что модель зависит от признака, который никак не помогает решить задачу. Из-за этого качество модели и ее обобщающая способность окажутся ниже, чем хотелось бы.

Другой пример. Пусть для решения задачи используется решающее дерево. В выборке присутствуют 1000 признаков, все они информативные. Но при этом, чтобы учесть каждый из них хотя бы один раз, понадобится дерево глубины как минимум 10. У этого дерева будет около 1000 листьев. Для того чтобы построить хорошие прогнозы и избежать переобучения, необходимо, чтобы в каждый лист попало большое количество примеров, а значит обучающая выборка должна быть очень большой. При этом нужно обратить внимание на следующее: такое дерево учит кажды признак один раз. Для того чтобы учесть признаки большее число раз, понадобится более глубокое дерево, и необходимый для обучения такого алгоритма размер выборки увеличится ещё сильнее.

Еще одна причина, по которой может понадобиться отбирать признаки — это ускорение модели. Дело в том, что чем больше признаков, тем более сложная модель получается, и тем больше времени необходимо, чтобы построить прогноз. Существуют задачи, в которых прогнозы нужно строить очень быстро, например, выдача рекомендаций товаров на сайте интернет-магазина. Пользователь что-то ищет, нажимает на ссылку в поисковой выдаче и переходит на страницу интересующего его товара. На этой странице есть поле, в котором показываются рекомендации к этому товару, например похожие товары, которые должна выдавать модель. Важно, чтобы она выдавала рекомендации очень быстро, страница не должна долго загружаться, чтобы пользователь не подумал, что с сайтом что-то не так, и не ушел к конкуренту. В этом случае необходимо,

чтобы модель была очень быстрой, и один из подходов к ускорению модели — это отбор признаков, которых достаточно, чтобы прогнозы были хорошими.

3.1.2. Метода отбора признаков

Опишем некоторые подходы к отбору признаков. Самый простой — это одномерный подход. В нём оценивается связь каждого признака с целевой переменной, например, измеряется корреляция. Такой подход — довольно простой, он не учитывает сложные закономерности, в нём все признаки считаются независимыми, тогда как в машинном обучении модели учитывают взаимное влияние признаков, их пар или даже более сложные действия на целевую переменную. Этот подход не всегда хорош, но иногда его можно использовать, чтобы ранжировать признаки, найти наиболее мощные среди них.

Более сложные подходы к отбору признаков могут быть устроены следующим образом: некоторым способом перебираются различные подмножества признаков, для каждого такого подмножества обучается модель, которая использует только эти признаки, и оценивается ее качество. В итоге выбирается то подмножество, которое дает наилучшее качество. Такие подходы оказываются сложными из-за того, что нужно перебирать подмножества признаков. Понятно, что всех подмножеств очень много, поэтому поиск должен быть более направленным.

Можно использовать саму модель, чтобы оценивать важность признаков и отбирать их. Например, в курсе уже шла речь о методе Lasso, L_1 -регуляризации, которая позволяет отбирать признаки с помощью линейных моделей. Похожие методы используются и в решающих деревьях, случайном лесе и градиентном бустинге — об этом будет рассказано далее.

Наконец, может сложиться ситуация, когда все признаки важны, но их слишком много. В этом случае используются методы понижения размерности — такое построение новых признаков на основе старых, чтобы при этом сохранялось максимальное количество информации из исходных признаков.

3.2. Одномерный отбор признаков

Самые простые и наивные методы отбора признаков — это одномерные методы, о них пойдёт речь в этой части.

3.2.1. Постановка задачи

Необходимо ввести следующие обозначения:

- x_{ij} — значение признака j на объекте i ,
- \bar{x}_j — среднее значение признака j по всей выборке,
- y_i — значение целевой переменной или ответа на объекте i ,
- \bar{y} — среднее значение целевой переменной на всей выборке.

Задача — оценить предсказательную силу (информативность) каждого признака, то есть насколько хорошо по данному признаку можно предсказывать целевую переменную. Данные оцененной информативности можно использовать, чтобы отобрать k лучших признаков или признаки, у которых значение информативности больше порога (например, некоторой квантили распределения информативности).

3.2.2. Отбор признаков с использованием корреляции

Один из самых простых методов измерения связь между признаком и ответами — это корреляция:

$$R_j = \frac{\sum_{i=1}^{\ell} (x_{ij} - \bar{x}_j)(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{\ell} (x_{ij} - \bar{x}_j)^2 \sum_{i=1}^{\ell} (y_i - \bar{y})^2}}$$

Чем больше по модулю корреляция между признаком и целевой переменной, тем более информативным является данный признак. При этом она максимальна по модулю ($R_j = \pm 1$), если между признаком и целевой

переменной есть линейная связь, то есть если целевую переменную можно строго линейно выразить через значение признака. Это означает, что корреляция измеряет только линейную информативность, то есть способность признака линейно предсказывать целевую переменную. Вообще говоря, корреляция рассчитана на вещественные признаки и вещественные ответы. Тем не менее, её можно использовать в случае, если признаки и ответы бинарные (имеет смысл кодировать бинарный признак с помощью значений ± 1).

3.2.3. Использование бинарного классификатора для отбора признаков

Пусть решается задача бинарной классификации, и необходимо оценить важность признака j для решения именно этой задачи. В этом случае можно попробовать построить классификатор, который использует лишь этот один признак j , и оценить его качество. Например, можно рассмотреть очень простой классификатор, который берёт значение признака j на объекте, сравнивает его с порогом t , и если значение меньше этого порога, то он относит объект к первому классу, если же меньше порога — то к другому, нулевому или минус первому, в зависимости от того, как мы его обозначили. Далее, поскольку этот классификатор зависит от порога t , то его качество можно измерить с помощью таких метрик как площадь под ROC-кривой или Precision-Recall кривой, а затем по данной площади отсортировать все признаки и выбирать лучшие.

3.2.4. Использование метрик теории информации для отбора признаков

Существует ещё один подход в одномерном оценивании качества признаков, который основан на метриках, используемых в теории информации. Примером такой метрики является взаимная информация, или mutual information. Она рассчитана на случай, когда и признак, и целевая переменная — дискретные.

Пусть необходимо решить задачу многоклассовой классификации. В этом случае целевая переменная принимает m различных значений:

$$1, 2, \dots, m,$$

а признак — n значений:

$$1, 2, \dots, n.$$

Поскольку для метрики взаимной информации не важны конкретные значения признаков, можно обозначать их с помощью натуральных чисел. Вероятностью некоторого события будем обозначать долю объектов, для которых это событие выполнено. Например, вероятность того, что признак принимает значение v , а целевая переменная — k , вычисляется следующим образом:

$$P(x_j = v, y = k) = \frac{1}{\ell} \sum_{i=1}^{\ell} [x_{ij} = v][y_i = k]$$

Или, например, вероятность того, что признак равен v :

$$P(x_j = v) = \frac{1}{\ell} \sum_{i=1}^{\ell} [x_{ij} = v]$$

Взаимная информация между признаком j и целевой переменной вычисляется по следующей формуле:

$$MI_j = \sum_{v=1}^n \sum_{k=1}^m P(x_j = v, y = k) \log \frac{P(x_j = v, y = k)}{P(x_j = v)P(y = k)}$$

Главная особенность взаимной информации состоит в следующем: она равна нулю, если признак и целевая переменная независимы. Если же между ними есть какая-то связь, то взаимная информация будет отличаться от нуля, причём она может быть как больше, так и меньше нуля. Это означает, что информативность признаков нужно оценивать по модулю взаимной информации.

3.2.5. Проблемы одномерного отбора признаков

У подхода, при котором важности всех признаков оцениваются по отдельности, есть свои недостатки. На рисунке 3.1 изображена двумерная выборка, для которой необходимо решить задачу классификации. Если спроектировать данную выборку на ось абсцисс, то она будет разделима, хотя и будут присутствовать ошибки. Если же спроектировать данную выборку на ось ординат, то все объекты разных классов перемешаются,

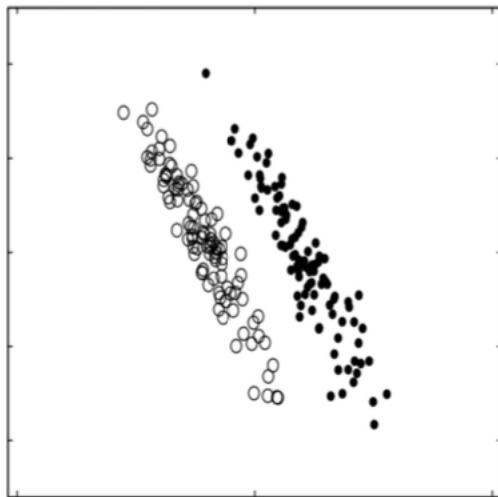


Рис. 3.1: Пример выборки, для которой необходимо решить задачу классификации

и выборка будет неразделима. В этом случае при использовании любого метода одномерного оценивания информативности первый признак будет информативен, а второй — совершенно неинформативен. Тем не менее, видно, что если использовать эти признаки одновременно, то классы будут разделимы идеально. На самом деле, второй признак важен, но он важен только в совокупности с первым, и методы одномерного оценивания информативности не способны это определить.

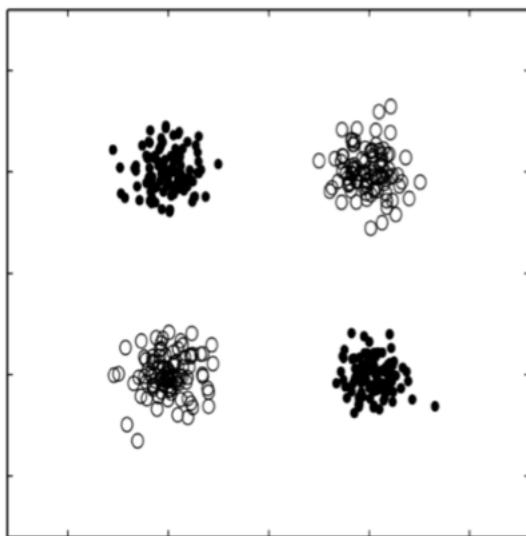


Рис. 3.2: Пример выборки, для которой необходимо решить задачу классификации

На рисунке 3.2 показана выборка, на которой одномерные методы оценки информативности работают ещё хуже. В этом случае, если спроецировать выборку на ось абсцисс или ординат, то объекты классов перемешиваются, и в обоих случаях данные будут совершенно неразделимы. И согласно любому из описанных методов, оба признака неинформативны. Тем не менее, если использовать их одновременно, то, например,

решающее дерево может идеально решить данную задачу классификации.

3.3. Жадные методы отбора признаков

3.3.1. Принцип работы жадных методов

Жадные методы отбора признаков, по сути своей, являются надстройками над методами обучения моделей. Они перебирают различные подмножества признаков и выбирают то из них, которое дает наилучшее качество определённой модели машинного обучения.

Данный процесс устроен следующим образом. Обучение модели считается черным ящиком, который на вход принимает информацию о том, какие из его признаков можно использовать при обучении модели, обучает модель, и дальше каким-то методом оценивается качество такой модели, например, по отложенной выборке или кросс-валидации. Таким образом, задача, которую необходимо решить, — это оптимизация функционала качества модели по подмножеству признаков.

3.3.2. Переборные методы

Самый простой способ решения данной задачи — это полный перебор всех подмножеств признаков и оценивание качества на каждом подмножестве. Итоговое подмножество — то, на котором качество модели наилучшее. Этот перебор можно структурировать и перебирать подмножества последовательно: сначала те, которые имеют мощность 1 (наборы из 1 признака), потом наборы мощности 2, и так далее. Это подход очень хороший, он найдет оптимальное подмножество признаков, но при этом очень сложный, поскольку всего таких подмножеств 2^d , где d — число признаков. Если признаков много, сотни или тысячи, то такой перебор невозможен: он займет слишком много времени, возможно, сотни лет или больше. Поэтому такой метод подходит либо при небольшом количестве признаков, либо если известно, что информативных признаков очень мало, единицы.

3.3.3. Метод жадного добавления

Если же признаков много и известно, что многие из них информативны, то нужно применять жадную стратегию. Жадная стратегия используется всегда, когда полный перебор не подходит для решения задачи. Например, может оказаться неплохой стратегия жадного наращивания (жадного добавления). Сначала находится один признак, который дает наилучшее качество модели (наименьшую ошибку Q):

$$i_1 = \operatorname{argmin} Q(i).$$

Тогда множество, состоящее из этого признака:

$$J_1 = \{i_1\}$$

Дальше к этому множеству добавляется еще один признак так, чтобы как можно сильнее уменьшить ошибку модели:

$$i_2 = \operatorname{argmin} Q(i_1, i), \quad J_2 = \{i_1, i_2\}.$$

Далее каждый раз добавляется по одному признаку, образуются множества J_3, J_4, \dots . Если в какой-то момент невозможно добавить новый признак так, чтобы уменьшить ошибку, процедура останавливается. Жадность процедуры заключается в том, что как только какой-то признак попадает в оптимальное множество, его нельзя оттуда удалить.

3.3.4. Алгоритм ADD-DEL

Описанный выше подход довольно быстрый: в нем столько итераций, сколько признаков в выборке. Но при этом он слишком жадный, перебирается слишком мало вариантов. Процедуру можно усложнить. Один из подходов к усложнению — это алгоритм ADD-DEL, который не только добавляет, но и удаляет признаки из оптимального множества. Алгоритм начинается с процедуры жадного добавления. Множество признаков наращивается до тех пор, пока получается уменьшить ошибку, затем признаки жадно удаляются из подмножества, то есть перебираются все возможные варианты удаления признака, оценивается ошибка и удаляется тот признак, который приводит к наибольшему уменьшению ошибки на выборке. Эта процедура повторяет

добавление и удаление признаков до тех пор, пока уменьшается ошибка. Алгоритм ADD-DEL всё еще жадный, но при этом он менее жадный, чем предыдущий, поскольку может исправлять ошибки, сделанные в начале перебора: если вначале был добавлен неинформативный признак, то на этапе удаления от него можно избавиться.

3.4. Отбор признаков на основе моделей

3.4.1. Использование линейных моделей для отбора признаков

Для оценки информативности признаков и их отбора можно использовать обученные модели, например, линейные.

Линейная модель вычисляет взвешенную сумму значений признаков на объекте:

$$a(x) = \sum_{j=1}^d w_j x^j$$

При этом возвращается сама взвешенная сумма, если это задача регрессии, и знак этой суммы, если это задача классификации. Если признаки масштабированы, то веса при признаках можно интерпретировать как информативности: чем больше по модулю вес при признаке j , тем больший вклад этот признак вносит в ответ модели. Однако если признаки не масштабированы, то так использовать веса уже нельзя. Например, если есть два признака, и один по масштабу в 1000 раз меньше другого, то вес первого признака может быть очень большим, только чтобы признаки были одинаковыми по масштабу.

Если необходимо обнулить как можно больше весов, чтобы линейная модель учитывала только те признаки, которые наиболее важны для нее, можно использовать L1-регуляризацию. Чем больше коэффициент при L1-регуляризаторе, тем меньше признаков будет использовать линейная модель.

3.4.2. Применение решающих деревьев для отбора признаков

Еще один вид моделей, обсуждаемых в предыдущем курсе, — это решающие деревья. Решающие деревья строятся «жадно»: они растут от корня к листьям, и на каждом этапе происходит попытка разбить вершину на две. Чтобы разбить вершину, нужно выбрать признак, по которому будет происходить разбиение, и порог, с которым будет сравниваться значение данного признака. Если значение признака меньше этого порога, то объект отправляется в левое поддерево, если больше — в правое поддерево. Выбор признака и порога осуществляется по следующему критерию:

$$Q(X_m, j, t) = \frac{|X_l|}{|X_m|} H(X_l) + \frac{|X_r|}{|X_m|} H(X_r) \rightarrow \min_{j, t},$$

где $H(X)$ — это критерий информативности. Ранее в курсе были рассмотрены различные критерии информативности. Например, в задаче регрессии используется функционал среднеквадратичной ошибки, а в классификации — критерий Джини или энтропийный критерий.

Критерий Q вычисляет взвешенную сумму критериев информативности $H(X)$ в обеих дочерних вершинах — левой и правой. Чем меньше данная взвешенная сумма, тем больше признак j и порог t подходят для разбиения.

Если в данной вершине происходит разбиение по признаку j , то чем сильнее уменьшается значение критерия информативности, тем важнее этот признак оказался при построении дерева. Таким образом, можно оценивать важность признака на основе того, насколько сильно он смог уменьшить значение критерия информативности. Пусть, в вершине m произведено разбиение по признаку j . Тогда можно вычислить уменьшение критерия информативности в ней по следующей формуле:

$$H(X_m) - \frac{|X_l|}{|X_m|} H(X_l) - \frac{|X_r|}{|X_m|} H(X_r)$$

R_j — сумма данного уменьшения по всем вершинам дерева, в которых происходило разбиение по признаку j . Чем больше R_j , тем важнее данный признак был при построении дерева.

3.4.3. Использование композиций алгоритмов для отбора признаков

Сами по себе решающие деревья не очень полезны, но они очень активно используются при построении композиций, в частности, в случайных лесах и в градиентном бустинге над деревьями. В данных композициях измерить важность признака можно аналогичным образом: суммируется уменьшение критерия информативности R_j по всем деревьям композиции, и чем больше данная сумма, тем важнее признак j для композиции. То есть признаки оцениваются с помощью того, насколько сильно они смогли уменьшить значение критерия информативности в совокупности по всем деревьям композиции.

Для случного леса можно предложить еще один интересный способ оценивания информативности признаков. В этой композиции каждое базовое дерево b_n обучается по подмножеству объектов обучающей выборки. Таким образом, есть объекты, на которых дерево не обучалось, и набор этих объектов является валидационной выборкой для дерева n . Такая выборка называется out-of-bag. Итак, метод заключается в следующем: ошибку Q_n базового дерева b_n оценивают по out-of-bag-выборке и запоминают. После этого признак j превращают в абсолютно бесполезный, шумовой: в матрицу «объекты-признаки» все значения в столбце j перемешивают. Затем то же самое дерево b_n применяют к данной выборке с перемешанным признаком j и оценивают качество дерева на out-of-bag-подвыборке. Q'_n — ошибка out-of-bag-подвыборке, она будет тем больше, чем сильнее дерево использует признак j . Если он активно используется в дереве, то ошибка сильно уменьшится, поскольку значение данного признака испорчено. Если же данный признак совершенно не важен для дерева и не используется в нем, то ошибка практически не изменится. Таким образом, информативность признака j оценивают как разность

$$Q'_n - Q_n.$$

Далее эти информативности усредняют по всем деревьям случного леса, и чем больше будет среднее значение, тем важнее признак. На практике оказывается, что информативности, вычисленные описанным образом, и информативности, вычисленные как сумма уменьшения критерия информативности, оказываются очень связаны между собой.

3.5. Понижение размерности

3.5.1. Примеры использования методов понижения размерности

Зачем нужно понижать размерность и чем этот подход отличается от отбора признаков? Для ответа на этот вопрос полезно рассмотреть несколько примеров.

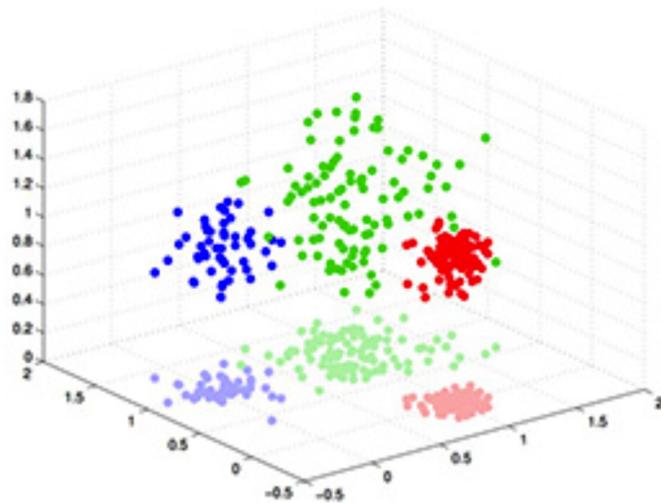


Рис. 3.3: Пример выборки, для которой необходимо решить задачу классификации

На рисунке 3.3 изображена выборка с тремя размерностями. Видно, что если убрать из нее признак, отложенный по оси Z , получится двумерная выборка, в которой синий, зеленый и красный кластеры будут разделены даже линейными методами.

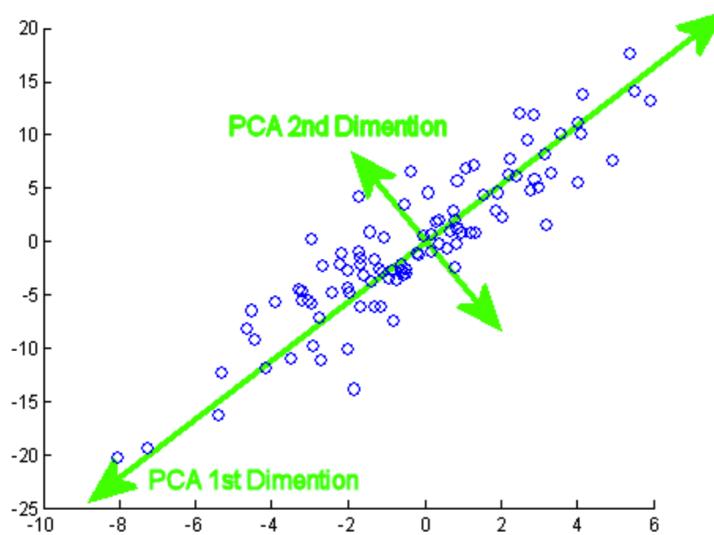


Рис. 3.4: Пример выборки с линейно зависимыми признаками

На рисунке 3.4 показан более сложный случай. В данной выборке оба признака значимые, но при этом они линейно зависимы, и этим можно воспользоваться, чтобы устраниТЬ избыточность в данных. Однако отбора признаков для этого не хватит: необходимо сформировать новый признак на основе двух исходных.

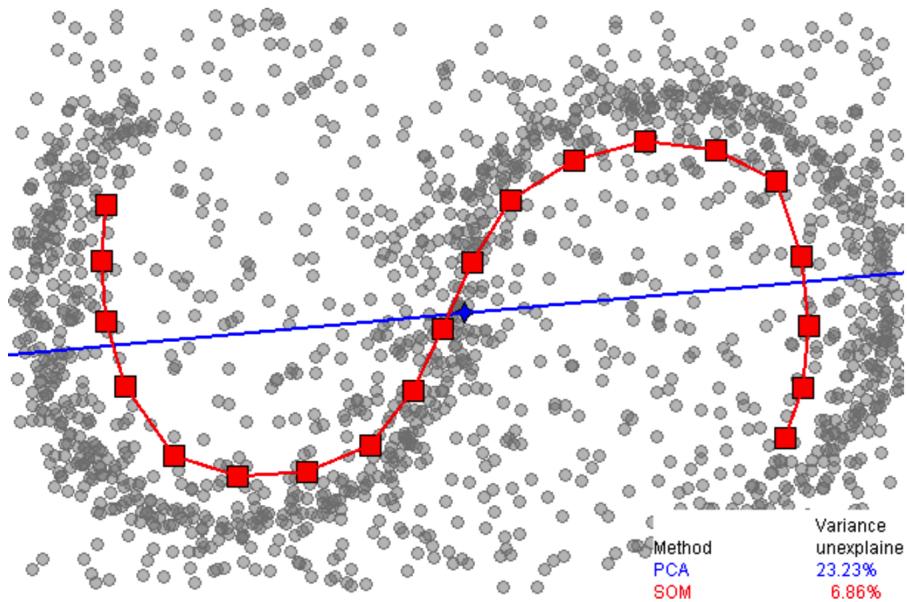


Рис. 3.5: Пример данных, которые необходимо спроектировать на кривую

Бывают и еще более сложные случаи, как, например, на рисунке 3.5. Здесь также можно спроектировать выборку на некоторую кривую, но при этом кривая очень нелинейная, и её, скорее всего, будет сложно найти.

Эти примеры подводят к задаче понижения размерности, которая состоит в формировании новых признаков на основе исходных. При этом количество признаков становится меньше, но они должны сохранять в себе как можно больше информации, присутствующей в исходных признаках.

3.5.2. Метод случайных проекций

Один из основных подходов к понижению размерности — это линейный подход, в котором каждый новый признак представляет собой линейную комбинацию исходных признаков. Необходимо ввести обозначения:

- D — количество исходных признаков;
- x_{ij} — это значение исходного признака j на объекте выборки i ;
- d — число новых признаков;
- z_{ij} — это значение нового признака j на объекте выборки i

Новый признак j на объекте i линейно выражается через исходные признаки на этом же объекте:

$$z_{ij} = \sum_{k=1}^D w_{jk} x_{ik}$$

где w_{jk} — вес, который показывает, какой именно вклад исходный признак k даёт в новый признак j на каждом объекте.

Один из простейших подходов к такому понижению размерности — это метод случайных проекций, в котором веса генерируются случайно, например, из нормального распределения:

$$w_{jk} \sim \mathcal{N}(0, \frac{1}{d}),$$

где d — это количество новых признаков. Данный подход обосновывает лемма Джонсона-Линденаштраусса.

Лемма (Джонсона-Линденаштраусса) *Если в выборке мало объектов, но много признаков, то её можно спроектировать в пространство меньшей размерности так, что расстояния между объектами практически не изменяются (то есть топология в новом признаковом пространстве сохраняется).*

При этом, если необходимо, чтобы расстояния изменились не больше, чем на ϵ , то размерность нового признакового пространства должна удовлетворять условию

$$d > \frac{8 \ln l}{\epsilon^2},$$

где l — количество объектов в выборке.

В лемме утверждается только, что существует такая проекция и что она будет сохранять расстояния между парами признаков. Однако на практике оказывается, что если использовать d , соответствующее описанной выше формуле, то даже метод случайных проекций так понижает размерность выборки, что расстояния сохраняются неплохо, и это понижение оказывается качественным.

Такой подход хорошо работает для текстов, когда пространства оказываются очень большими (обычно тексты кодируются при помощи мешка слов, и количество признаков — это число различных слов в тексте).

3.6. Метод главных компонент: постановка задачи

3.6.1. Метод главных компонент как линейный метод понижения размерности

Метод главных компонент — широко используемый способ понижения размерности. Задачу метода главных компонент можно вывести из линейного подхода к понижению размерности, где каждый новый признак линейно выражается через исходные:

$$z_{ij} = \sum_{k=1}^D w_{jk} x_{ik}$$

Если произвести небольшие преобразования:

$$z_{ij} = \sum_{k=1}^D w_{jk} x_{ik} = \sum_{k=1}^D x_{ik} w_{kj}^T,$$

то это выражение можно записать в матричном виде:

$$Z = XW^T.$$

Чтобы у этого уравнения существовало единственное решение, необходимо ввести ограничение на матрицу весов, она должна быть ортогональной:

$$W^T W = I.$$

Итак, если данное требование будет выполнено, то можно получить формулу для матрицы X :

$$X = ZW$$

Поскольку в матрице Z будет меньше признаков, чем в X , то если ранг матрицы X больше, чем число новых признаков d , то данное равенство точно нельзя будет выполнить строго. В этом случае требуется, чтобы отклонение исходной матрицы признаков X от ZW было как можно меньше, чтобы эти матрицы были как можно сильнее похожи друг на друга. Размер этого отклонения будем вычислять с помощью нормы Фробениуса:

$$\|X - ZW\|^2 \rightarrow \min_{Z, W}$$

Норма Фробениуса матрицы — это сумма квадратов ее значений, аналог векторной нормы l_2 .

Получившаяся задача — это задача матричного разложения. Необходимо представить матрицу X в виде произведения двух матриц Z и W , которые будут иметь меньший ранг. То есть нужно уменьшить ранг матрицы, при этом потеряв как можно меньше информации в ней.

3.6.2. Метод главных компонент как способ проекции данных на гиперплоскость

Существует иной подход к постановке задачи метода главных компонент.

Пусть имеется выборка, изображенная на рисунке 3.6, и её необходимо спроектировать на некоторую прямую. В этом случае прямая будет тем лучше, чем меньше будет ошибка проецирования сумма по всей выборке расстояний от объекта до его проекции на эту прямую. Чем меньше эти расстояния, тем лучше прямая приближает данные, тем меньше будет ошибка и тем больше информации сохраняется. В идеальном случае прямая должна проходить через все объекты выборки, но в рассматриваемой ситуации это невозможно.

В общем случае, когда признаков много, выборка проецируется на гиперплоскость. Из аналитической геометрии известно, что есть два способа задания гиперплоскости. Первый — с помощью вектора нормали, он использовался в линейных методах. Второй — с помощью направляющих векторов. Пусть в исходном пространстве размерности D строится гиперплоскость размерности $D - 1$, тогда если выбрать на этой плоскости D линейно независимых векторов, то они будут однозначно задавать эту гиперплоскость. Если направляющие векторы составить в матрицу W , так что каждый столбец этой матрицы — это один направляющий вектор, то проекция точки x_i на данную гиперплоскость будет вычисляться по формуле $x_i * W$. Тогда для того чтобы уменьшить ошибку проецирования на гиперплоскость, необходимо минимизировать следующее выражение:

$$\sum_{i=1}^{\ell} \|x_i - x_i W\|^2 \rightarrow \min_W.$$

Задача ставится аналогично при проецировании объектов на гиперплоскость любой размерности $d \leq D - 1$.

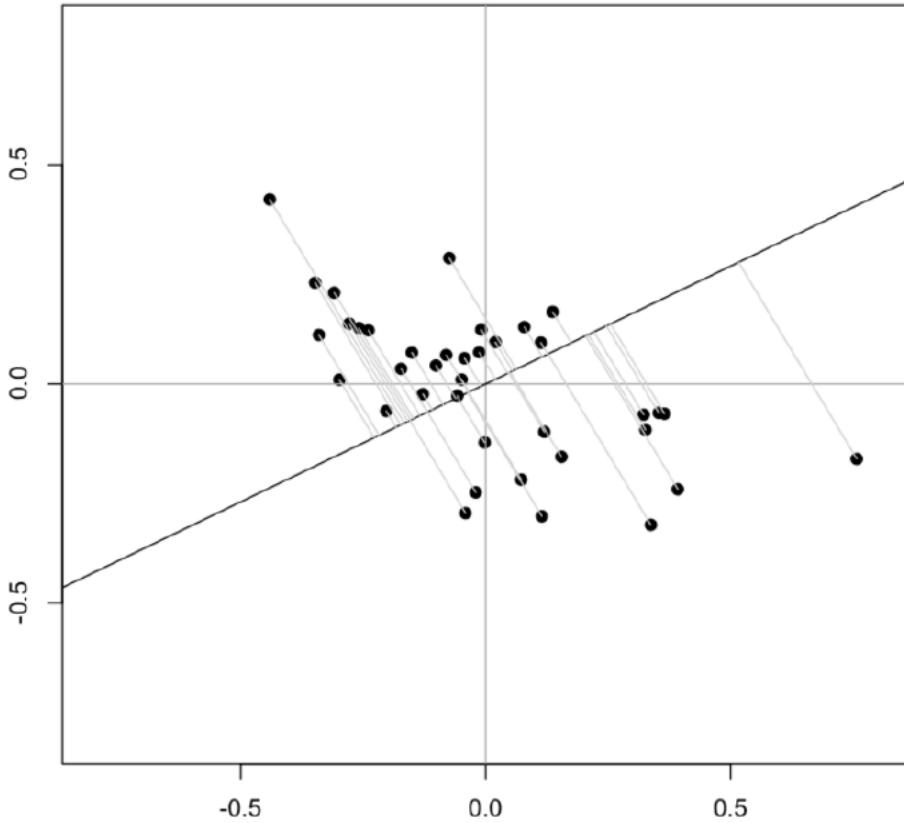


Рис. 3.6: Пример выборки, которую необходимо спроектировать на прямую

3.6.3. Максимизация дисперсии выборки после понижения размерности

Есть и третий взгляд на метод главных компонент. Пусть имеется выборка, показанная на рисунке 3.7, и требуется выбрать прямую, на которую можно будет оптимально спроектировать эту выборку. Синяя прямая лучше подходит для решения данной задачи, поскольку при проецировании на нее сохраняется гораздо больше информации выборки.

Формализовать понятие информации можно с помощью дисперсии: чем больше дисперсия выборки после проецирования на прямую, тем больше сохраняется информации. Для данного случая этот критерий хорошо подходит: дисперсия выборки после проецирования на синюю прямую будет гораздо больше, чем после проецирования на красную прямую.

Формально дисперсию выборки после проецирования можно записать следующим образом:

$$\sum_{j=1}^d w_j X^T X w_j \rightarrow \max_W$$

Чем больше значение этой суммы, тем больше оказывается дисперсия выборки после проецирования на гиперплоскость, которая задается матрицей весов W . Таким образом, это выражение нужно максимизировать, чтобы сохранить как можно больше информации.

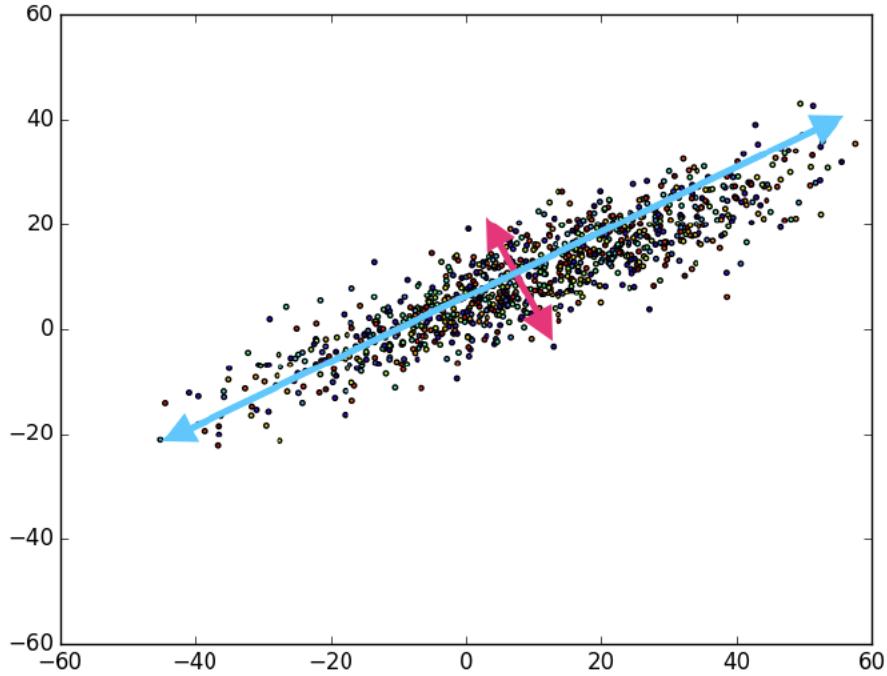


Рис. 3.7: Пример выборки, для которой необходимо решить задачу понижения размерности

3.7. Метод главных компонент: решение

3.7.1. Вывод решения задачи метода главных компонент

Ранее была описана формулировка задачи метода главных компонент, теперь необходимо её решить. Одна из постановок задачи метода главных компонент — это максимизация дисперсии:

$$\begin{cases} \sum_{j=1}^d w_j^T X^T X w_j \rightarrow \max \\ W^T W = I \end{cases}$$

В первой строке записана дисперсия после проецирования, а во второй — ограничение, обеспечивающее наличие единственного решения.

В методе главных компонент есть один нюанс: выражение, через которое записана дисперсия, будет означать именно дисперсию выборки только в том случае, если матрица объекты-признаки центрирована (среднее каждого признака равно нулю). Далее считается, что, выборка центрирована, и среднее из каждого столбца в матрице объекты-признаки уже вычли.

Итак, чтобы разобраться, как устроено решение этой задачи, необходимо сначала рассмотреть простой частный случай: требуется найти ровно одну компоненту, на которую проецируется вся выборка, так, чтобы дисперсия после проецирования была максимальной:

$$\begin{cases} w_1^T X^T X w_1 \rightarrow \max \\ w_1^T w_1 = 1 \end{cases}$$

Для решения подобных задач условной оптимизации необходимо выписать лагранжиан:

$$L(w_1, \lambda) = \frac{T}{1} w_1^T X^T X w_1 - \lambda (\frac{T}{1} w_1^T w_1 - 1).$$

Далее этот лагранжиан необходимо продифференцировать по искомой величине:

$$\frac{\partial L}{\partial w_1} = 2X^T X_{\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}} - 2\lambda w_{\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}} = 0$$

После преобразований получается следующее выражение:

$$X^T X_{\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}} = \lambda w_{\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}}.$$

Из него следует, что w_1 — это собственный вектор матрицы $X^T X$, и число λ является собственным значением, соответствующим этому вектору. После подстановки полученного выражения в функционал задачи, оказывается, что дисперсия выборки после проецирования будет равна собственному значению, соответствующему выбранному собственному вектору:

$$w_{\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}}^T X^T X_{\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}} w_{\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}} = \lambda$$

Таким образом, поскольку требуется максимизировать дисперсию, необходимо выбирать максимальное собственное значение и собственный вектор, который соответствует этому значению.

Итак, в методе главных компонент первая компонента — это собственный вектор матрицы $X^T X$, который соответствует максимальному собственному значению этой матрицы. Стоит обратить внимание, что $X^T X$ — это матрица ковариации, то есть именно та матрица, которая характеризует дисперсию выборки.

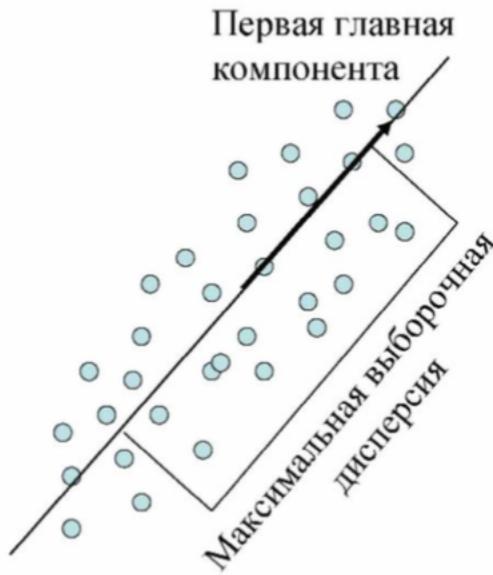


Рис. 3.8: Пример выборки, для которой необходимо решить задачу метода главных компонент

Визуально это выглядит следующим образом: есть облако точек (рисунок 3.8), и необходимо выбрать именно то направление, при проецировании на которое сохраняется как можно больше дисперсии. Это направление и будет задаваться первым собственным вектором матрицы ковариации.

Если продолжить выкладки и дальше искать оптимальные направления, то обнаружится, что, w_2, w_3, \dots, w_d — собственные векторы матрицы $X^T X$, соответствующие наибольшим собственным значениям $\lambda_2, \lambda_3, \dots, \lambda_d$. Отсюда же можно получить, какая доля дисперсии сохранилась после проецирования выборки на главные компоненты:

$$\frac{\sum_{i=1}^d \lambda_i}{\sum_{i=1}^D \lambda_i}$$

3.7.2. Использование сингулярного разложения

При решении задачи метода главных компонент оказывается полезным сингулярное разложение:

$$X = UDV^T,$$

где U и V — это ортогональные матрицы, а D — диагональная матрица. Столбцы матрицы U — это собственные векторы матрицы XX^T , столбцы матрицы V — это собственные векторы матрицы X^TX , а на диагонали матрицы D стоят собственные значения этих матриц, которые, оказывается, совпадают (с точностью до некоторого количества нулевых собственных значений у той матрицы, которая имеет большую размерность). Собственные значения матриц X^TX и XX^T называются сингулярными числами.

Итак, для решения задачи главных компоненты необходимо совершить следующие действия: найти сингулярное разложение матрицы X , сформировать матрицу весов W из собственных векторов (из столбцов матрицы V , соответствующих максимальным сингулярным числам), и после этого произвести преобразование

$$Z = XW,$$

где Z является матрицей объекты-признаки для нового сокращенного признакового описания.

Урок 4

Матричные разложения

4.1. Матричные разложения

4.1.1. Постановка задачи матричного разложения

В задаче матричного разложения требуется приблизить матрицу произведением двух других (рисунок 4.1):

$$X_{l,n} \approx U_{l,k} \cdot V_{k,n}^T,$$

где матрицы U и V задают новое признаковое описание объектов и изначальных признаков.

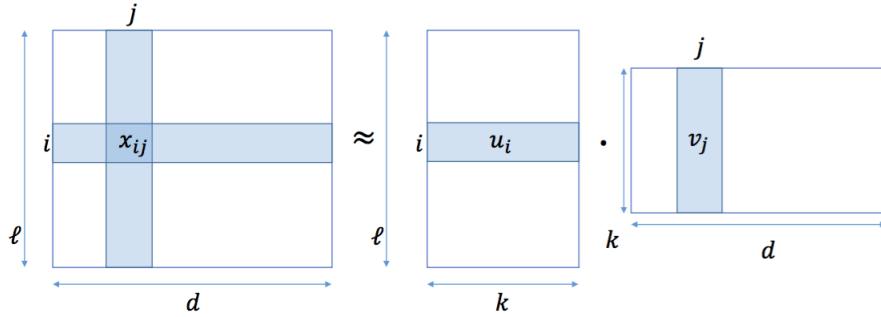


Рис. 4.1: Матрицы X , U , V

Формально задачу можно свести к поиску минимума:

$$\|X - U \cdot V^T\| \rightarrow \min.$$

Если использовать норму Фробениуса:

$$\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2},$$

то задача записывается следующим образом:

$$\sum_{i,j} (x_{ij} - \langle u_i, v_j \rangle)^2 \rightarrow \min,$$

где u_i — новое описание объекта i , а v_j — новое описание признака j .

4.1.2. Сингулярное разложение

Сингулярное разложение — это способ представить некоторую исходную матрицу в виде произведения трех других:

$$X = U \Sigma V^T,$$

где U — ортогональная матрица, Σ — диагональная матрица, V — ортогональная матрица.

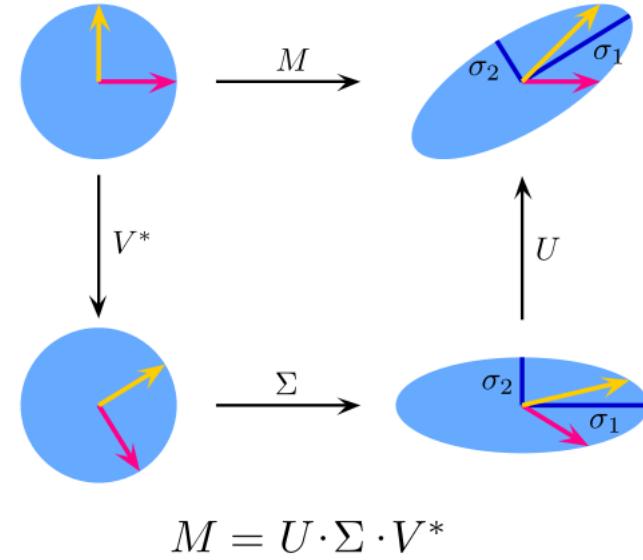


Рис. 4.2: Геометрическая интерпретация сингулярного разложения

Геометрически это можно проиллюстрировать так (рисунок 4.2): если рассматривать матрицу X как задающую некоторое отображение, то оно раскладывается на составляющие: сначала пространство поворачивают, потом растягивают вдоль осей координат, а затем снова поворачивают (говоря более точно, здесь могут быть не только повороты, но и вращения некоторых осей, но в упрощённом виде это выглядит так).

Сингулярное разложение матриц может быть полезно для рассматриваемой задачи. Можно рассмотреть сингулярное разложение матрицы X :

$$X = \tilde{U} \Sigma \tilde{V}^T.$$

Здесь приходится прибегнуть к обозначениям \tilde{U} и \tilde{V} , чтобы не было пересечения с обозначениями для разложения на две матрицы U и V .

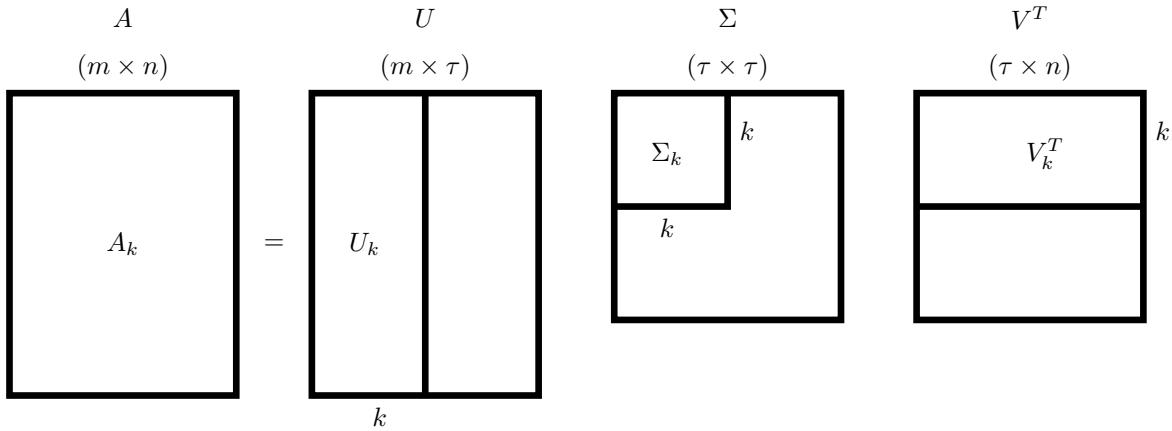


Рис. 4.3: Усечённые матрицы из SVD

Итак, с помощью сингулярного разложения получается представление исходной матрицы в виде произведения трех матриц. Теперь у каждой матрицы можно взять какую-то часть (рисунок 4.3): от матрицы \tilde{U} — первые k столбцов, от матрицы Σ — квадрат размера $k \times k$, от матрицы \tilde{V}^T — первые k строк. Таким образом определяются усеченные матрицы сингулярного разложения:

$$\tilde{U}_k, \quad \Sigma_k, \quad \tilde{V}_k.$$

Оказывается, если в задаче матричного разложения взять в качестве искомых матриц

$$U = \tilde{U}_k \Sigma_k, \quad V = \tilde{V}_k,$$

то это будет решением оптимизационной задачи, которое даст наилучшее приближение исходной матрицы по норме Фробениуса. При этом в качестве искомых матриц можно использовать другие:

$$U = \tilde{U}_k, \quad V = \tilde{V}_k \Sigma_k$$

или

$$U = \tilde{U}_k \sqrt{\Sigma_k}, \quad V = \tilde{V}_k \sqrt{\Sigma_k}.$$

Это указывает на неоднозначность решения задачи матричного разложения.

4.1.3. Использование термина SVD в машинном обучении

Так или иначе, часто оказывается, что производить SVD (т.е. сингулярное) разложение — не очень быстрая операция, и, в принципе, можно мыслить о задаче получения матриц U и V как о задаче оптимизации

$$\sum_{i,j} (x_{ij} - \langle u_i, v_j \rangle)^2 \rightarrow \min,$$

и подбирать u_i и v_j каким-либо оптимизационным методом. В таком случае оказывается, что SVD уже не имеет отношения к задаче, она решается напрямую. Но другой стороны, известно, что задача связана с SVD, и решение этой задачи можно применить к вычислению усечённых матриц из SVD. Поэтому в машинном обучении термин SVD прижился как обозначение решения оптимизационной задачи, указанной выше, и встречая этот термин, нужно помнить, что речь не идёт об обычном сингулярном разложении.

4.1.4. Практическое применение матричного разложения

Матричные разложения часто используются в рекомендательных системах, например, при рекомендации фильмов, когда необходимо заполнить неизвестные значения в матрице произведением векторов, задающих интересы пользователя и параметры фильма. Похожий подход применяется в задаче анализа текстов.

4.1.5. Используемые обозначения

Для полной ясности, необходимо вернуться к обозначениям. Дело в том, что здесь встречается не просто представление матрицы

$$X = UV$$

а в виде

$$X = UV^T.$$

Возникает вопрос: зачем понадобилось транспонировать? Это сделано из желания сделать так, чтобы обе матрицы, U и V представляли собой матрицы размера количество каких-то сущностей (либо объектов, либо исходных признаков) на количество новых признаков k (рисунок 4.4).

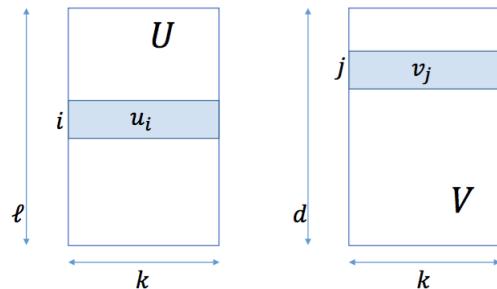


Рис. 4.4: Матрицы U , V

Помимо этого, можно вспомнить запись для x_{ij} в виде скалярного произведения векторов:

$$x_{ij} \approx \langle u_i, v_j \rangle$$

Обычно, когда речь идёт о скалярном произведении, имеется в виду скалярное произведение векторов, которые обычно представляются в виде столбцов. Из этих соображений становится понятно, почему можно встретить запись

$$x_{ij} = u_i^T v_j.$$

Это просто расписанное скалярное произведение, сумма покоординатных произведений векторов. Однако эта же запись может вызвать вопросы, потому что изначально u_i — это строка в матрице U , а не столбец, и v_j — это строка в матрице V (рисунок 4.4). Не стоит путаться из-за этого и нужно понимать, что имеется в виду именно покоординатное произведение u_i и v_j , которые были строчками в матрицах U и V .

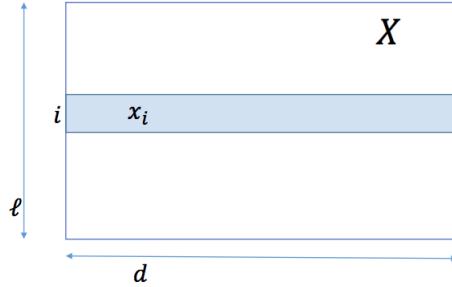


Рис. 4.5: Матрица X

Аналогичная ситуация встречалась, когда шла речь линейных моделях. В этой задаче также присутствовала матрица признаков X (рисунок 4.5), в которой строка i соответствовала объекту с номером i . В то же время встречалась следующая запись скалярного произведения:

$$\langle w, x_i \rangle = w^T x_i$$

Хотя в действительности x_i является строчкой, в этой записи подразумевается столбец.

Кроме того, в матричных разложениях встречается много различных обозначений в зависимости от того, для чего они применяются:

- $X \approx UV^T$ — часто встречаемое в рекомендациях обозначение;
- $X \approx PQ^T$ — это обозначение также часто можно встретить в рекомендациях;
- $X \approx WH$ — встречается в неотрицательных матричных разложениях и в анализе текстов;
- $X \approx \Phi\Theta$ — используется в вероятностных тематических моделях.

4.2. SGD и ALS

4.2.1. Применение стохастического градиентного спуска в задаче матричного разложения

Задача, описанная в предыдущей части, — это минимизация функционала

$$Q = \sum_{i,j} (\langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min_{u_i, v_j} .$$

Этот функционал можно минимизировать методом градиентного спуска. Для этого требуется выписать градиент по u_I :

$$\frac{\partial Q}{\partial u_i} = \sum_{i,j} \frac{\partial}{\partial u_i} (\langle u_i, v_j \rangle - x_{i,j})^2 = \sum_j 2(\langle u_i, v_j \rangle - x_{ij}) \frac{\partial \langle u_i, v_j \rangle}{\partial u_i} = \sum_j 2(\langle u_i, v_j \rangle - x_{ij}) v_j.$$

Из этого можно получить значение ошибки на x_{ij} :

$$\varepsilon_{ij} = (\langle u_i, v_j \rangle - x_{ij})$$

и формулу для обновления u_i на каждой итерации градиентного спуска:

$$u_i^{(t+1)} = u_i^{(t)} - \gamma_t \sum_j \varepsilon_{ij} v_j.$$

Аналогичную формулу можно получить для v_j :

$$v_j^{(t+1)} = v_j^{(t)} - \eta_t \sum_i \varepsilon_{ij} u_i$$

В этих формулах присутствует сумма большого количества слагаемых. Матрицы, на которых используется этот метод, могут быть очень велики, и считать такие суммы на каждом шаге градиента может быть ресурсо-затратно. Именно поэтому для решения задачи неизбежно потребуется метод стохастического градиентного спуска, где на каждом шаге берутся случайные слагаемые, и используются только они:

$$u_i^{(t+1)} = u_i^{(t)} - \gamma_t \varepsilon_{ij} v_j,$$

$$v_j^{(t+1)} = v_j^{(t)} - \eta_t \varepsilon_{ij} u_i.$$

Плюсами стохастического градиентного спуска являются простота реализации и его сходимость. Однако, есть и минусы: алгоритм сходится медленно, и совершенно неясно, как выбирать шаг градиентного спуска (при константном шаге алгоритм сходится очень медленно).

4.2.2. Метод ALS

Метод ALS (alternating least squares) помогает избежать проблем, возникающих при использовании стохастического градиентного спуска. Идея этого метода заключается в следующем: в точке, где функционал будет принимать минимальное значение, его производные по u_i и по v_j должны быть равны нулю. Тогда можно действовать следующим образом: на одной итерации обновляется значение u_i из выражения

$$\frac{\partial Q}{\partial u_i} = 0.$$

На следующей итерации находится новое значение v_j :

$$\frac{\partial Q}{\partial v_i} = 0.$$

Это продолжается до тех пор, пока значения u_i и v_j не стабилизируются.

Можно явно выписать шаги в ALS:

$$\frac{\partial Q}{\partial u_i} = \sum_j 2(\langle u_i, v_j \rangle - x_{ij}) v_j = 0$$

$$\sum_j v_j \langle v_j, u_i \rangle = \sum_j x_{ij} v_j$$

В результате получаются достаточно простые выражения и, сделав несложные преобразования, можно получить задачу в виде решения системы линейных уравнений:

$$\sum_j v_j v_j^T u_i = \sum_j x_{ij} v_j$$

$$\left(\sum_j v_j v_j^T \right) u_i = \sum_j x_{ij} v_j.$$

В итоге необходимо до сходимость повторять решение систем по случайным i, j :

$$\left(\sum_j v_j v_j^T \right) u_i = \sum_j x_{ij} v_j$$

$$\left(\sum_i u_i u_i^T \right) v_j = \sum_i x_{ij} u_i$$

	Пила	Улица Вязов	Ванильное небо	1 + 1
Маша	5	4	1	2
Юля	5	5	2	
Вова			3	5
Коля	3	?	4	5
Петя				4
Ваня		5	3	3

Таблица 4.1: Пример матрицы пользователи/оценки

4.2.3. Использование регуляризации

Важно отметить, что эта задача поставлена не совсем корректно, потому что имеет много решений. В таких случаях для того чтобы получить адекватное решение, обычно используют регуляризацию, то есть добавляют некоторые штрафы за большие значения параметров (u_i, v_j) . Можно применить, например, L2-регуляризатор:

$$Q = \sum_{i,j} (\langle u_i, v_j \rangle - x_{ij})^2 + \alpha \sum_i \|u_i\|^2 + \beta \sum_j \|v_j\|^2 \rightarrow \min_{u_i, v_j}$$

где α и β — небольшие положительные числа (0.001, 0.01, 0.5).

Удивительно, но добавление регуляризаторов очень сильно сказывается на качестве в этой задаче.

4.3. Прогнозирование неизвестных значений в матрице

4.3.1. Постановка задачи в рекомендациях

Одно из применений матричных разложений — это прогнозирование неизвестных значений матрицы, такая задача часто возникает в рекомендательных системах. Итак, в таких системах, как правило, имеется матрица пользователи / оценки:

В этой матрице стоят оценки, которые пользователи поставили тем объектам, которые видели (например, тем фильмам, которые они смотрели). Многие значения в этой матрице неизвестны, потому, как правило, не все пользователи видели все объекты. Можно посмотреть на эту задачу следующим образом: в известных значениях матрицы эти значения приближаются как скалярное произведение некоторого профиля пользователя на профиль объекта:

$$x_{ij} \approx \langle u_i, v_j \rangle.$$

В данном случае оптимизируется функционал, в котором суммы берутся только по известным значениям матрицы:

$$\sum_{i,j: x_{ij} \neq 0} (\langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min.$$

В дальнейшем, взяв скалярное произведение профиля пользователя и профиля фильма, можно получить прогноз оценки, которую пользователь поставил бы фильму, если бы видел его, для тех элементов матрицы, которые нам неизвестны.

4.3.2. Добавление дополнительных параметров

В этой задаче можно добавить учет дополнительных факторов. Например, если шкала подразумевает, что существует некоторая средняя оценка μ , от которой отталкиваются пользователи при оценке фильмов, то правильнее будет приближать оценки следующим образом:

$$x_{ij} \approx \mu + \langle u_i, v_j \rangle,$$

а параметр μ также настраивать, исходя из оптимизационной задачи:

$$\sum_{i,j} (\mu + \langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min$$

Есть и другое соображение: пользователи бывают разными не только в силу своих интересов, но и в силу строгости оценок, которые они ставят фильмам. Одни пользователи ставят оценки выше, другие — ниже. Поэтому можно добавить еще одно слагаемое: b_i^u . Это базовый предиктор пользователя i , который будет отражать предпочтения этого пользователя в среднем по оценкам. Но при этом не стоит считать, что это честное среднее оценок, которые он поставит фильму, потому что присутствует слагаемое μ . Ещё можно добавить в сумму b_j^v , базовый предиктор по фильму j . Это слагаемое в какой-то мере приближает рейтинг фильма самого по себе, без мнения конкретного пользователя о нём.

$$x_{ij} \approx \mu + b_i^u + b_j^v + \langle u_i, v_j \rangle$$

$$\sum_{i,j} (\mu + b_i^u + b_j^v + \langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min$$

И теперь в задаче производится оптимизация взаимодействия интересов пользователя и особенности фильма, без влияния качества фильма, строгости пользователя и без учета средней оценки, которую пользователь ставит фильмам.

Так как были добавлены новые параметры, уместно обсудить вопрос регуляризации. Новых параметров в задаче значительно меньше, чем старых, потому что b_i^u и b_j^v — это просто числа, соответствующие каждому фильму и каждому пользователю. Для них тоже можно добавить l2-регуляризацию:

$$\sum_{i,j} (\mu + b_i^u + b_j^v + \langle u_i, v_j \rangle - x_{ij})^2 + \alpha \sum_i \|u_i\|^2 + \beta \sum_j \|v_j\|^2 + \gamma \sum_i b_i^{u^2} + \delta \sum_j b_j^{v^2} \rightarrow \min .$$

4.4. Проблема отсутствия негативных примеров и implicit методы

4.4.1. Задача рекомендации товаров и её отличия от предыдущей

В отличие от случая рекомендаций фильмов, в случае рекомендации товаров имеется не матрица с оценками пользователей для объектов, а матрица каких-то событий, например, матрица покупок. Она будет устроена следующим образом: в каких-то ячейках матрицы будут нули, в каких-то — единички, реже — числа побольше.

	Вечернее платье	Поднос для писем	iPhone 6s	Шуба D&G
Маша	1		1	
Юля	1	1		1
Вова		1	1	
Коля	1	?	1	
Петя		1	1	
Ваня			1	1

Таблица 4.2: Пример матрицы покупок

Так или иначе, в матрице будут присутствовать только положительные примеры, то есть случаи, когда человек что-то купил: ему понравился товар, пользователь был готов потратить на это деньги, и в итоге потратил деньги. В то же время в матрице не будет примеров товаров, про которые точно известно, что человек никогда это не купит. Невозможно понять: человек не видел этот товар и поэтому не купил, или он его не купил, потому что он ему не нравится.

Пусть в матрице присутствуют только единицы и нули (такое может получиться, если никакой товар не покупался больше одного раза). Кажется, что можно применить все те же методы:

$$x_{ij} = 1 \approx \langle u_i, v_j \rangle$$

$$\sum_{i,j: x_{ij} \neq 0} (\langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min$$

Но если рассматривать задачу матричного разложения как прогнозирование неизвестных значений матрицы, то на такую матрицу, в которой все известные значения — это единицы, легко настроятся константные

профили для пользователей и объектов, которые, не будут нести никакой содержательной информации:

$$u_i = \frac{1}{\sqrt{d}}(1 \dots 1)$$

$$v_j = \frac{1}{\sqrt{d}}(1 \dots 1)$$

Это наводит на мысль, что в постановке задачи что-то не так. И действительно, как уже было сказано ранее, в данных есть только примеры, когда что-то человеку понравилось. Примеров, когда не понравилось, — нет. Поэтому модель в принципе нельзя научить понимать, что что-то кому-то нравится не будет. Конечно, такая модель неприменима на практике.

Таким образом, обратную связь от пользователя можно разделить на два типа:

- Explicit feedback: можно получить как положительные, так и отрицательные примеры (низкие и высокие оценки фильмов, лайки и дизлайки и т.д.)
- Implicit feedback: есть только положительные (лайки, покупки, просмотры) или отрицательные (дизлайки) примеры.

4.4.2. Implicit matrix factorization, implicit ALS

Это наводит на мысль, что нужно адаптировать методы матричных разложений для случая implicit feedback'a. Идея очень простая: можно заполнить неизвестные значения матрицы каким-то числом, например нулем, и настроить матричное разложение на всей матрице. Однако если поступить таким образом, то нулевые и ненулевые значения будут приближаться с одинаковой степенью неопределенности. Это не очень хорошо, потому что в ненулевых значениях есть уверенность, а в нулевых — нет. С этим нужно бороться. Можно решать оптимизационную задачу, но каждому слагаемому в этой задаче присвоить некоторый вес, который зависит от того, известно ли это значение в матрице. В итоге получается задача минимизации некоторого взвешенного расстояния между настоящими ответами и прогнозами:

$$\sum_{i,j} w_{ij} (\langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min$$

При этом для нулевых элементов берутся веса поменьше, чтобы не слишком настраиваться на нули, в которых нет уверенности. Получается, алгоритм склонен ошибаться в тех ячейках матрицы, где стоят нули, в большей степени, нежели в тех, где стоят ненулевые элементы. Это достаточно логично, если интерпретировать нулевые элементы как результат того, что человек на наблюдал данный объект, а не как то что этот объект ему не понравился. Для выбора весов часто используют следующую формулу:

$$w_{ij} = 1 + \alpha |x_{ij}|,$$

где α — это некоторый параметр, который нужно подбирать. Обычно подбирается порядок значения: 10, 100, 1000, и т. д.

Этот метод можно использовать не только для случая implicit feedback'a, потому что в выражении стоит модуль, и можно считать, что значения, близкие к нулю, — это значения, в которых есть уверенность, а чем больше значение по модулю, тем меньше уверенности в нём.

Если ко всему описанному выше добавить оптимизацию с помощью метода ALS, то получится часто используемый на практике алгоритм implicit ALS.

4.5. Вероятностный взгляд на матричные разложения

4.5.1. Матричное разложение по норме Фробениуса и нормальное распределение

В случае построения матричного разложения по норме Фробениуса оптимизационная задача выглядит просто:

$$Q = \sum_{i,j} (\langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min_{u_i, v_j}$$

Она напоминает обобщение метода наименьших квадратов на случай матриц.

Полезно вспомнить, как выглядит нормальное распределение (рисунок 4.6):

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Нормальное распределение очень часто используется для моделирования погрешностей, шумов и других

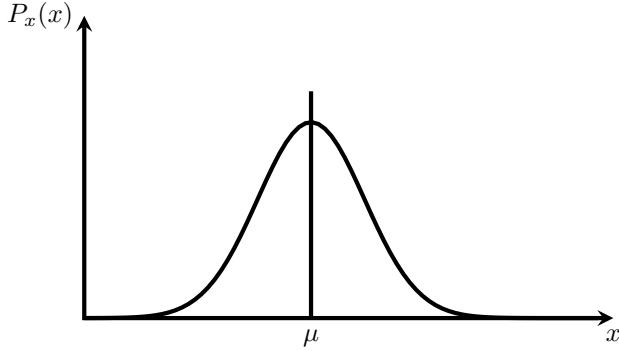


Рис. 4.6: Нормальное распределение

подобных отклонений. Поэтому логично представить, что значения в исходной матрице — это истинные значения (математическое ожидание) плюс нормальный шум:

$$x_{ij} = \langle u_i, v_j \rangle + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

$$x_{ij} \sim \mathcal{N}(\langle u_i, v_j \rangle, \sigma^2)$$

Для такой постановки задачи можно записать принцип максимума правдоподобия:

$$\prod_{i,j} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_{ij} - \langle u_i, v_j \rangle)^2}{2\sigma^2}} \rightarrow \max,$$

и из него получить всё тот же функционал, который напоминает метод наименьших квадратов, который также связан с нормальным распределением:

$$\sum_{i,j} \frac{(x_{ij} - \langle u_i, v_j \rangle)^2}{2\sigma^2} - \frac{1}{2} \ln 2\pi\sigma^2 \rightarrow \min$$

$$\sum_{i,j} (\langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min$$

4.5.2. Распределение Пуассона и неотрицательное матричное разложение

С другой стороны, нормальное распределение оказывается не очень удачным, если значения матрицы — это целые числа, или если в матрице часто встречаются нули. Такая ситуация часто встречается в задачах анализа текстов, когда в качестве признаков выступают частоты слов. Возможно, в данной ситуации лучше подошло бы другое распределение. Если речь идёт о признаках-счётчиках, то для решения задачи лучше всего подходит распределение Пуассона (рисунок 4.7):

$$X \sim Poiss(\lambda)$$

$$P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}, \quad \mathbb{E}X = \lambda$$

	database	SQL	index	regression	likelihood	linear
d1	24	21	9	0	0	3
d2	32	10	5	0	3	0
d3	12	16	5	0	0	0
d4	6	7	2	0	0	0
d5	43	32	20	0	3	0
d6	2	0	0	18	7	16
d7	0	0	1	32	12	0
d8	3	0	0	22	4	2
d9	1	0	0	34	27	25
d10	6	0	0	17	4	23

Таблица 4.3: Таблица частот слов в текстах

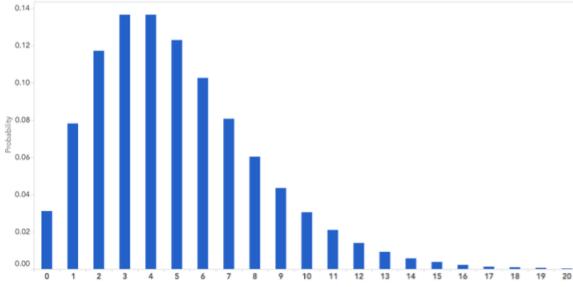


Рис. 4.7: Распределение Пуассона

Распределение Пуассона — это дискретное распределение; значения, которые может принимать случайная величина — целочисленные, поэтому оно действительно подходит для решения данной задачи. В этом случае также можно записать принцип максимизации правдоподобия:

$$P(x_{ij}) = \frac{\langle u_i, v_j \rangle^{x_{ij}}}{x_{ij}!} e^{-\langle u_i, v_j \rangle}$$

$$\prod_{i,j} \frac{\langle u_i, v_j \rangle^{x_{ij}}}{x_{ij}!} e^{-\langle u_i, v_j \rangle} \rightarrow \max$$

Тогда после некоторых преобразований получится выражение, которое нужно оптимизировать, чтобы подобрать профили u_i и v_j , исходя из предположения, что значения в исходной матрице были распределены по Пуассону:

$$\sum_{i,j} \langle u_i, v_j \rangle - x_{ij} \ln \langle u_i, v_j \rangle \ln x_{ij}! \rightarrow \min$$

$$\sum_{i,j} \langle u_i, v_j \rangle - x_{ij} \ln \langle u_i, v_j \rangle \rightarrow \min$$

В этом случае получается разложение исходной матрицы на некоторые матрицы с неотрицательными элементами. Такие разложения называются неотрицательными матричными разложениями, и их гораздо легче интерпретировать, чем разложения с отрицательными элементами. Каждая компонента может означать степень представленности термина, а некоторыми нормировками все эти числа можно привести к вероятностям.

В случае неотрицательного матричного разложения, использующего распределение Пуассона, можно точно так же выписать метод стохастического градиентного спуска и получить простые правила обновления профилей u_i и v_j :

$$Q = \sum_{i,j} \langle u_i, v_j \rangle - x_{ij} \ln \langle u_i, v_j \rangle \rightarrow \min$$

$$\frac{\partial Q}{\partial u_i} = \sum_j v_j - \frac{x_{ij}}{\langle u_i, v_j \rangle} v_j = \sum_j \frac{\langle u_i, v_j \rangle - x_{ij}}{\langle u_i, v_j \rangle} v_j \rightarrow \min$$

$$u_i^{(t+1)} = u_i^{(t)} - \gamma_t \tilde{\varepsilon}_{ij} v_j$$

$$v_j^{(t+1)} = v_j^{(t)} - \eta_t \tilde{\varepsilon}_{ij} u_i$$

Но в то же время использовать распределение Пуассона — это не единственный способ получить неотрицательные матрицы. Можно по-прежнему использовать норму Фробениуса, но при этом добавлять дополнительные ограничения на значения элементов матрицы:

$$Q = \sum_{i,j} (\langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min_{\substack{u_i, v_j: \\ u_{ik} \geq 0 \\ v_{jk} \geq 0}}$$

Подробнее об этом будет рассказано далее.

4.6. Неотрицательные матричные разложения: постановка и решение

4.6.1. Постановка задачи

Прежде чем начать изучать неотрицательные матричные разложения, полезно снова вспомнить постановку задачи факторизации матриц — разложения матрицы на произведение двух матриц меньшего ранга, — но уже в других обозначениях (рисунок 4.8):

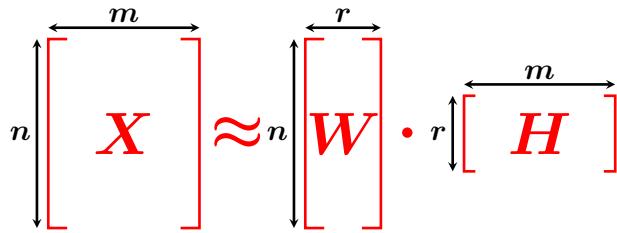


Рис. 4.8: Матричное разложение

$$r < \min(n, m)$$

Задача поиска таких матриц может быть представлена, как оптимизационная:

$$(W^*, H^*) = \underset{W \geq 0, H \geq 0}{\operatorname{argmin}} D(X, WH).$$

Функционал $D(X, WH)$ (функция потерь) показывает насколько точно найденное произведение приближает исходную матрицу X .

Особенность задачи неотрицательных матричных разложений (NMF), заключается в том, что решение W и H ищут в множестве матриц с неотрицательными элементами. Такая постановка часто имеет смысл, если матрицы W и H по итогам разложения нужно интерпретировать. Например, эти матрицы могут представлять собой вероятности появления слов в каких-то текстах, документах, или концентрации веществ в растворах и т.д.

Пусть

$$\hat{X} = WH$$

Интерес представляют функции потерь D , которые распадаются на суммы функции потерь между компонентами матриц X и \hat{X} :

$$D(X, \hat{X}) = \sum_{i=1}^m \sum_{j=1}^n d(x_{ij}, \hat{x}_{ij}).$$

Такие функции потерь называются сепарабельными.

Чаще всего в задаче неотрицательного матричного разложения в качестве функции потерь используют норму Фробениуса:

$$d_F(x, \hat{x}) = (x - \hat{x})^2,$$

$$D_F(X, \hat{X}) = \sum_{i=1}^m \sum_{j=1}^n (x_{ij} - \hat{x}_{ij})^2 \equiv \|X - WH\|_F^2$$

4.6.2. Проблемы задачи поиска неотрицательного матричного разложения

В задаче получения неотрицательных матричных разложений есть несколько проблем. Во-первых, она некорректно поставлена. Если имеется какое-то решение этой задачи W_0, H_0 , то существует большое количество матриц Y , таких, что матрицы

$$W = W_0 Y, \quad H = Y^{-1} H_0$$

также будут решением этой задачи, если существует Y^{-1} и преобразование с помощью Y сохраняет неотрицательность элементов матриц W_0, H_0 . Таким образом, решение никогда не определяется однозначно.

Кроме того, функция потерь D и норма Фробениуса, которая используется чаще всего, не выпукла по совокупности аргументов W и H , поэтому нельзя использовать методы, которые находят глобальный минимум функции потерь, и приходится искать другие методы.

Чаще всего используется блочно-покоординатная минимизация (f и g - некоторые функции):

Вход: $W^0 \geq 0, H^0 \geq 0$

Цикл

$$\begin{aligned} H &\leftarrow f(X, W, H); \\ W &\leftarrow g(X, W, H). \end{aligned}$$

Цикл повторяется до тех пор, пока значения H и W не сойдутся. Поскольку эта задача симметричная, функция f должна быть очень похожа на функцию g , в частности:

$$f(X, W, H) = g^T(X^T, H^T, W^T)$$

4.6.3. Применение градиентных методов

Итак, постановка задачи с использованием нормы Фробениуса выглядит следующим образом:

$$(W^*, H^*) = \underset{W \geq 0, H \geq 0}{\operatorname{argmin}} \|X - WH\|_F^2.$$

Если бы не было ограничения на неотрицательность, решение можно было бы легко получить, используя сингулярное разложение. Но поскольку присутствует ограничение, можно воспользоваться градиентными методами. Базовым методом будет для решения задачи поочередный градиентный спуск:

$$h_{kj} \leftarrow h_{kj} - \nu_{kj} \frac{\partial D_F}{\partial h_{kj}}, w_{ik} \leftarrow w_{ik} - \eta_{ik} \frac{\partial D_F}{\partial w_{ik}}. k = 1, \dots, r; \quad i = 1, \dots, m; \quad j = 1, \dots, n$$

Проблема в том, что градиентный спуск не учитывает ограничение неотрицательности. Существует несколько способов это исправить. Во-первых, можно выбрать шаг градиентного спуска так, что обновления матриц станут мультипликативными, и тогда знак будет автоматически сохраняться:

$$\frac{\partial D_F}{\partial h_{kj}} = \sum_{i=1}^m w_{ik} \hat{x}_{ij} - \sum_{i=1}^m w_{ik} x_{ij}, \quad \nu_{kj} = \frac{h_{kj}}{\sum_{i=1}^m w_{ik} \hat{x}_{ij}}, \quad h_{kj} \leftarrow h_{kj} - \frac{h_{kj}}{\sum_{i=1}^m w_{ik} \hat{x}_{ij}} \left(\sum_{i=1}^m w_{ik} \hat{x}_{ij} - \sum_{i=1}^m w_{ik} x_{ij} \right) = h_{kj} \frac{\sum_{i=1}^m w_{ik} x_{ij}}{\sum_{i=1}^m w_{ik} \hat{x}_{ij}}.$$

В итоге старый элемент матрицы каждый раз умножается на положительное число. Таким образом, если матрица H инициализирована удачно, без отрицательных компонент, то в ходе итерационного процесса они не могут появиться. В матричном виде такие мультипликативные обновления можно записать следующим образом:

$$H \leftarrow H \otimes (W^T X) \oslash (W^T \hat{X}),$$

где \otimes — поэлементное умножение матриц, \oslash — поэлементное деление. Можно показать, что в алгоритме с мультиплекативными обновлениями функция потерь D на каждом шаге монотонно не возрастает.

Методы, использующие мультиплекативные обновления, очень популярны. Во-первых, они просты в реализации. Во-вторых, они хорошо масштабируются и легко приспособляются к работе с разреженными матрицами. Кроме того, исторически эти методы были предложены в самой первой статье по неотрицательным матричным разложениям. Однако скорость сходимости таких методов не очень большая. Впрочем, ее можно увеличить, если обновлять матрицы W и H не по одному разу в цикле, а по несколько раз подряд.

4.6.4. Метод попеременных наименьших квадратов

Задача неотрицательного матричного разложения с нормой Фробениуса может решаться еще несколькими эффективными способами. Если фиксировать одну из двух матриц внутри нормы Фробениуса, получается задача минимизации наименьших квадратов, которую можно решать аналитически. Для этого можно использовать метода попеременных наименьших квадратов (ALS):

$$H \leftarrow \max_H \left(\operatorname{argmin}_H \|X - WH\|_F^2, 0 \right) = \max \left((W^T W)^{-1} W^T X, 0 \right).$$

На каждом шаге находится точное решение задачи наименьших квадратов по одной из компонент, но, к сожалению, это решение может давать и отрицательные значения в матрицах. Их можно заменять нулями, то есть проецировать матрицу на неотрицательную область. Этот метод быстрый и грубый: итерационный процесс может не сходиться при таких обновлениях. Функция потерь не монотонна, то есть она может увеличиваться, что является результатом проецирования. Метод ALS можно использовать для инициализации более сложных методов.

4.6.5. Метод попеременных неотрицательных наименьших квадратов

На другом полюсе методов неотрицательного матричного разложения с нормой Фробениуса находится метод попеременных неотрицательных наименьших квадратов (ANLS). Он действует сначала точно так же, как метод попеременных наименьших квадратов, но минимум функции потерь ищется только в неотрицательной области:

$$H \leftarrow \operatorname{argmin}_{H \geq 0} \|X - WH\|_F^2.$$

Решение такой задачи найти сложнее, его нельзя записать аналитически, поэтому этот метод медленнее, но он точнее. Каждая итерация метода требует существенных вычислительных затрат, поэтому им можно пользоваться для уточнения решения, которое найдено более простыми методами.

4.6.6. Метод иерархических попеременных наименьших квадратов

Наконец, один из самых эффективных и популярных методов неотрицательного матричного разложения с нормой Фробениуса — это метод иерархических попеременных наименьших квадратов (HALS). Идея заключается в том, чтобы найти аналитически минимум нормы Фробениуса не только по матрице W , но и по отдельному ее столбцу, или по строке матрицы H , а затем этот аналитический минимум проецировать на неотрицательную область:

$$h_k \leftarrow \operatorname{argmin}_{h_k \geq 0} \|X - WH\|_F^2 = \max \left(0, \frac{w_k^T X - \sum_{l \neq k} w_k^T w_l h_l}{w_k^T w_k} \right).$$

Проекция в этом случае оказывает менее разрушительное действие, чем в методе попеременных наименьших квадратов, поскольку одновременно обновляются только небольшие куски матриц. Такой метод вычислительно эффективен и сходится быстрее, чем метод мультиплекативных обновлений, однако он более чувствителен к начальному приближению.

4.7. Неотрицательные матричные разложения: функционалы и инициализация

4.7.1. Функции потерь

Существует огромное количество функций потерь, единственное накладываемое ограничение в данной задаче — это сепарабельность, то есть функция потерь должна распадаться на сумму поэлементных расстояний между матрицами X и \hat{X}

$$(W^*, H^*) = \underset{W \geq 0, H \geq 0}{\operatorname{argmin}} D(X, WH).$$

Кроме того, это расстояние должно быть всегда неотрицательным:

$$D(X, \hat{X}) = \sum_{i=1}^m \sum_{j=1}^n d(x_{ij}, \hat{x}_{ij}), d(x, \hat{x}) \geq 0, d(x, \hat{x}) = 0 \Leftrightarrow x = \hat{x}$$

Этим условиям удовлетворяют функции потерь, указанные в таблице ниже.

Название	$d(x, \hat{x})$
норма l_1	$d_1(x, \hat{x}) = x - \hat{x} $
норма Фробениуса	$d_F(x, \hat{x}) = (x - \hat{x})^2$
обобщённая дивергенция Кульбака-Лейблера	$d_{KL}(x, \hat{x}) = x \ln \frac{x}{\hat{x}} - x + \hat{x}$
дивергенция Итакура-Сайто	$d_{IS}(x, \hat{x}) = \ln \frac{\hat{x}}{x} + \frac{x}{\hat{x}} - 1$
расстояние Хеллингера	$d_H(x, \hat{x}) = (\sqrt{x} - \sqrt{\hat{x}})^2$

Таблица 4.4: Функции потерь, подходящие для решения задачи неотрицательного матричного разложения

4.7.2. Связь различных функций потерь с правдоподобием

В различных прикладных областях используются разные функции потерь. Во многих биологических приложениях используется норма Фробениуса, в анализе аудиозаписи — дивергенция Итакура-Сайто, в задачах моделирования текстов чаще всего используется обобщённая дивергенция Кульбака-Лейблера. Почему какие то функции являются оптимальными для своих классов задач? Дело в том, что функции потерь часто представляют собой замаскированные правдоподобия. То есть существуют такие плотности $p(X|\hat{X})$, что

$$D(X, \hat{X}) \propto -\ln p(X|\hat{X}).$$

Таким образом, при минимизация функции потерь происходит максимизация логарифма правдоподобия.

Функция потерь	Порождающая модель
Фробениуса	аддитивная гауссовская
Кульбака-Лейблера	пуассоновская
Итакура-Сайто	мультипликативная гамма

Таблица 4.5: Порождающие модели для функций потерь

4.7.3. Использование дивергенции Кульбака-Лейблера в качестве функционала потерь

Дивергенция Кульбака-Лейблера находится на втором месте по популярности среди всех функционалов потерь для решения задач неотрицательного матричного разложения. Для неё можно точно так же как и для

нормы Фробениуса записать блочно-покоординатный спуск с мультиплекативными обновлениями:

$$h_{kj} \leftarrow h_{kj} \frac{\sum_{i=1}^m w_{ik} \frac{x_{ij}}{\hat{x}_{ij}}}{\sum_{i=1}^m w_{ik}}.$$

И так же можно показать, что функция потерь будет монотонно не возрастать при таких обновлениях. Ещё один алгоритм, который решает самую задачу минимизации дивергенции Кульбака-Лейблера, но немного иначе, — это метод PLSA, о нём будет рассказано позже.

4.7.4. Инициализация искомых матриц

Инициализация в задачах неотрицательного матричного разложения играет очень большую роль, поскольку у минимизируемого функционала очень много локальных минимумов, и сходимость метода тоже локальная. В зависимости от того, насколько точно начальное приближение, получаются разные по качеству решения. Какими же могут быть начальные приближения?

- Случайная инициализация: матрицы W и H можно заполнить случайными неотрицательными числами.
- Кластеризация: можно разделить столбцы матрицы X на r кластеров (например, методом K-means), затем инициализировать столбцы матрицы W центроидами этих кластеров, а строки матрицы H инициализировать в соответствии с матрицей смежности.
- SVD: получить сингулярное разложение матрицы X , отобрать r собственных троек, относящихся к наибольшим собственным числам, и затем из соответствующих собственных векторов сконструировать неотрицательные матрицы W и H .
- ALS: на любом из предыдущих начальных приближений можно произвести несколько итераций метода попарных наименьших квадратов, это улучшает качество приближения.
- Мультистарт: генерируются $10 - 20$ случайных матриц W и H . Каждое начальное приближение улучшается методом ALS, затем провести $10 - 20$ итераций основного метода (например, метода с мультиплекативными обновлениями). Из полученных $10 - 20$ пар выбирается пара с наименьшим значением целевого функционала, это и будет начальное приближение.

4.8. Обработка пропусков

4.8.1. Постановка задачи

Пусть имеется матрица $X \in \mathbb{R}^{\ell \times d}$, строки этой матрицы — это пользователи сайта «КиноПоиск», столбцы — это фильмы, а элементы матрицы — это оценки, которые пользователи выставляют фильмам. Большая часть людей смотрела малую долю фильмов, которые есть в базе «КиноПоиска». Поэтому в матрице X будет огромное количество неизвестных значений. Можно ли пропуски в такой матрице как-то заполнить?

Ещё один пример. Пусть матрица X — это матрица объекты-признаки, дальше на этих данных нужно выполнить задачу классификации или регрессии. Как это правильно делать?

Ключевое предположение всех методов работы с пропусками — случайность расположения пропусков в матрице. Это предположение очень сильное и в каждой задаче нужно проверять, действительно ли оно выполняется. Вот пример задачи, в которой оно заведомо не выполняется: если в опросе есть вопрос о годовом доходе респондента, многие могут отказаться на него отвечать. Более того, может оказаться так, что респонденты с большим годовым доходом чаще будут отказываться отвечать на этот вопрос и тогда в итоговой матрице, составленной по результатам опроса, будут неслучайные пропуски. Еще один классический пример взят из практики венгерского статистика Абрахама Вальда, который работал в США во время второй мировой войны и анализировал повреждения самолетов, которые возвращались с бомбардировок. На основании этого анализа принималось решение о том, какие части самолетов нужно укрепить. Нельзя укрепить весь самолет целиком, потому что от этого он будет плохо летать, поэтому необходимо выбрать, какие части самолетов важнее всего. Для того чтобы это понять, собиралась статистика о количестве пулевых отверстий в разных частях самолетов. Это задача, в которой пропуски не просто неслучайны, а несут в себе самую важную информацию о признаке, потому что именно те самолеты, которые не возвращаются с боевых действий,

получили наиболее критические повреждения. Поэтому Вальд принял решение о том, что укреплять нужно те части самолетов, в которых было меньше всего повреждений на вернувшихся самолётах. Эти примеры показывают, что нужно всегда следить за выполнением предположения о случайности пропусков.

4.8.2. Методы обработки пропусков

Что же можно сделать с матрицей X в случае, когда данное предположение выполняется?. Самый простой способ — выбрасывать те объекты, на которых значение хотя бы одного из признаков пропущено. Перед этим стоит избавиться от признаков, у которых очень много пропусков, потому что иначе можно остаться совсем без выборки. Этот способ очень прост, но он не очень хорош, потому что исключаются объекты, для которых какие-то признаки известны, информация теряется.

Другая крайность работы с пропусками — это методы, которые пытаются заполнить пропуски. Это можно делать большим количеством разных способов. Например, для каждой строки x_{i1} , в которой есть хотя бы один пропуск, находят самую похожую (по функции потерь) на нее строку x_{i2} , и заменяют пропущенное значение в первой строке на аналогичное во второй.

Другой способ — это заполнение пропусков средними или медианами по столбцу. Таким образом, не используется никакой информация об объекте, а только среднее значение признака во всей выборке.

Эти две крайности сочетает друг с другом ЕМ-алгоритм. ЕМ-алгоритм предполагает, что полные данные каким-то образом распределены (например, имеют совместное многомерное нормальное распределение). По имеющимся данным оценивается среднее и ковариационная матрица признаков у этого распределения. Затем пропуски заполняются наиболее вероятными значениями в соответствии с полученной оценкой распределения. Этот процесс повторяется несколько раз: после заполнения пропусков, параметры распределения переоцениваются, после получения новых параметров пропуски заполняются. И так, пока процесс не сойдет ся.

Еще один способ заполнения пропусков — это матричное разложение. Можно представить имеющуюся большую разреженную матрицу в виде произведения двух плотных матриц меньшего ранга, а затем пропуски в исходной матрице представить, как значение матрицы $\hat{X} = WH$.

Все эти методы вызывают подозрение, поскольку непонятно, откуда в них берётся информация. В некоторых задачах (например, линейная регрессия или метод главных компонент) можно обойтись вовсе без заполнения пропусков. В таких задачах часто матрица объекты-признаки X используется только для подсчета величин вида $\frac{1}{\ell} X^T X$ и $\frac{1}{\ell} X^T y$. Значение таких матриц можно вычислять только по полным парам, если оба необходимых элемента не пропущены:

$$\frac{1}{\ell} (X^T X)_{jk} = \frac{1}{\ell} \sum_{i=1}^{\ell} x_{ij} x_{ik} \approx \frac{1}{\ell_{jk}} \sum_{i=1}^{\ell} x_{ij} x_{ik} [x_{ij} \neq NA, x_{ik} \neq NA],$$

где ℓ_{jk} — число полных пар. При использовании таких методов информация никуда не исчезает, и не появляется ниоткуда. Кажется, что когда этот метод применим, он оптимален.

4.8.3. Важные детали

Стоит обсудить еще несколько деталей в задаче заполнения пропусков. Если пропущены значения некоторой категориальной переменной, то удобно закодировать их в виде новой категории. Этот метод работает, в том числе и для ситуации, когда пропуски неслучайны. При работе с деревьями или лесами, если пропущены значения в каких-то непрерывных признаках, их можно заполнить значением, которое сильно отличается от всех типичных значений признака (например, очень большим отрицательным значением). Если так поступить, эти значения будут попадать в отдельный лист, и будут обрабатываться отдельно.

Заполнение пропусков — это творческая задача, поэтому к ней нужно подходить вдумчиво в каждой конкретной задаче анализа данных. Если используется какой-либо метод обучения с учителем, и в матрице объекты-признаки есть пропуски, нужно следить за тем, какой метод обработки пропусков используется по умолчанию в используемом методе. Например, в большей части методов, реализующих линейную регрессию, по умолчанию используется отбрасывание объектов, на которых есть пропуски, — не очень хороший метод. В методе xgboost по умолчанию используется достаточно сложный метод обработки пропусков и, этот алгоритм можно использовать менее вдумчиво.

Урок 5

Поиск аномалий

5.1. Задача обнаружения аномалий

5.1.1. Связь обнаружения аномалий с другими задачами машинного обучения

В курсе уже встречалось несколько примеров задач поиска структуры данных (обучения без учителя). Например, задача кластеризации, в которой требуется найти такие группы объектов, что объекты внутри внутри каждой из них были похожи друг на друга (рисунок 5.1).

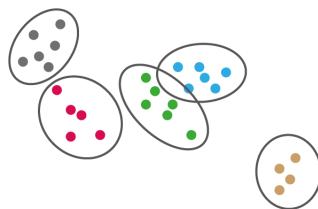


Рис. 5.1: Пример задачи кластеризации

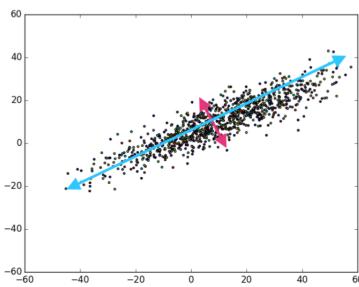


Рис. 5.2: Пример задачи понижения размерности

Другой пример задачи поиска структуры в данных — это задача понижения размерности, когда имеется некоторая выборка, и требуется её спроецировать в пространство меньшей размерности так, чтобы сохранить как можно больше информации (рисунок 5.2), то есть найти какие-то направления, которые наиболее информативны для данной выборки.

Задача поиска или обнаружения аномалии немного отличается от вышеуказанных. В ней нужно найти в выборке объекты, которые не похожи на большинство объектов, которые выделяются, являются аномальными (рисунок 5.3). При этом примеров аномалий либо нет вообще, либо их очень мало. Именно поэтому эта задача относится к обучению без учителя: данные не размечены. Итак, необходимо научиться понимать, похож ли новый объект на остальные, те, которые были известны до этого.

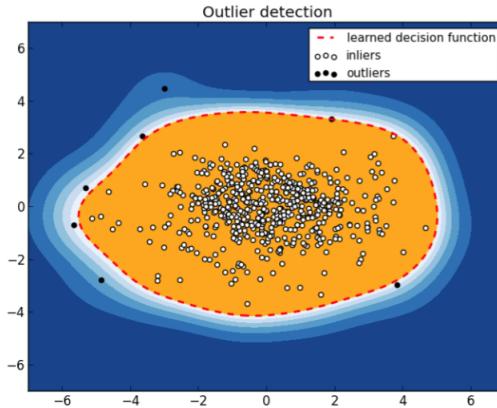


Рис. 5.3: Пример задачи поиска аномалий

5.1.2. Применение обнаружения аномалий на практике

Полезно рассмотреть несколько примеров применения задачи обнаружения аномалий. **Первый пример** касается изучения клиентов банка. Пусть каждый объект — это клиент банка в определенный момент времени. Его можно описать характеристиками транзакций в данный момент, поведением в интернет-банке и т.д. Вопрос, на который требуется ответить: не является ли его поведение необычным? Если оно выбивается из среднего по всем клиентам, то это повод заподозрить, что клиент делает что-то не так. Возможно, это мошенник, который украл карту и пытается вывести с нее деньги — будет повод заблокировать карту и позвонить клиенту банка.

Другой пример задачи обнаружения аномалий — это мониторинг сложной компьютерной системы, которая состоит из большого количества взаимосвязанных машин. Можно отслеживать много показателей: загрузку процессоров, использование памяти на каждой машине, нагрузку на сеть и т.д. Вопрос, на которой требуется ответить: отличается ли текущее состояние системы от характеристик тех состояний, про которые известно, что они нормальные. Если отличается, и система ведет себя как-то иначе, то это повод задуматься, не случилась ли какая-то поломка, нужно ли продиагностировать систему и что-то починить в ней.

Наконец, **третий пример**. Пусть имеется модель, которая по отзыву о банке определяет его тональность: позитивную или негативную. Процедура следующая: клиент заходит на сайт банка, в специальную форму вводит некоторый отзыв, дальше этот отзыв приходит на вход модели, которая определяет, позитивный он или негативный. Если он негативный, то нужно сообщить об этом сотрудникам банка, чтобы они решили возникшую проблему. Но помимо этого хотелось бы понимать, когда приходит новый отзыв, применима ли к нему имеющаяся модель машинного обучения, возможно ли его классифицировать этой же моделью. Дело в том, что распределение признаков этого объекта могло измениться. Например, банк мог поменять название продуктов, и поэтому теперь слова, встречающиеся в отзыве, совершенно другие, — модель к ним не готова. Или банк мог изменить ограничение на длину отзыва. До клиенты писали довольно длинные тексты, а теперь длину ограничили, из-за этого отзыв должен быть очень коротким. В этом случае объекты станут совершенно другими: клиенты будут стараться максимально скжато объяснить свою проблему. Из-за этого модель может стать непригодной для решения задачи. Если стало известно, что объекты стали другими, аномальными, по сравнению с теми, на которых обучалась модель, то это повод её обучить на новых размеченных данных.

Далее будут описаны два подхода к обнаружению аномалий. Первый основан на восстановлении плотности распределения, второй подход использует методы классификации.

5.2. Параметрическое восстановление плотности

5.2.1. Вероятностный подход к обнаружению аномалий

В вероятностном подходе к обнаружению аномалий считается, что аномалия — это объект, который был получен из распределения, отличного от того, с помощью которого сгенерирована обучающая выборка. Возникает вопрос: как найти распределение, из которого была получена выборка? Если найти это распределение, то можно оценить вероятность принадлежности нового объекта этому распределению. Если вероятность получить новый объект из этого распределения очень мала, то это, скорее всего, аномалия.

Существует три основных подхода к восстановлению вероятностных плотностей:

- параметрический подход,
- непараметрический подход,
- восстановление смесей.

5.2.2. Параметрический подход

Итак, существует некоторое вероятностное распределение $p(x)$ на всех объектах, которые можно получить. Считается, что это распределение является параметрическим:

$$p(x) = \phi(x|\theta),$$

где θ задаёт параметры распределения.

Один из самых известных примеров параметрического семейства распределений — это нормальное распределение (рисунок 5.4):

$$\phi(x|\theta) = \mathcal{N}(\mu, \Sigma)$$

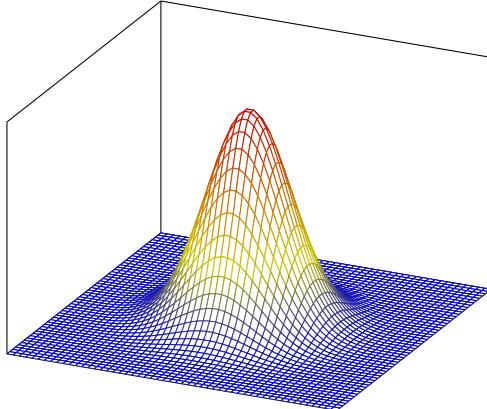


Рис. 5.4: Нормальное распределение

Нормальное распределение задаётся параметрами μ, Σ (центр и ковариационная матрица). По форме оно похоже на шляпу, при этом параметр μ определяет, где находится центр этой шляпы, а параметр Σ — то, насколько она сосредоточена вокруг центра или размазана по всему пространству. Таким образом, если пытаться моделировать выборку с помощью нормального распределения, то необходимо определить параметры μ и Σ . Как это делать?

5.2.3. Метод максимального правдоподобия

Логично искать параметр распределения так, чтобы максимизировать вероятность объектов обучающей выборки быть порождёнными этим распределением. В этом случае объекты, которые не похожи на эту выборку, будут получать низкие вероятности. Именно так работает метод максимального правдоподобия, который старается подобрать такое распределение из параметрического семейства, что с его точки зрения объекты

обучающей выборки будут как можно более вероятны. Работать с самим правдоподобием неудобно, поскольку это — произведение значений плотности во всех точках обучающей выборки. Вместо можно взять его логарифм и пытаться максимизировать полученную сумму:

$$\sum_{i=1}^{\ell} \log \phi(x_i | \theta) \rightarrow \max_{\theta}$$

Для некоторых распределений эту задачу можно решить аналитически, если посчитать частные производные и приравнять их к 0. Например, для нормального распределения такие решения существуют:

$$\mu = \frac{1}{\ell} \sum_{i=1}^{\ell} x_i$$

$$\Sigma = \frac{1}{\ell} \sum_{i=1}^{\ell} (x_i - \mu)(x_i - \mu)^T$$

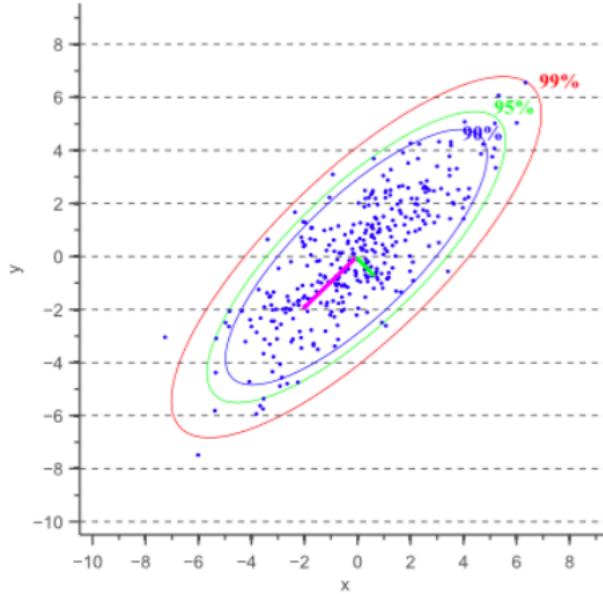


Рис. 5.5: Выборка, порождённая нормальным распределением

Пусть выборка, изображённая на рисунке 5.5, порождена нормальным распределением. Если найти параметры этого распределения методом максимального правдоподобия, то оно будет выглядеть так, как задают линии уровня на рисунке: внутри синей линии уровня находится 90 % всей вероятности, внутри зеленой — 95 %, внутри красной — 99 %. Таким образом, вероятность получить объект вне красного эллипса очень мала. При этом в выборке присутствуют две точки, которые находятся вне красного эллипса. Поскольку основная выборка очень хорошо описывается этим нормальным распределением, а те две синие точки им описываются плохо, то можно предположить, что они пришли из другого распределения, что это аномалия.

Итак, если уже найдено некоторое распределение $p(x)$, и приходит новый объект x , необходимо вычислить вероятность порождения этого объекта данным распределением и сравнить её с некоторым порогом t . Если вероятность меньше этого порога, объект объявляется аномалией.

Возникает вопрос: как выбирать порог t ? На этот счёт не существует однозначных рекомендаций. Можно, например, выбирать его из априорных соображений (как было в примере, объявлять аномалиями все объекты, находящиеся вне линии уровня 99 %). Или, если имеются объекты, про которые точно известно, что это — аномалии, то можно подобрать порог t так, чтобы эти объекты были объявлены аномальными, а все остальные — сгенерированными из распределения $p(x)$.

5.2.4. Модель смеси распределений, ЕМ-алгоритм

В некоторых случаях параметрического подхода оказывается недостаточно. Например, на рисунке 5.6 изображена выборка, которая сгенерирована из двух нормальных распределений с одинаковыми матрицами ковариаций, но разными центрами, таким образом, получается два облака точек. Описать эту выборку одним нормальным распределением будет невозможно, зато для этого отлично подходит модель смеси распределений.

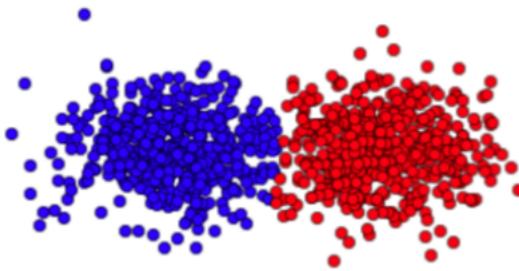


Рис. 5.6: Выборка, порождённая двумя нормальными распределениями

Смесью называется такое распределение $p(x)$, которое представляется в виде взвешенной суммы других распределений:

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

Распределения $p_j(x)$ называются компонентами смеси, и, как правило, они являются параметрическими распределениями. Собственно, каждая компонента p_j является членом параметрического семейства $\phi(x)$ со своим параметром θ_j .

Смеси распределений уже упоминались в уроке, когда рассказывалось про кластеризацию с помощью ЕМ-алгоритма, который можно использовать и для решения этой задачи. Этот алгоритм состоит из повторения Е-шага и М-шага до тех пор, пока не будет достигнута сходимость. На Е-шаге вычисляются апостериорные вероятности того, что объект i принадлежит компоненте j смеси:

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

На М-шаге апостериорные вероятности используются, чтобы обновить оценки на параметров θ . Эти оценки вычисляются путем решения задачи максимизации взвешенного правдоподобия:

$$\begin{aligned} w_j &= \frac{1}{N} \sum_{j=1}^N g_{ji} \\ \theta_j &= \operatorname{argmax}_{\theta} \sum_{j=1}^N g_{ji} \ln \phi(\theta, x). \end{aligned}$$

Благодаря этому алгоритму можно определить, какая именно смесь из K распределений порождает выборку.

5.3. Непараметрическое восстановление плотности

5.3.1. Формула Парзена-Розенблатта и его параметры

Непараметрический подход к восстановлению плотности состоит в том, что вид распределения пытаются восстановить, не вводя никаких семейств распределений, используя только сами данные.

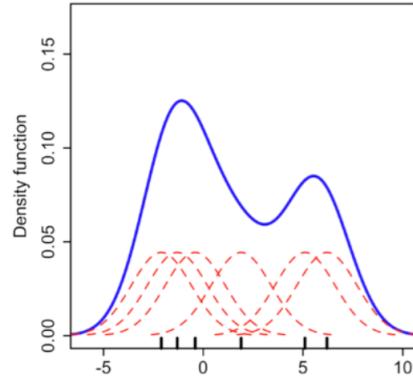


Рис. 5.7: Пример непараметрического восстановления плотности

Пусть имеется одномерная выборка (рисунок 5.7, объекты выборки обозначены засечками по оси абсцисс). В каждой точке обучающей выборки помещают центр небольшой гауссианы (показаны красной линией), таким образом всем точкам на оси присваиваются некоторые вероятности. Далее, в каждой точке эти гауссианы суммируются, и получается итоговое распределение (на рисунке показано синим).

Формально это можно сделать с помощью формулы Парзена-Розенблatta:

$$p_h(x) = \frac{1}{\ell h} \sum_{i=1}^{\ell} K\left(\frac{x - x_i}{h}\right),$$

где $K(r)$ — ядро, параметр метода, характеризующий вероятность того, что точки x и x_i похожи друг на друга. Ядро — это чётная функция, также для него должно выполняться условие

$$\int K(r)dr = 1.$$

Если это требование не выполнено, плотности будут получаться ненормированными. Всё написанное выше верно для одномерных выборок, x и x_i — это вещественные числа.

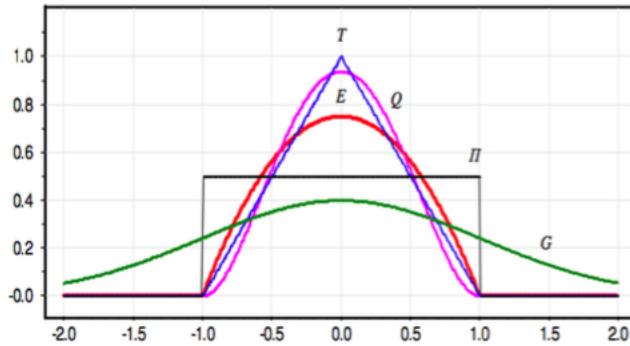


Рис. 5.8: Примеры ядер

Существует много различных ядер. Некоторые из них изображены на рисунке 5.8:

- $E(r) = \frac{3}{4}(1 - r^2)[|r| \leq 1]$ — оптимальное;
- $Q(r) = \frac{15}{16}(1 - r^2)^2[|r| \leq 1]$ — квадратичное;
- $T(r) = (1 - |r|)[|r| \leq 1]$ — треугольное;

- $G(r) = (2\pi)^{-\frac{1}{2}} \exp(-\frac{1}{2}r^2)$ — гауссовское;
- $\Pi(r) = \frac{1}{2}[|r| \leq 1]$ — прямое.

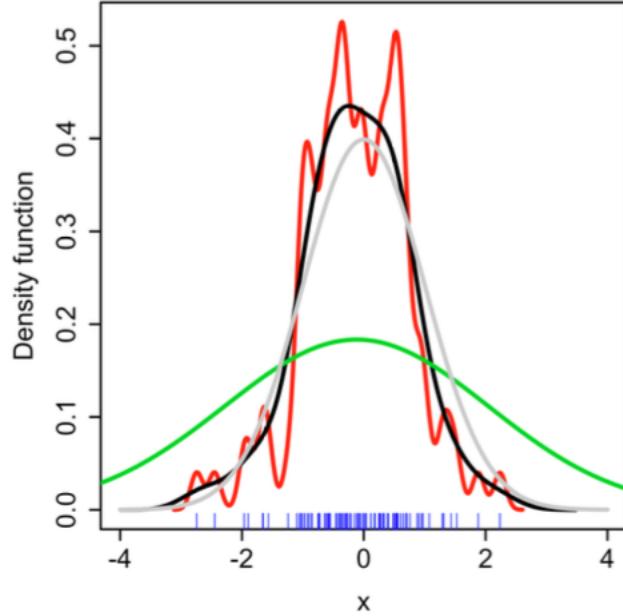


Рис. 5.9: Влияние ширины окна на восстановленную плотность вероятности

Еще один параметр оценки Парзена-Розенблатта — это ширина окна h . Влияние этого параметра на результирующую плотность вероятности можно рассмотреть на примере выборки, изображённой на рисунке 5.9. Элементы выборки на изображены синими засечками на оси абсцисс. Красной, черной и зеленой кривыми показаны непараметрические оценки с гауссовским ядром для разных значений ширины окна. Красная кривая соответствует очень маленькой ширине окна, и результирующая плотность очень чувствительна ко всем точкам. Она получается не очень гладкой и, скорее всего, переобученной. Черная кривая соответствует более высокому значению ширины окна. Эта плотность очень неплохо восстанавливает нормальное распределение, из которого были сгенерированы данные (на рисунке показана серым цветом). Зеленая кривая соответствует оценке плотности с очень большой шириной окна, результирующая плотность характеризуется большой дисперсией. Даже точки на прямой, которые очень далеки от объектов обучающей выборки, получают высокое значение плотности из-за того, что окно широкое.

5.3.2. Многомерный случай, возникающие проблемы

Непараметрическое оценивание плотности хорошо обобщается на многомерный случай, при этом необходимо разность между точками x и x_i заменить метрикой и ввести нормировочную константу $V(h)$, чтобы плотность была нормирована:

$$p_h(x) = \frac{1}{\ell V(h)} \sum_{i=1}^{\ell} K\left(\frac{\rho(x, x_i)}{h}\right)$$

$$V(h) = \int K\left(\frac{\rho(x, x_i)}{h}\right) dx$$

У многомерного непараметрического подхода к восстановлению плотности есть одна большая проблема. Для примера можно рассмотреть выборку, изображённую на рисунке 5.10. Если спроектировать эту выборку на одну ось, то для одномерного случая имеется достаточно объектов, чтобы восстановить плотность вероятности. Однако в двумерном пространстве на один элемент площади приходится гораздо меньше объектов,

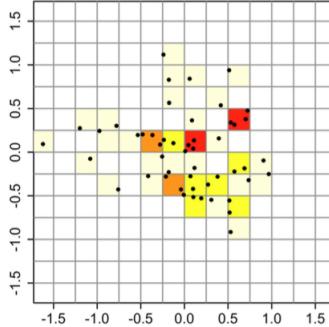


Рис. 5.10: Пример многомерное непараметрического восстановления плотности

чем на соответствующий ему отрезок в одномерном пространстве. Поэтому оценка плотности становится более шумной и менее репрезентативной. Число объектов, необходимых для восстановления плотности, растёт экспоненциально с увеличением размерности пространства, и на практике этот подход применим в пространствах не очень высокой размерности.

5.3.3. Обнаружение аномалий с использованием непараметрического подхода

Обнаружение аномалий с помощью этого подхода происходит так же, как и с использованием параметрического восстановления плотности: для каждого нового объекта вычисляется плотность вероятности $p(x)$, если $p(x) < t$, где t — заданный порог, то объект считается аномалией.

5.4. Одноклассовый SVM

5.4.1. Связь обнаружения аномалий с задачей классификации

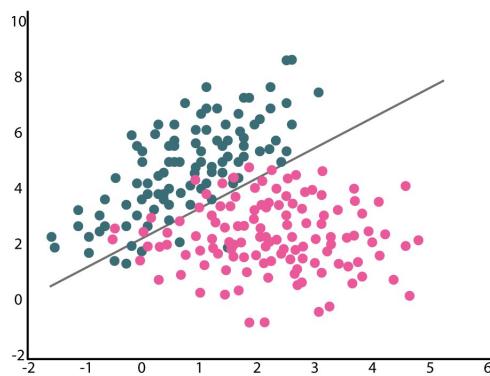


Рис. 5.11: Задача классификации

Задачу обнаружения аномалий можно связать с задачей классификацией. В задаче бинарной классификации (рисунок 5.11) 2 класса необходимо разделить прямой (в пространствах высокой размерности — гиперплоскостью).

В задаче обнаружения аномалий тоже имеется выборка, но требуется построить некоторую кривую, которая отделит выборку от всего остального (рисунок 5.12). Все, что находится вне этой кривой, будет называться аномалиями, потому что это нечто, что не попадает в множество типичных объектов из выборки.

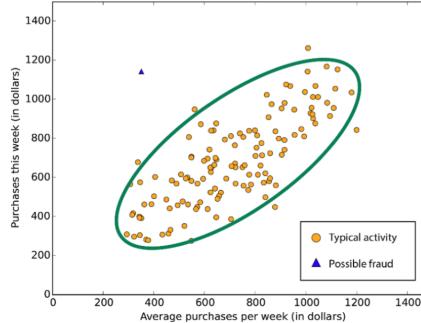


Рис. 5.12: Задача обнаружения аномалий

Как можно подружить эти две задачи? В задаче обнаружения аномалий тоже необходимо построить некоторую разделяющую поверхность, но при этом в наличии нет примеров аномалий, есть только примеры типичных объектов. Можно считать, что в задаче присутствуют 2 класса. Первый класс — это нормальные объекты, к нему относится вся обучающая выборка. Второй класс — это аномалии, и считается, что аномальным является начало координат. Теперь можно решать эту задачу, используя методы классификации классификации.

5.4.2. Применение метода опорных векторов для обнаружения аномалий

Для решения будет использоваться линейный способ, при этом необходимо выбирать разделяющую гиперплоскость так, чтобы она давала максимальный размер зазора, тогда вероятность переобучения будет минимальный. Это лучше всего делать с помощью метода опорных векторов, или SVM. Вот, как выглядит задача отделения выборки от начала координат:

$$\begin{cases} \frac{1}{2} \|w\|^2 + \frac{1}{v\ell} \sum_{i=1}^{\ell} \xi_i - \rho \rightarrow \min_{w, \xi, \rho} \\ \langle w, x_i \rangle \geq \rho - \xi_i, \quad \xi_i \geq 0 \end{cases}$$

Эта задача похожа на обычную задачу метода опорных векторов, но всё-таки отличается. Важный параметр задачи, v , задаёт верхнюю оценку для доли аномальных объектов в выборке. Если выбрать v так, что аномальными могут быть объявлены 3 объекта, то разделение может произойти как на рисунке 5.13.

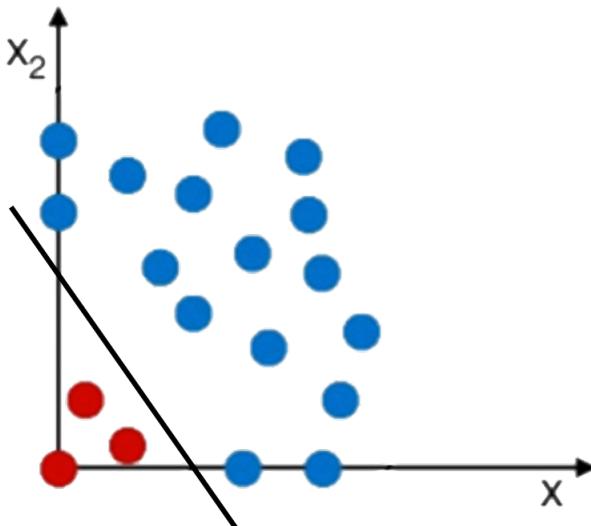


Рис. 5.13: Задача обнаружения аномалий

5.4.3. Классификация с использованием нелинейного ядра

Предположение о том, что 0 — это аномальный объект, очень странное. Например, если выборка центрирована вокруг начала координат, то этот подход в принципе не может дать правильный результат. На самом деле, одноклассовый SVM с линейным ядром никогда не используется. Скалярное произведение в сформулированной задаче можно заменить на ядро K . Популярный выбор — это RBF-ядро, которое вычисляется по формуле

$$K(x, z) = \exp\left(\frac{\|x - z\|^2}{\sigma^2}\right).$$

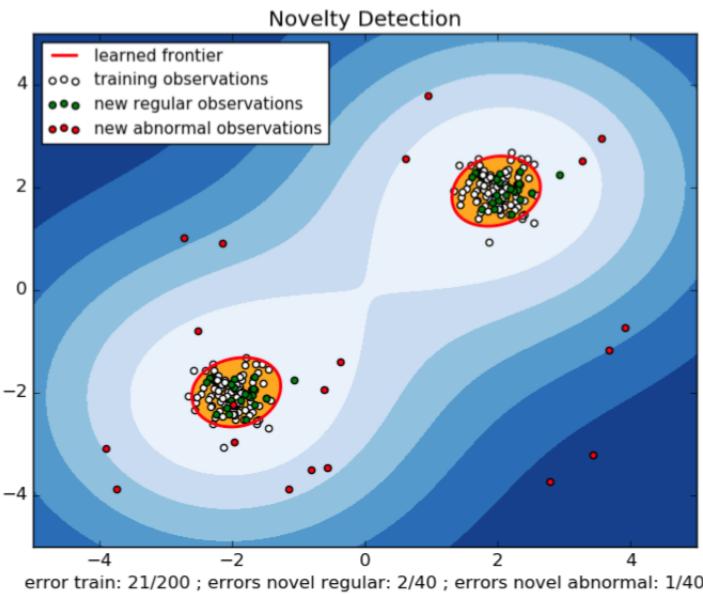


Рис. 5.14: Поиск аномалий с использованием ядрового одноклассового SVM с RBF-ядром

После замены ядра разделяющая плоскость будет строиться в пространстве более высокой размерности. Пример применения ядрового одноклассового SVM с RBF-ядром в выборке показан на рисунке 5.14.

Урок 6

Визуализация данных

6.1. Задача визуализации

6.1.1. Связь задачи визуализации и методов понижения размерности

Этот урок посвящён визуализации данных. Зачем визуализировать данные? Ответить на этот вопрос поможет пример: пусть имеются выборка, изображённая на рисунке 6.1, и требуется понизить её размерность до одного признака. Если не знать структуры данных, то можно долго и безуспешно пытаться спроектировать объекты на прямую, теряя при этом много информации. Однако если посмотреть на визуализацию данных, то становится понятно, что чтобы сохранить структуру, необходимо спроектировать данные на красную кривую, а для этого нужно использовать нелинейные методы понижения размерности.

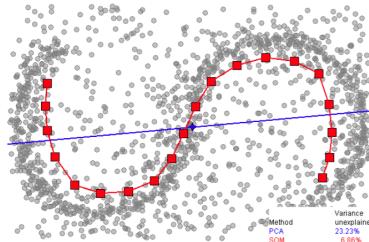


Рис. 6.1: Пример данных

В машинном обучении имеют дело с высокоразмерными выборками, то есть с выборками, в которых очень много признаков. Хочется понимать, как устроены данные, какие в них есть взаимосвязи, какие признаки важны. Для этого можно, как на рисунке 6.2, для каждой пары признаков спроектировать выборку на эту пару, а для каждого признака в отдельности построить гистограмму. Из этих графиков можно извлечь много информации: заметить, какие признаки лучше разделяют классы, или что даже какой-то один признак их хорошо разделяет.

Недостаток такого подхода — невозможность видеть всю выборку в целом. Хочется отобразить данные в двумерное или трёхмерное пространство, чтобы была видна их структура (как на рисунке 6.3): какие классы хорошо разделимы, какие — перемешаны между собой.

Итак, задача визуализации данных — это частный случай нелинейного понижения размерности, когда данные проецируются на плоскость или в трёхмерное пространство. При этом хочется спроектировать данные так, чтобы сохранить всю структуру и закономерности.

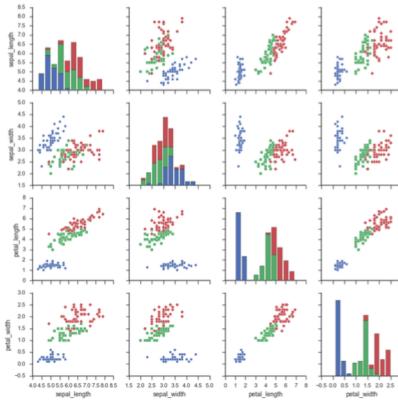


Рис. 6.2: Визуализация проекции данных на пары признаков

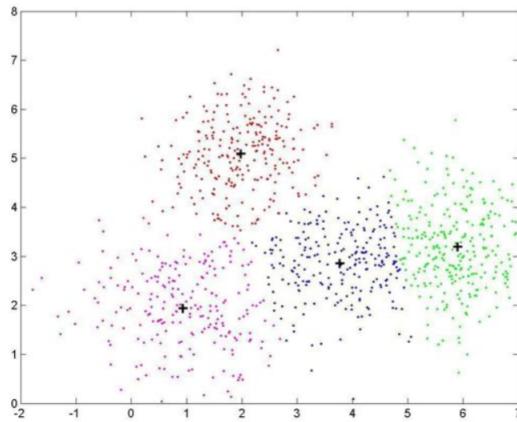


Рис. 6.3: Пример данных, отображённых в двумерное пространство

6.1.2. Поиск структуры в данных при помощи визуализации

Один из классических примеров, с помощью которого легко демонстрировать задачу визуализации — это набор данных MNIST (рисунок 6.4). Это изображения цифр, написанных от руки, причём все цифры очень разные.

0	1	2	3	4	5	0	1	3	4	5	0	1	2	3	4	5	0	5
5	5	0	4	1	3	5	1	0	0	2	2	0	1	2	3	3	3	3
4	4	1	5	0	5	1	2	0	0	1	3	2	1	4	3	1	3	1
3	4	4	0	5	3	1	5	4	4	2	2	2	5	5	4	4	0	0
2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5	0	5	5
0	4	4	3	5	1	0	0	2	2	1	0	1	2	3	3	3	4	4
4	5	0	5	2	1	0	0	1	3	2	1	3	4	3	4	4	3	4
1	5	0	5	2	1	0	0	1	3	2	1	3	4	3	4	4	3	4
0	5	7	4	5	4	4	1	2	2	5	5	4	4	0	0	1	2	3
5	0	4	1	2	3	4	5	0	4	2	3	4	5	0	5	5	5	0
4	5	4	0	0	2	2	2	0	4	2	3	3	3	4	4	1	5	0
0	5	2	2	0	0	1	3	2	1	0	1	2	3	3	3	4	4	5
5	2	2	0	0	1	3	2	1	0	1	2	3	3	3	4	4	5	0
3	1	5	4	4	2	2	2	5	5	4	4	0	3	0	1	1	3	4
1	5	4	4	2	2	2	5	5	4	4	0	3	0	1	1	3	4	5
0	1	2	3	4	5	0	4	1	2	3	4	5	0	5	5	5	0	4
1	5	4	4	2	2	2	5	5	4	4	0	3	0	1	1	3	4	5
2	2	0	0	1	3	2	1	0	1	2	3	3	3	4	4	1	5	0
0	0	1	2	3	4	5	0	4	1	2	3	3	3	4	4	1	5	0
5	1	0	0	1	2	2	0	1	3	3	3	3	4	4	4	1	5	0
1	2	0	0	1	3	2	1	4	4	3	1	3	1	4	3	1	4	0
0	1	2	0	0	1	3	2	1	4	4	3	1	3	1	4	3	1	4
5	2	2	0	0	1	3	2	1	4	4	3	1	3	1	4	3	1	4
1	5	4	4	2	2	2	5	5	4	4	0	0	1	2	3	4	5	0
2	3	4	5	0	1	2	3	4	5	0	5	5	5	0	4	1	3	5
3	1	0	0	1	2	2	0	1	3	3	3	3	4	4	4	1	5	0
0	0	1	2	2	0	1	3	3	3	3	4	4	4	4	4	1	5	0
5	0	1	2	3	4	5	0	4	1	2	3	3	3	4	4	1	5	0
1	5	4	4	2	2	2	5	5	4	4	0	0	1	2	3	4	5	0
2	3	4	5	0	1	2	3	4	5	0	5	5	5	0	4	1	3	5
3	1	0	0	1	2	2	0	1	3	3	3	3	4	4	4	1	5	0
0	0	1	2	2	0	1	3	3	3	3	4	4	4	4	4	1	5	0
5	0	1	2	3	4	5	0	4	1	2	3	3	3	4	4	1	5	0
1	5	4	4	2	2	2	5	5	4	4	0	0	1	2	3	4	5	0
2	3	4	5	0	1	2	3	4	5	0	5	5	5	0	4	1	3	5
3	1	0	0	1	2	2	0	1	3	3	3	3	4	4	4	1	5	0
0	0	1	2	2	0	1	3	3	3	3	4	4	4	4	4	1	5	0
5	0	1	2	3	4	5	0	4	1	2	3	3	3	4	4	1	5	0
1	5	4	4	2	2	2	5	5	4	4	0	0	1	2	3	4	5	0
2	3	4	5	0	1	2	3	4	5	0	5	5	5	0	4	1	3	5
3	1	0	0	1	2	2	0	1	3	3	3	3	4	4	4	1	5	0
0	0	1	2	2	0	1	3	3	3	3	4	4	4	4	4	1	5	0
5	0	1	2	3	4	5	0	4	1	2	3	3	3	4	4	1	5	0
1	5	4	4	2	2	2	5	5	4	4	0	0	1	2	3	4	5	0
2	3	4	5	0	1	2	3	4	5	0	5	5	5	0	4	1	3	5
3	1	0	0	1	2	2	0	1	3	3	3	3	4	4	4	1	5	0
0	0	1	2	2	0	1	3	3	3	3	4	4	4	4	4	1	5	0
5	0	1	2	3	4	5	0	4	1	2	3	3	3	4	4	1	5	0
1	5	4	4	2	2	2	5	5	4	4	0	0	1	2	3	4	5	0
2	3	4	5	0	1	2	3	4	5	0	5	5	5	0	4	1	3	5
3	1	0	0	1	2	2	0	1	3	3	3	3	4	4	4	1	5	0
0	0	1	2	2	0	1	3	3	3	3	4	4	4	4	4	1	5	0
5	0	1	2	3	4	5	0	4	1	2	3	3	3	4	4	1	5	0
1	5	4	4	2	2	2	5	5	4	4	0	0	1	2	3	4	5	0
2	3	4	5	0	1	2	3	4	5	0	5	5	5	0	4	1	3	5
3	1	0	0	1	2	2	0	1	3	3	3	3	4	4	4	1	5	0
0	0	1	2	2	0	1	3	3	3	3	4	4	4	4	4	1	5	0
5	0	1	2	3	4	5	0	4	1	2	3	3	3	4	4	1	5	0
1	5	4	4	2	2	2	5	5	4	4	0	0	1	2	3	4	5	0
2	3	4	5	0	1	2	3	4	5	0	5	5	5	0	4	1	3	5
3	1	0	0	1	2	2	0	1	3	3	3	3	4	4	4	1	5	0
0	0	1	2	2	0	1	3	3	3	3	4	4	4	4	4	1	5	0
5	0	1	2	3	4	5	0	4	1	2	3	3	3	4	4	1	5	0
1	5	4	4	2	2	2	5	5	4	4	0	0	1	2	3	4	5	0
2	3	4	5	0	1	2	3	4	5	0	5	5	5	0	4	1	3	5
3	1	0	0	1	2	2	0	1	3	3	3	3	4	4	4	1	5	0
0	0	1	2	2	0	1	3	3	3	3	4	4	4	4	4	1	5	0
5	0	1	2	3	4	5	0	4	1	2	3	3	3	4	4	1	5	0
1	5	4	4	2	2	2	5	5	4	4	0	0	1	2	3	4	5	0
2	3	4	5	0	1	2	3	4	5	0	5	5	5	0	4	1	3	5
3	1	0	0	1	2	2	0	1	3	3	3	3	4	4	4	1	5	0
0	0	1	2	2	0	1	3	3	3	3	4	4	4	4	4	1	5	0
5	0	1	2	3	4	5	0	4	1	2	3	3	3	4	4	1	5	0
1	5	4	4	2	2	2	5	5	4	4	0	0	1	2	3	4	5	0
2	3	4	5	0	1	2	3	4	5	0	5	5	5	0	4	1	3	5
3	1	0	0	1	2	2	0	1	3	3	3	3	4	4	4	1	5	0
0	0	1	2	2	0	1	3	3	3	3	4	4	4	4	4	1	5	0
5	0	1	2	3	4	5	0	4	1	2	3	3	3	4	4	1	5	0
1	5	4	4	2	2	2	5	5	4	4	0	0	1	2	3	4	5	0
2	3	4	5	0	1	2	3	4	5	0	5	5	5	0	4	1	3	5
3	1	0	0	1	2	2	0	1	3	3	3	3	4	4	4	1	5	0
0	0	1	2	2	0	1	3	3	3	3	4	4	4	4	4	1	5	0
5	0	1	2	3	4	5	0	4	1	2	3	3	3	4	4	1	5	0
1	5	4	4	2	2	2	5	5	4	4	0	0	1	2	3	4	5	0
2	3	4	5	0	1	2	3	4	5	0	5	5	5	0	4	1	3	5
3	1	0	0	1	2	2	0	1	3	3	3	3	4	4	4	1	5	0
0	0	1	2	2	0	1	3	3	3	3	4	4	4	4	4	1	5	0
5	0	1	2	3	4	5	0	4	1	2	3	3	3	4	4	1	5	0
1	5	4	4	2	2	2	5	5	4	4	0	0	1	2	3	4	5	0
2	3	4	5	0	1	2	3	4	5	0								



Рис. 6.5: Случайно сгенерированные изображения размера 28×28 пикселей

Требуется их разделить на 10 классов, то есть определить по изображению, какая цифра на нем нарисована. Каждая картинка в этом наборе данных имеет размер 28×28 пикселей, то есть всего у каждого объекта 784 признака. Но можно использовать намного меньше признаков, чтобы характеризовать каждую рукописную цифру. Это помогает понять следующий пример: если генерировать случайные картинки размером 28×28 пикселей, то результат будет похож на картинки, показанные на рисунке 6.5. На них нет ничего общего с изображениями цифр, и вероятность получить при случайной генерации что-то похожее на цифры очень мала. Это даёт понять, что внутренняя размерность данного набора данных сильно ниже.

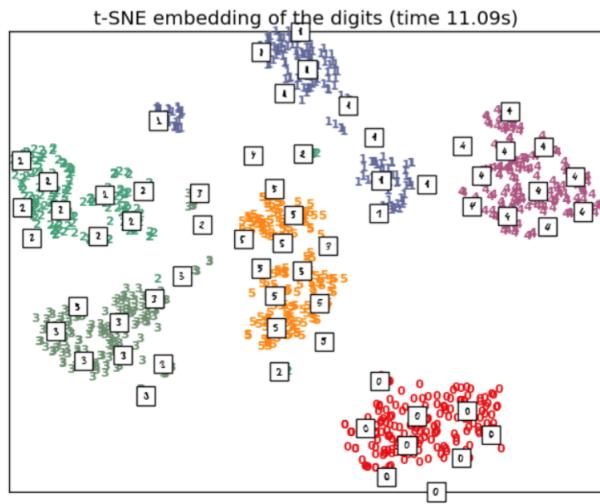


Рис. 6.6: Визуализация набора данных MNIST методом t-SNE

Если воспользоваться методом визуализации t-SNE (подробнее о нём будет сказано далее), то получится рисунок 6.6. Всего двух признаков достаточно, чтобы изобразить эти цифры так, что все классы будут идеально разделимы.

Ещё один пример взят из конкурса на сайте Kaggle. В этом конкурсе требовалось, используя описание и характеристики товара, определить, к какой из 9 категорий он относится. Всего признаков было 93. Если визуализировать этот набор данных, получится изображение, показанное на рисунке 6.7. При этом, если в исходном признаковом пространстве использовать для классификации сложные методы (случайный лес, градиентный бустинг), то видно, что в данных сохраняются те же закономерности, какие показаны на рисунке: хорошо отделимые на изображении классы разделяются хорошо, перемешанные классы — плохо.

6.2. Многомерное шкалирование

6.2.1. Постановка задачи

До этого речь уже шла о линейных методах понижения размерности, например, описывался метод случайных проекций:

$$z_{ij} = \sum_{k=1}^D w_{jk} x_{ik}$$

$$w_{jk} \sim \mathcal{N}(0, \frac{1}{d})$$

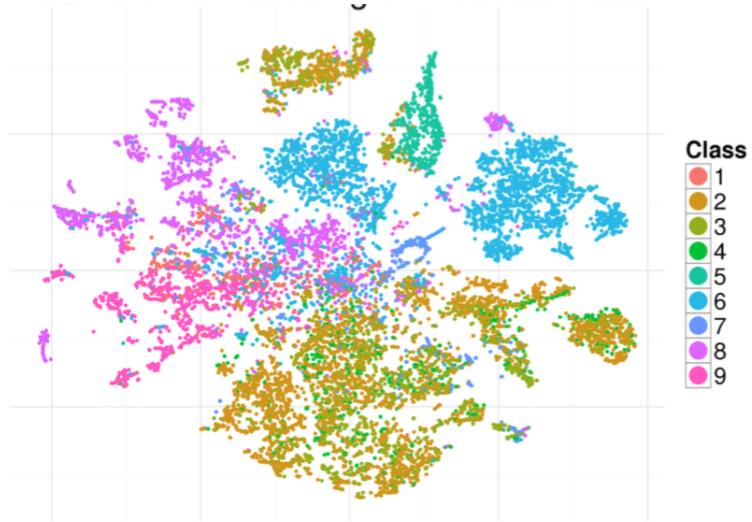


Рис. 6.7: Визуализация описаний товаров из конкурса на сайте Kaggle

Также обсуждался метод главных компонент, который выбирает веса более грамотно: он выражает их через сингулярные векторы матрицы «объекты-признаки».

Оба этих метода не могут определить нелинейные зависимости в данных. Найти их может, например, метод многомерного шкалирования (MDS). Для дальнейшего описания метода требуется ввести формальные обозначения:

- x_1, \dots, x_ℓ — объекты в исходном пространстве;
- $\tilde{x}_1, \dots, \tilde{x}_\ell$ — объекты в маломерном пространстве;
- $d_{ij} = \rho(x_i, x_j)$ — расстояния в исходном пространстве, ρ — метрика, необязательно евклидова;
- $\tilde{d}_{ij} = \|\tilde{x}_i - \tilde{x}_j\|$ — расстояния в маломерном пространстве, измеряются с использованием евклидовой метрики.

Итак, в задача многомерного шкалирования требуется, чтобы попарные расстояния между объектами изменялись как можно меньше:

$$\sum_{i < j}^{\ell} (\|\tilde{x}_i - \tilde{x}_j\| - d_{ij})^2 \rightarrow \min_{\tilde{x}_1, \dots, \tilde{x}_\ell} .$$

Стоит обратить внимание, что при использовании данного метода не требуется знать признаковое описание объектов, достаточно лишь уметь вычислять расстояния между ними.

6.2.2. Оптимизация метода многомерного шкалирования

Эта задача сложнее тех, которые встречались до этого, потому что в этот раз суммирование производится не по всем объектам, а по всем парам объектов. Решить задачу можно с помощью метода оптимизации SMACOF, но он довольно сложный, поэтому здесь описан не будет.

Результат многомерного шкалирования набора данных MNIST показан на рисунке 6.8. Видно, что данные неплохо разделены, однако и объекты внутри одного класса, и сами классы располагаются довольно близко друг к другу. Это результат того, что данные спроектированы из 728-мерного пространства, а чем больше размерность пространства, тем более похожи расстояния между различными парами объектов (проклятье размерности).

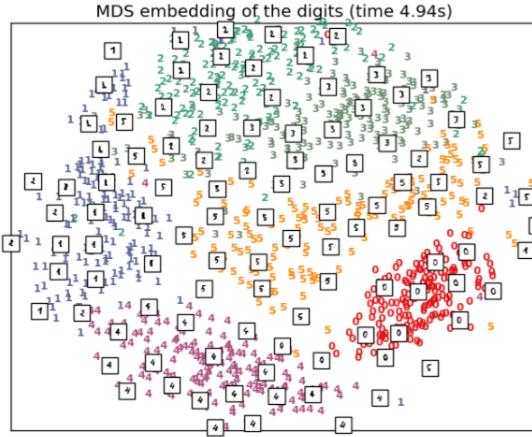


Рис. 6.8: Визуализация набора данных MNIST с помощью многомерного шкалирования

6.3. Метод t-SNE

6.3.1. Метод SNE

Решения задачи многомерного шкалирования, описанной ранее, получаются не очень качественными, потому что непросто сохранить попарные расстояния между объектами при радикальном уменьшении размерности пространства. Метод SNE (Stochastic Neighbor Embedding) пытается решить эту проблему: требуется не сохранение расстояний, а сохранение пропорций расстояний между объектами. Если

$$\rho(x_i, x_j) = \alpha \rho(x_i, x_k),$$

то

$$\rho(\tilde{x}_i, \tilde{x}_j) = \alpha \rho(\tilde{x}_i, \tilde{x}_k).$$

Чтобы формализовать эти понятия, необходимо ввести следующие условные вероятности:

$$p(x_j|x_i) = \frac{\exp(||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq i} \exp(||x_i - x_k||^2/2\sigma^2)}$$

$$q(\tilde{x}_j|\tilde{x}_i) = \frac{\exp(||\tilde{x}_i - \tilde{x}_j||^2/2\sigma^2)}{\sum_{k \neq i} \exp(||\tilde{x}_i - \tilde{x}_k||^2/2\sigma^2)}.$$

Это распределение учитывает только соотношения между расстояниями. Чтобы сравнить эти два распределения, необходимо использовать функционал, который находит различия между двумя вероятностными распределениями. Для этих целей хорошо подходит дивергенция Кульбака-Лейблера:

$$p(x_j|x_i) \log \frac{p(x_j|x_i)}{q(\tilde{x}_j|\tilde{x}_i)} \rightarrow \min_{\tilde{x}_1, \dots, \tilde{x}_\ell}$$

Для нахождения описания объекта в маломерном признаковом пространстве необходимо минимизировать этот функционал (например, методом стохастического градиентного спуска).

6.3.2. Метод t-SNE

У метода SNE есть проблема: объекты в многомерном пространстве легче разместить рядом, чем в маломерном, а используемая евклидова метрика слишком сильно штрафует за несохранение пропорций. Метод t-SNE старается решить эту проблему, используя другой способ вычисления распределений в новом пространстве:

$$q(\tilde{x}_j|\tilde{x}_i) = \frac{(1 + \|\tilde{x}_i - \tilde{x}_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|\tilde{x}_i - \tilde{x}_k\|^2)^{-1}}$$

При использовании этого метода разделение оказывается гораздо более чётким. Визуализация набора данных MNIST с помощью этого метода показана на рисунке 6.6.

Урок 7

Тематическое моделирование

Этот модуль будет посвящен вероятностному тематическому моделированию — направление, которое развивается в машинном обучении очень активно последние 15 лет. Постоянно появляются новые постановки задач, новые прикладные задачи и новые методы.

7.1. Введение в тематическое моделирование

7.1.1. Понятие темы в тематическом моделировании

Тематическое моделирование — современное направление статистического анализа текстов. Методы тематического моделирования призваны ответить на вопрос, какой теме посвящена большая коллекция текстовых документов.

Интуитивно под темой понимается специальная терминология определенной предметной области. Более точно, **тема** — набор **терминов**, то есть слов или словосочетаний, которые совместно часто встречаются в документах. Интуитивное понимание исходит из того факта, что документ по математике легко можно отличить от документа по биологии за счет узнаваемых терминов даже без знания этих предметных областей. Более того, если термины в документах будут случайным образом переставлены, все равно можно будет понять тему этого документа.

Более формально эти соображения можно изложить на языке теории вероятностей и статистики. В этом случае речь пойдет о частотах встречаемости слов в коллекции документов.

Каждая тема характеризуется своим словарем и своими вероятностями терминов из этого словаря, $p(w|t)$ — вероятность (частота) термина w в теме t . Так как документ может относиться одновременно к нескольким темам (как говорят, к вероятностной смеси тем), существует понятие тематики документа. **Тематика документа** — условное распределение $p(t|d)$, где t — тема, а d — документ.

Тематическая модель должна автоматически выявлять латентные темы по наблюдаемым частотам терминов в документе $p(w|d)$. Поскольку неизвестно, о какой теме думал автор, когда писал документ, тема документа не наблюдается непосредственно и по этой причине называется латентной.

7.1.2. Цели тематического моделирования

Автоматический анализ текста

Тематическое моделирование может, в том числе, использоваться для автоматического анализа текстов для решения целого множества задач:

- **Классификация и категоризация документов:** необходимо, опираясь на тематику текстов, «разложить по папкам» большую коллекцию или поток документов. Тематическое моделирование обладает особенностью, что позволяет относить документ сразу ко многим темам.
- **Автоматическое аннотирование документов:** необходимо выделить в документе наиболее важные фразы и составить на их основе его краткий обзор. Тематическое моделирование позволяет, определив тематику документа, выделить наиболее «тематичные» фразы и сделать из них выборку таким образом, чтобы были покрыты все имеющиеся в документе темы.
- **Автоматическая суммаризация коллекций** очень похожа на предыдущую задачу, но в этом случае необходимо подготовить обзор не одного документа, а большой коллекции или какой-то ее части.

- **Тематическая сегментация документов:** необходимо разбить длинный документ на тематически однородные фрагменты и определить тематику каждого такого фрагмента.

Общая идея решения всех представленных задач состоит в том, что распределение тем $p(t|d)$ в документе становится признаком описанием документа d . Часто при анализе текстов используются векторные модели, в которых каждая частота каждого слова или термина дает свой признак. Число признаков в таких моделях очень велико и совпадает с числом терминов в словаре. Тематическое моделирование позволяет перейти к сжатому признаковому описанию, в котором каждый признак соответствует одной теме.

Систематизация больших объёмов информации

Другой большой класс задач составляют задачи, связанные с новыми методами поиска текстовой информации и помощи в ее понимании. Это особенно необходимо людям, которые постоянно занимаются самообразованием, чтобы быстро находить нужные знания в новых для себя областях. Такой тип поиска называется семантическим или разведочным (название еще не устоялось). Для решения такого рода задач необходимо структурировать общие представления об устройстве предметной области. В частности определить, как тема делится на подтемы.

Тематическое моделирование для цели систематизации больших объемов информации позволяет решать следующие задачи:

- **Семантический (разведочный) поиск информации**
- **Визуализация тематической структуры коллекции**
- **Анализ динамики развития тем**, особенно, если к каждому документу привязана временная метка.
- **Тематический мониторинг новых поступлений**, который предоставляет возможность автоматически сообщать пользователям о появлении в сети Интернет или какой-то библиотеке новых тематических документов без необходимости постоянно производить повторный поиск.
- **Рекомендация документов пользователям**, то есть когда необходимо построить рекомендательную систему, которая использует данные о прошлой активности пользователя и данные об активности других пользователей.

7.1.3. Приложения тематического моделирования

Можно кратко перечислить основные приложения тематического моделирования:

- **Поиск научной информации, трендов, фронта исследований.** По разным оценкам множество статей и публикаций ежегодно увеличивается примерно на 10 миллионов документов.
- **Подбор экспертов, рецензентов, исполнителей проектов.** Для автоматизации деятельности экспертовых советов необходимо анализировать огромное количество заявок на инновационные проекты и научные гранты, чтобы быстро подобрать экспертов и распределить пришедшую коллекцию документов на экспертизу.
- **Агрегирование новостных потоков:** необходимо в приходящем из разных источников новостном потоке определять тематику каждого документа, находить дубликаты и отслеживать каждую тему во времени. С этой задачей сталкиваются многие компании, которые занимаются социологическими исследованиями, аналитические компании, информационные агентства и так далее. Тематическое моделирование позволяет автоматизировать эту деятельность так, чтобы с этой задачейправлялось меньшее число сотрудников.
- **Аннотирование и поиск изображений** — задача, не связанная с текстовой аналитикой, но при решении которой применять те же методы для совершенно другого типа данных. Действительно, если существуют методы выделения тех или иных элементов изображений, то такие элементы можно рассматривать в качестве терминов, а сами изображения — в качестве документов.
- Более того, некоторые изображения сопровождаются текстом. Появляется возможность сразу строить два представления документа: основанное на выделении графических элементов и основанное на анализе сопровождающего текста. Это позволяет для такого рода многомодальных коллекций реализовывать поиск текстов по изображениям, изображения по тексту, а также аннотировать изображения и так далее.
- **Анализ видеопоследовательностей, аннотация генома и другие задачи биоинформатики, анализ дискретизированных биомедицинских сигналов, мониторинг состояния технических**

систем — еще несколько примеров задач, в которых все чаще используются методы тематического моделирования. В некоторых задачах сигнал был исходно дискретным, в некоторых — изначально вещественным. Во втором случае его необходимо дискретизировать, чтобы после дискретизации была возможность представить его в виде символьной последовательности и использовать методов символьной лингвистики, машинного обучения и тематического моделирования для ее анализа.

7.1.4. Примеры использования тематического моделирования

Мультиязычная модель Википедии

При тематизации википедии была построена мультиязычная модель (на русском и английском языка), которая смогла сама, без помощи эксперта, собрать темы на обоих языках. Было обработано 216 175 русско-английских пар статей Википедии и собрано 400 тем.

тема 68			тема 79		
research	4.56	институт	6.03	goals	4.48
technology	3.14	университет	3.35	league	3.99
engineering	2.63	программа	3.17	club	3.76
institute	2.37	учебный	2.75	season	3.49
science	1.97	технический	2.70	scored	2.72
program	1.60	технология	2.30	cup	2.57
education	1.44	научный	1.76	goal	2.48
campus	1.43	исследование	1.67	apps	1.74
management	1.38	наука	1.64	debut	1.69
programs	1.36	образование	1.47	match	1.67
тема 88			тема 251		
opera	7.36	опера	7.82	windows	8.00
conductor	1.69	оперный	3.13	microsoft	4.03
orchestra	1.14	дирижер	2.82	server	2.93
wagner	0.97	певец	1.65	software	1.38
soprano	0.78	певица	1.51	user	1.03
performance	0.78	театр	1.14	security	0.92
mozart	0.74	партия	1.05	mitchell	0.82
sang	0.70	сопрано	0.97	oracle	0.82
singing	0.69	вагнер	0.90	enterprise	0.78
operas	0.68	оркестр	0.82	users	0.78

Таблица 7.1: Первые 10 слов (с их вероятностями $p(w|t)$ в %) в каждой из представленных тем.

Эти темы оказались легко интерпретируемыми. Более того, модель выявляет двуязычные темы без выравнивания, без словарей, даже когда тексты не являются точными переводами. В этом эксперименте независимый эксперт оценил 396 тем из 400 как хорошо интерпретируемые.

Биграммная модель термины – словосочетания

Оказывается, что интерпретируемость тем, понятность для человека резко возрастает, если в качестве терминов использовать не отдельные слова, а словосочетания. Например, была построена тематическая модель для коллекции научных статей. Это статьи конференций «Математические методы распознавания образов» и «Интеллектуализация обработки информации» на русском языке.

распознавание образов в биоинформатике		теория вычислительной сложности	
unigrams	bigrams	unigrams	bigrams
объект	задача распознавания	задача	разделять множества
задача	множество мотивов	множество	конечное множество
множество	система масок	подмножество	условие задачи
мотив	вторичная структура	условие	задача о покрытии
разрешимость	структура белка	класс	покрытие множества
выборка	распознавание вторичной	решение	сильный смысл
маска	состояние объекта	конечный	разделяющий комитет
распознавание	обучающая выборка	число	минимальный аффинный
информационность	оценка информативности	аффинный	аффинный комитет
состояние	множество объектов	случай	аффинный разделяющий
закономерность	разрешимость задачи	покрытие	общее положение
система	критерий разрешимости	общий	множество точек
структура	информационность мотива	пространство	случай задачи
значение	первичная структура	схема	общий случай
регулярность	тупиковое множество	комитет	задача MASC

Таблица 7.2: Несколько тем, построенные по 850 статьям конференций ММРО, ИОИ на русском языке.

Модель, в которой в качестве терминов используются отдельные слова, легко интерпретируется, но понять авторов и научную школу по списку терминов практически невозможно. Другое дело в случае биграммной модели, где словарь состоит из пар слов: разбираясь в тематике конференции, можно даже соотнести конкретных авторов с некоторой темой.

Поиск этно-релевантных тем в социальных сетях

В целях поддержки социологических исследований в области межэтнических отношений была поставлена задача создать систему разведочного поиска для систематизации и мониторинга этно-релевантных тем.

Заранее специалистами был создан словарь этнонимов (более 800 слов), который потом использовался, чтобы выделить в социальной сети контент, который относится к обсуждению подобного рода тем (его оказалось не более 1%). После этого, уже без участия экспертов, в выделенном контенте были выделены этно-релевантные темы. Например:

русские: русский, князь, россия, татарин, великий, царь, царина, император, империя, грозить, государь, екатерина, акция, организация, митинг, движение, активный, мероприятие, пикет, русский, участник, москва, оппозиция.

сирийцы: сирийский, асад, боевик, район, террорист, уничтожать, группировка, дамаск, оружие, алесио, оппозиция

таджики и узбеки: мигрант, страна, россия, миграция, азия, нелегальный, миграционный, таджикистан, гастарбайтер

канадцы: команда, игра, игрок, канадский, сезон, хоккей, сборная, играть, болельщик, победа, кубок, счет, чемпионат

норвежцы: дитя, ребенок, родиться, детский, семья, воспитанный, право, возраст, отец, опека, норвежский, сын

китайцы: китайский, россия, производство, китай, страна, продукция, предприятие, компания, технология, военный

По этим данным можно понять, что, если обсуждаются русские, то тема либо связана с историей России, либо с текущей политикой. Сирийцев обсуждают в связи с войной в Сирии, таджиков и узбеков — в связи с трудовой миграцией. Разговоры про канадцев в нашей социальной сети идут про то, как они играют в хоккей. Когда говорят про норвежцев, обсуждают чаще всего проблематику ювенальной юстиции, а когда говорят про китайцев, то производство и деловые отношения.

7.2. Постановка задачи тематического моделирования

В этом видео речь пойдет о формальной постановке задачи тематического моделирования и о вероятностной порождающей модели текста.

7.2.1. Подготовка данных для тематического моделирования

Прежде чем применять формальные методы, тексты необходимо подготовить. Предварительная обработка и очистка текстов производится по-разному для разных типов текста и может включать в себя:

- **Удаление форматирования и переносов**
- **Удаление обрывочной и нетекстовой информации.** Например, если коллекция была получена из PDF файлов, в файлах могут содержаться артефакты: разрывы слов на месте перевода строки, колонтитулы, остатки графиков, таблиц, а также различные нечитаемые символы и так далее. Все это должно быть удалено из исходных текстов.
- **Исправление опечаток**
- **Слияние слишком коротких текстов.** Тексты из социальных сетей, например сообщения из twitter, часто очень короткие, поэтому необходимо применять разные методики для того, чтобы сливать короткие тексты в один более длинный и более удобный для тематического моделирования. Как именно это делается, зависит от цели исследования. В частности, можно слить в один все тексты автора за некоторый день. Или же объединять в один вообще все тексты данного автора.

После того, как это сделано, текст представляет собой последовательность слов без лишнего. После этого необходимо выполнить формирование словаря:

- **Выделение терминов (term extraction).** В научных текстах существует хорошо устоявшаяся терминология, причем чаще всего термины представляют собой словосочетания, а не отдельные слова. Поэтому важно сделать выделение терминов, используя специальные методики. Некоторые из них, так называемые методы предобработки, выполняются отдельно от тематического моделирования. Другие можно прямо встроить в тематические модели, но о таких сложных моделях речь идти не будет.
- **Приведение слов к нормальной форме** позволяет решить проблему со словоформами и может быть выполнено используя **стемминг** (лучше подходит для английских текстов) или **лемматизацию** (лучше для русских текстов).
- **Удаление стоп-слов и слишком редких слов.** Слова, которые встречаются слишком часто, не помогают определить тематику текста и называются **стоп-словами**. Также не помогут слишком редкие слова. Действительно, тема — это некое статистическое явление, набор слов, которые совместно часто встречаются в определенных текстах. Если слово встречается очень редко, как правило, ни о какой статистике речи идти не может. Поэтому слова, которые встретились реже примерно десяти раз, просто удаляются из коллекции. Удаление стоп-слов и слишком редких слов позволяет сократить словарь и уменьшить вычислительную сложность задачи.

7.2.2. Базовые предположения простых тематических моделей

Базовые предположения простых тематических моделей включают в себя следующие:

- **Порядок документов в коллекции не важен**
- **Порядок терминов в документе не важен (bag of words).** Каждый текст предполагается по-следовательностью слов, которые генерируются из некоторого вероятностного распределения. В самых простых моделях используется предположение о «мешке слов», согласно которому, даже переставив в документе слова или выделенные словосочетания, можно определить его тематику.
- **Употребление каждого слова в каждом документе связано с некоторой темой**, то есть каждая пара (d, w) связана с некоторой темой $t \in T$. Следовательно, коллекция документов представляет собой последовательность троек (d, w, t) , в которой темы являются латентными: они не видны и для их определения как раз используется тематическая модель.

- Гипотеза условной независимости: $p(w|t, d) = p(w|t)$ заключается в том, что вероятность слова в документах определяется только темой, а не самим документом. Это предположение позволяет строить легко оцениваемые тематические модели.

Часто используются дополнительные предположения разреженности:

- Предположение, что документ относится к небольшому числу тем.
- Предположение, что тема состоит из небольшого числа терминов, лексического ядра, которое существенно отличает эту тему от остальных.

7.2.3. Вероятностный процесс порождения текстовой коллекции

В вероятностной порождающей модели документ d — это смесь распределений $p(w|t)$ с весами $p(t|d)$:

$$p(w|d) = \sum_{t \in T} p(w|t) p(t|d).$$

Условное распределение тем в документе $p(t|d)$ — важный параметр модели, который и необходимо оценивать.

Таким образом, процесс порождения текста следующий. Для каждой словопозиции w сначала из распределения тем в документе выбирается тема, к которой это слово будет относиться. После этого из распределения слов в выбранной теме выбирается конкретное слово, которое будет записано в данную словопозицию. Слово за словом так появляется весь текст.

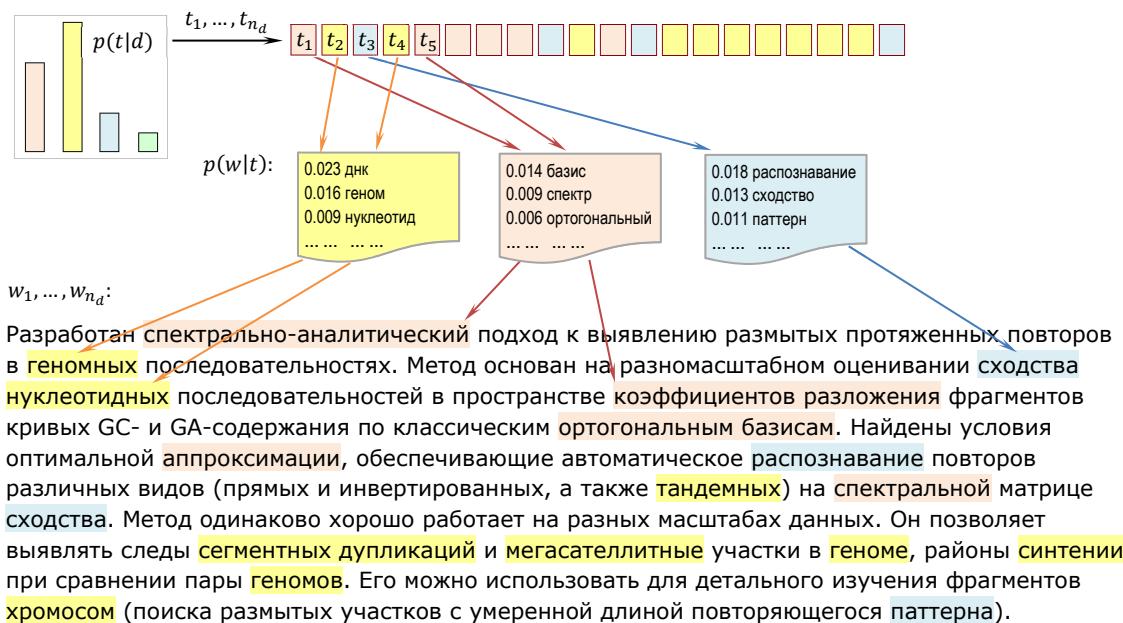
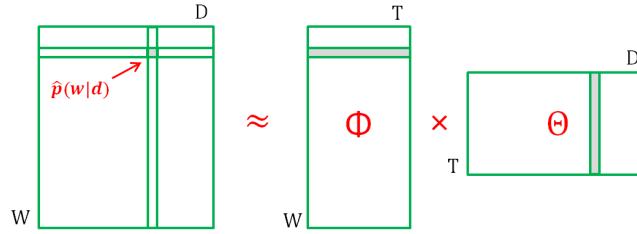


Рис. 7.1: Процесс порождения текстового документа вероятностной тематической моделью

Поскольку выполняется гипотеза «мешка слов» сгенерированный текст вряд ли будет осмысленным. Можно только говорить о том, что с точностью до произвольной перестановки слов, этот текст вполне мог бы нести в себе какую-то тематику. А именно тематику текста и нужно выявить. Другими словами, тематическое моделирование не обеспечивает понимание компьютером смысла текста, а только позволяет выполнить кластеризацию документов по темам.

7.2.4. Постановка задачи тематического моделирования

Формальная постановка задачи тематического моделирования следующая. Пусть зафиксирован словарь терминов W , из элементов которого складываются документы, и дана коллекция D документов $d \subset W$. Для каждого документа d известна его длина n_d и количество n_{dw} использований каждого термина w .



Требуется найти параметры вероятностной порождающей тематической модели, то есть представить вероятность появления $p(w|d)$ слов в документе в виде:

$$p(w|d) = \sum_{t \in T} \phi_{wt} \theta_{td},$$

где $\phi_{wt} = p(w|t)$ — вероятности терминов w в каждой теме t , $\theta_{td} = p(t|d)$ — вероятности тем t в каждом документе d .

Порождающая модель описывает процесс построения коллекции по ϕ_{wt} и θ_{td} . Тематическое моделирование представляет собой обратную задачу: по наблюдаемой коллекции необходимо понять, какими распределениями ϕ_{wt} и θ_{td} она могла бы быть получена.

7.2.5. Задача тематического моделирования как задача матричного разложения

Фактически, эту задачу можно трактовать как задачу матричного разложения. Пусть Φ — матрица распределений терминов в темах, а Θ — матрица распределений тем в документах:

$$\Phi = (\phi_{wt}), \quad \Theta = (\theta_{td}).$$

Матрицы называются стохастическими, если каждый их столбец представляет собой дискретное распределение вероятностей, а, следовательно, сумма значений по каждому столбцу равна 1 (условие нормировки) и каждое значение является неотрицательным (условие неотрицательности). Следует особо отметить, что стохастические матрицы — это НЕ такие матрицы, элементы которых генерируются случайно. Обе определенные выше матрицы Φ и Θ — стохастические. Согласно вероятностной тематической модели, произведение матриц Φ и Θ должно давать частотные оценки $p(w|d)$ условных вероятностей слов в документах коллекции. Наблюдаемые частоты терминов в документах известны:

$$\hat{p}(w|d) = \frac{n_{dw}}{n_d}.$$

Задача тематического моделирования, таким образом, стала задачей стохастического матричного разложения матрицы $(\hat{p}(w|d))$ на стохастические матрицы Φ и Θ .

Теперь можно воспользоваться принципом максимума правдоподобия с ограничениями, следующими из условий нормировки и неотрицательности на элементы стохастических матриц. Если максимизировать логарифм правдоподобия, получается:

$$\begin{cases} \sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} \rightarrow \max_{\Phi, \Theta}; \\ \sum_{w \in W} \phi_{wt} = 1; \quad \phi_{wt} \geq 0; \\ \sum_{t \in T} \theta_{td} = 1; \quad \theta_{td} \geq 0. \end{cases}$$

7.2.6. Принцип максимума регуляризованного правдоподобия

Задача матричного разложения некорректно поставлена, поскольку её решение в общем случае не единственное:

$$\Phi \Theta = (\Phi S)(S^{-1} \Theta) = \Phi' \Theta'$$

С одной стороны, строящаяся математическая модель получается неустойчивой и невоспроизводимой (результат работы итерационных методов будет зависеть от начального приближения), но, с другой стороны, это

дает свободу выбора дополнительных ограничений на матрицы Φ и Θ . В теории некорректно поставленных задач, то есть решение которых не единственno, такие ограничения принято называть регуляризаторами.

Чтобы из множества решений выбрать наиболее подходящее, вводится **критерий регуляризации** $R(\Phi, \Theta)$, который представляет собой некоторый функционал, построенный исходя из тех или иных содержательных соображений данной задачи. Теперь вместо задачи максимизации логарифма правдоподобия рассматривается задача максимизации регуляризованного правдоподобия, то есть суммы логарифма правдоподобия и регуляризатора $R(\Phi, \Theta)$:

$$\begin{cases} \sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} + R(\Phi, \Theta) \rightarrow \max_{\Phi, \Theta}; \\ \sum_{w \in W} \phi_{wt} = 1; \quad \phi_{wt} \geq 0; \\ \sum_{t \in T} \theta_{td} = 1; \quad \theta_{td} \geq 0. \end{cases}$$

7.3. Базовые тематические модели и ЕМ-алгоритм

7.3.1. Регуляризованный ЕМ-алгоритм

Выше было получено, что для построения тематической модели $p(w|d) = \sum_{t \in T} \phi_{wt} \theta_{td}$ по наблюдаемым частотам $\hat{p}(w|d) = \frac{n_{dw}}{n_d}$ нужно решать задачу максимизации логарифма правдоподобия с регуляризатором $R(\Phi, \Theta)$:

$$\begin{cases} \sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} + R(\Phi, \Theta) \rightarrow \max_{\Phi, \Theta}; \\ \sum_{w \in W} \phi_{wt} = 1; \quad \phi_{wt} \geq 0; \\ \sum_{t \in T} \theta_{td} = 1; \quad \theta_{td} \geq 0. \end{cases}$$

Регуляризатор может быть произвольным, однако удобно выбрать его гладким, чтобы можно было применить условие Каруша — Куна — Таккера. В результате получается система уравнений относительно относительно ϕ_{wt} , θ_{td} и вспомогательных переменных $p_{tdw} = p(t|d, w)$ (выкладки не приводятся):

$$\begin{cases} p_{tdw} = \text{norm}_{t \in T} (\phi_{wt} \theta_{td}) \\ \phi_{wt} = \text{norm}_{w \in W} \left(\sum_{d \in D} n_{dw} p_{tdw} + \phi_{wt} \frac{\partial R}{\partial \phi_{wt}} \right) \\ \theta_{td} = \text{norm}_{t \in T} \left(\sum_{w \in d} n_{dw} p_{tdw} + \theta_{td} \frac{\partial R}{\partial \theta_{td}} \right) \end{cases}$$

где операция нормировки вектора определена следующим образом:

$$\text{norm}_{t \in T} (x_t) = \frac{\max\{x_t, 0\}}{\sum_{s \in T} \max\{x_s, 0\}}.$$

Полученную систему удобно решать методом простых итераций, который в данном случае по совместительству оказывается ЕМ-алгоритмом. Первое уравнение системы (**Е-шаг**):

$$p_{tdw} = \text{norm}_{t \in T} (\phi_{wt} \theta_{td})$$

представляет собой вычисление с помощью формулы Байеса вспомогательных переменных $p_{tdw} = p(t|d, w)$ через параметры модели ϕ_{wt} и θ_{td} . Оставшиеся уравнения (**М-шаг**):

$$\phi_{wt} = \text{norm}_{w \in W} \left(\sum_{d \in D} n_{dw} p_{tdw} + \phi_{wt} \frac{\partial R}{\partial \phi_{wt}} \right), \quad \theta_{td} = \text{norm}_{t \in T} \left(\sum_{w \in d} n_{dw} p_{tdw} + \theta_{td} \frac{\partial R}{\partial \theta_{td}} \right)$$

наоборот, выражают основные параметры модели через вспомогательные переменные. В случае отсутствия регуляризатора ($R(\Phi, \Theta) = 0$) эти формулы представляют собой частотные оценки условных вероятностей слов в темах и тем в документах.

EM-алгоритм заключается в последовательном выполнении **E-шага** и **M-шага** до достижения требуемой точности, то есть является итерационным процессом.

Следует обратить особое внимание на то, что оператор нормировки переводит произвольный вектор x_t в вектор, координаты которого неотрицательны и нормированы, то есть сумма которых в точности равна 1. Другими словами, дискретное распределение можно задавать с помощью любого вектора нужной размерности, если сперва отнормировать его таким образом.

Два самых известных частных случая этой системы уравнений (при разном выборе регуляризатора):

- **PLSA, вероятностный латентный семантический анализ.** В этом случае регуляризатор не используется, то есть:

$$R(\Phi, \Theta) = 0.$$

Этот метод был придуман и опубликован Томасом Хоффманом в 1999 году.

- **LDA, латентное размещение Дирихле:**

$$R(\Phi, \Theta) = \sum_{t,w} (\beta_w - 1) \ln \phi_{wt} + \sum_{d,t} (\alpha_t - 1) \ln \theta_{td}$$

где $\beta_w > 0$, $\alpha_t > 0$ — параметры регуляризатора. Этот метод был предложен четыре года спустя Дэвидом Блеем, Эндрю Энджи и Майклом Джорданом. Их статья по латентному размещению Дирихле — это, наверное, самая цитируемая работа в тематическом моделировании.

7.3.2. Байесовская интерпретация модели LDA

Авторы метода латентного размещения Дирихле рассматривали его с точки зрения байесовского обучения, то есть вычисляли апостериорные оценки параметров, а не использовали принцип максимизации правдоподобия. Использовалось предположение, что столбцы ϕ_t матрицы Φ порождаются $|W|$ -мерным распределением Дирихле с вектором параметров $\beta = (\beta_w)$. (Фактически, имеет место двухступенчатая порождающая модель текста: сначала из распределения Дирихле порождаются столбцы матриц Φ и Θ , а потом на их основе порождается текстовая коллекция.)

Распределение Дирихле нужной размерности порождает нормированные и неотрицательные векторы, которые могут использоваться в качестве вероятностных распределений. Распределение Дирихле удобно использовать в качестве априорных распределений в байесовской интерпретации, поскольку распределение Дирихле позволяет генерировать как разреженные (при малых значениях параметров), так и такие вот сильно сглаженные дискретные распределения (при больших значениях параметров). Если все параметры распределения Дирихле в точности равны 1, получается равномерное распределение.

Апостериорные оценки доставляют максимум правдоподобия с использованием регуляризатора

$$R(\Phi, \Theta) = \sum_{t,w} (\beta_w - 1) \ln \phi_{wt} + \sum_{d,t} (\alpha_t - 1) \ln \theta_{td}.$$

Другими словами, вместо вычисления априорных оценок можно использовать принцип максимума правдоподобия с представленным регуляризатором. Именно в этой формулировке метод LDA был представлен выше по тексту.

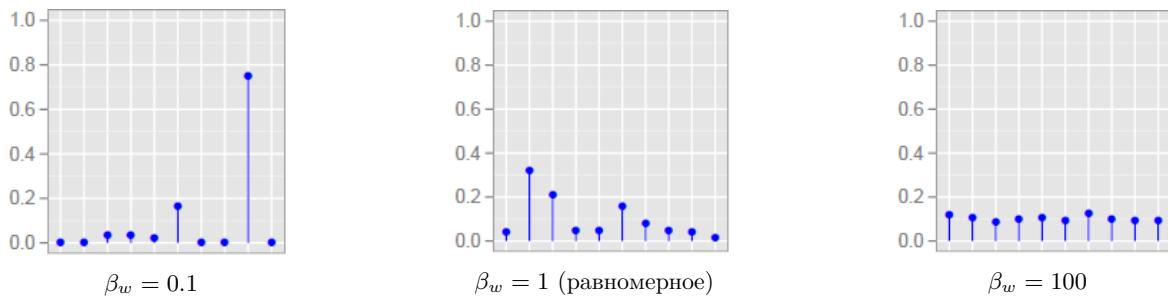


Рис. 7.2: Распределение $\phi \sim \text{Dir}(\beta)$ при $|W| = 10$, $\phi, \beta \in \mathbb{R}^{10}$

7.3.3. Дивергенция Кульбака–Лейблера

Существует другой, более простой, взгляд на латентное размещение Дирихле. Прежде, чем перейти к нему, необходимо ввести понятие дивергенции Кульбака — Лейблера. Оно часто встречается в машинном обучении и теории вероятностей и является чем-то вроде расстояния между распределениями.

Пусть даны два дискретных распределения $P = (p_i)_{i=1}^n$ и $Q = (q_i)_{i=1}^n$, тогда дивергенция Кульбака — Лейблера

$$\text{KL}(P\|Q) = \sum_i p_i \log \frac{p_i}{q_i}.$$

(В случае непрерывных распределений вместо суммы должен стоять интеграл)

Дивергенция Кульбака — Лейблера обладает следующими основными свойствами:

1. Неотрицательность:

$$\text{KL}(P\|Q) \geq 0; \quad \text{KL}(P\|Q) = 0 \Leftrightarrow P = Q.$$

2. Несимметричность:

$$\text{KL}(P\|Q) \neq \text{KL}(Q\|P).$$

Дивергенция Кульбака — Лейблера не является симметричной, а следовательно, не совсем правильно ее называть функцией расстояния между распределениями. Более корректно говорить, что она меряет в некотором смысле степень вложенности распределения P в распределение Q .

3. Связь с принципом максимума правдоподобия:

$$\sum_{i=1}^n p_i \ln \frac{p_i}{q_i(\alpha)} \rightarrow \min_{\alpha} \Leftrightarrow \sum_{i=1}^n p_i \ln q_i(\alpha) \rightarrow \max_{\alpha}.$$

То есть минимизация дивергенции Кульбака — Лейблера эквивалентна максимизации правдоподобия. Если P — эмпирическое распределение, а Q — это какая-то параметрическая модель распределения с параметром α , то чтобы определить такое значение α , при котором P как можно лучше соответствовало модели не имеет значения, минимизировать дивергенцию Кульбака — Лейблера или максимизировать правдоподобие.

Последнее свойство дает интуитивное понимание смысла дивергенции Кульбака — Лейблера. Далее она будет применена в роли регуляризатора.

7.3.4. Не-байесовская интерпретация модели LDA

Пусть $\beta = (\beta_w)$ — некоторый вектор над словарем W .

Если для некоторого слова $w \in W$ выполнено $\beta_w > 1$, то в модели LDA распределение вероятности ϕ_{wt} этого слова по темам будет сглаживаться, приближаясь к β_w^+ :

$$\text{KL}(\beta^+ \parallel \phi_t) \rightarrow \min, \quad \beta_w^+ = \underset{w \in W}{\text{norm}}(\beta_w - 1)$$

При $\beta_w < 1$, наоборот, значения ϕ_{wt} будут разреживаться, удаляясь от β_w^- к нулю:

$$\text{KL}(\beta^- \parallel \phi_t) \rightarrow \max, \quad \beta_w^- = \underset{w \in W}{\text{norm}}(1 - \beta_w),$$

то есть среди ϕ_{wt} будет больше нулевых или почти нулевых элементов.

В отличие от байесовской точки зрения, такая интерпретация не требует введения распределения Дирихле и априорных вероятностей. Здесь достаточно сказать, что столбцы матриц Φ и Θ либо приближаются к некоторому заданному распределению, либо удаляются от него, причем в одном из столбцов может происходить сглаживание, а в другом — разреживание. Более того, так как априорные распределения Дирихле больше не используется, можно снять ограничения $\beta_w > 0, \alpha_t > 0$. Это позволяет сильнее разреживать матрицы Φ и Θ .

7.3.5. Онлайновый ЕМ-алгоритм

В общем случае ЕМ-алгоритм представляет собой систему уравнений, которая решается методом простых итераций, то есть по вспомогательным переменным $p_{tdw} = p(t|d, w)$ будут рассчитываться основные параметры модели ϕ_{wt} , θ_{td} , а затем наоборот — по ϕ_{wt} и θ_{td} вычисляются значения p_{tdw} .

Можно, ничего не меняя в формулах, организовать вычислительный процесс метода простых итераций таким образом, чтобы он шел максимально быстро, причем именно на больших коллекциях текстовых документов. Дело в том, что матрицы Φ и Θ находятся в неравноправном положении: каждый столбец матрицы Φ относится ко всей коллекции целиком, а каждый столбец матрицы Θ — только к одному документу.

Поэтому, если процесс будет устроен таким образом, что матрицы Φ и Θ будут обновляться только после просмотра всей коллекции, матрица Φ будет обновляться слишком редко и весь процесс будет работать крайне медленно.

Другой подход заключается в обновлении матрицы Φ после просмотра очередного документа и простройки для него тематической модели. Можно также обрабатывать по несколько документов (по порциям, пакетам документов) и обновлять матрицу Φ после анализа очередного пакета документов. Оказалось, что такой способ организовать процесс является самым быстрым, и позволяет создавать онлайновые алгоритмы

Ввод: коллекция D , число тем $|T|$, параметры $i_{\max}, j_{\max}, \gamma$;

Вывод: матрицы терминов тем Θ и тем документов Φ ;

для всех $i = 1, \dots, i_{\max}$ (итерации по коллекции)

для всех документов $d \in D$

для всех $j = 1, \dots, j_{\max}$ (итерации по документу)

$$p_{tdw} := \text{norm}_{t \in T}(\phi_{wt}\theta_{td})$$

$$\theta_{td} := \text{norm}_{t \in T} \left(\sum_{w \in d} n_{dw} p_{tdw} + \theta_{td} \frac{\partial R}{\partial \theta_{td}} \right)$$

$$n_{wt} := \gamma n_{wt} + n_{dw} p_{tdw}$$

если пора обновить матрицу Φ **то**

$$\phi_{wt} := \text{norm}_{w \in W} \left(n_{wt} + \phi_{wt} \frac{\partial R}{\partial \phi_{wt}} \right)$$

Онлайновые алгоритмы появились не сразу, а примерно лет 5 назад, и стали эффективным инструментом анализа больших коллекций текстов. Оказалось, что матрица Φ может сходиться завершения просмотра всей коллекции: после просмотра нескольких первых десятков тысяч документов матрица Φ получается уже более-менее устоявшейся. После этого ЕМ-алгоритм можно использовать, чтобы тематизировать остальные документы.

7.4. Регуляризация тематических моделей

7.4.1. Аддитивная регуляризация тематических моделей (ARTM)

Как было показано ранее, при решении задача тематического моделирования с помощью ЕМ-алгоритма существует свобода выбора дополнительных критериев, которые называются регуляризаторами. Регуляризаторы бывают двух типов: для учёта дополнительных данных и для получения решения Φ , Θ с заданными свойствами.

На самом деле, запас неединственности решения основной задачи матричного разложения настолько большой, что можно на модель одновременно наложить несколько таких ограничений. Такой подход называется аддитивной регуляризацией и заключается в максимизации логарифма правдоподобия с k регуляризаторами R_i :

$$\sum_{d,w} n_{dw} \ln \sum_t \phi_{wt} \theta_{td} + \sum_{i=1}^k \tau_i R_i(\Phi, \Theta) \rightarrow \max_{\Phi, \Theta}$$

где τ_i — коэффициенты регуляризации.

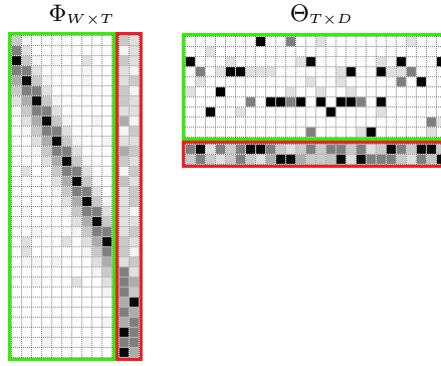


Рис. 7.3: Матрицы Φ и Θ

Такой подход дает возможность наложить сразу несколько условий, но также появляется проблема нахождения коэффициентов регуляризации. На данный момент, в основном, регуляризаторы добавляются по одному и у каждого регуляризатора оптимизируя этот коэффициент в ходе нескольких пробных запусков модели.

7.4.2. Разделение тем на предметные и фоновые

Продемонстрировать, как используя несколько регуляризаторов наделить модель нужными свойствами, можно на следующем примере. Наличие слов общей лексики в теме приводит к плохой интерпретируемости данной темы. Поэтому хотелось бы такие общеупотребительные слова выделить в отдельные темы, так называемые фоновые темы. Все остальные темы называются, соответственно, предметными, так как они описывают предметные области текстовой коллекции.

Предметные темы должны быть достаточно сильно разреженными, чтобы у каждой такой темы существовало свое лексическое ядро, существенно отличающее эту тему от остальных. Другими словами, требуется не только разреженность тем, но и их декоррелированность.

Эти требования можно выразить с помощью регуляризаторов. Пусть S — множество предметных тем, а B — множество фоновых. Поскольку для предметных тем ($t \in S$) матрицы $p(w|t)$ и $p(t|d)$ должны быть разреженными и существенно различными, а для фоновых ($t \in B$) — существенно отличными от нуля (больше половины слов в каждом документе — фоновые), имеет смысл применить регуляризатор, рассмотренный ранее в методе латентного размещения Дирихле. Единственное отличие состоит в том, что тогда он применялся для всего словаря, а в данном случае регуляризатор сглаживания необходимо применить только к фоновым темам:

$$R(\Phi, \Theta) = \beta_0 \sum_{t \in B} \sum_{w \in W} \beta_w \ln \phi_{wt} + \alpha_0 \sum_{d \in D} \sum_{t \in B} \alpha_t \ln \theta_{td} \rightarrow \max,$$

где β_0, α_0 — коэффициенты регуляризации. В этом случае распределения ϕ_{wt} будут близки к заданному распределению β_w , а распределения θ_{td} — к заданному распределению α_t . Распределения β_w и α_t вычисляются заранее. Например, в качестве β_w можно использовать распределение слов в используемом языке.

По аналогии можно построить разреживающий регуляризатор для предметных тем:

$$R(\Phi, \Theta) = -\beta_0 \sum_{t \in S} \sum_{w \in W} \beta_w \ln \phi_{wt} - \alpha_0 \sum_{d \in D} \sum_{t \in S} \alpha_t \ln \theta_{td} \rightarrow \max.$$

где β_0, α_0 — коэффициенты регуляризации. В этом случае распределения ϕ_{wt} и θ_{td} будут как можно далече от заданных распределений β_w и α_t . Определением параметров β_w и α_t занимается специалист, который занимается построением тематической модели. Часто в качестве β_w также используют распределение слов в используемом языке, а в качестве α_t — равномерное распределение.

7.4.3. Регуляризатор частичного обучения (semi-supervised learning)

Интересное обобщение этих двух регуляризаторов — сглаживающего и разреживающего — возникает в том случае, если векторы β_{wt} и α_{td} могут быть свои для каждого столбца:

$$R(\Phi, \Theta) = \beta_0 \sum_{t \in T} \sum_{w \in W} \beta_{wt} \ln \phi_{wt} + \alpha_0 \sum_{d \in D} \sum_{t \in T} \alpha_{td} \ln \theta_{td} \rightarrow \max,$$

Казалось бы, такое большое количество параметров дает лишнюю свободу. Но цель введения такого регуляризатора проста и может быть продемонстрирована с помощью следующего рассмотрения.

Для каждой темы можно задать свои подмножества типичных для этой темы слов и важных документов на эту тему. Такую информацию могут задавать эксперты. В таком случае говорят о semi-supervised learning, то есть о внесении частичной обучающей информации. Например, если в результате построения модели в ней остались некоторые ошибки (ключевое слово не попало в тему и так далее), то правки можно внести вручную, используя предложенный регуляризатор и следующие значения параметров:

- $\beta_{wt} = [w \in W_t]$ — белый список W_t терминов темы t
- $\alpha_{td} = [d \in D_t]$ — белый список D_t документов темы t
- $\beta_{wt} = -[w \in W_t]$ — чёрный список W_t терминов темы t
- $\alpha_{td} = -[d \in D_t]$ — чёрный список D_t документов темы t

7.4.4. Регуляризатор декоррелирования тем

Еще одно требование, которое уже было озвучено, состоит в том, чтобы в каждой теме выделялось свое лексическое ядро (множество терминов, отличающее её от других тем), то есть чтобы темы, как столбцы матрицы Φ , как можно меньше коррелировали друг с другом. Для этого вводится следующий ковариационный регуляризатор:

$$R(\Phi) = -\frac{\tau}{2} \sum_{t \in T} \sum_{s \in T \setminus t} \sum_{w \in W} \phi_{wt} \phi_{ws} \rightarrow \max,$$

где τ — коэффициент регуляризации. Такой регуляризатор добавляет условие, чтобы попарно все столбцы матрицы Φ были далеки друг от друга.

Такой регуляризатор легко можно продифференцировать по ϕ и, если подставить в формулу для M-шага, вычисления получаются несложными. То же самое касается и других рассмотренных в данном разделе регуляризаторов: модификации формул M-шага включают добавление новой константы или легко вычислимой величины.

7.4.5. Регуляризатор для отбора тем

Пусть $p(t)$ — распределение тем в коллекции документов:

$$p(t) = \sum_d p(d) \theta_{td}.$$

Еще одним полезным на практике регуляризатором, который основан на представлениях о дивергенции Кульбака-Лейблера, является регуляризатор разреживающий распределение $p(t)$.

$$R(\Theta) = -\tau \sum_{t \in S} \ln \sum_{d \in D} p(d) \theta_{td} \rightarrow \max.$$

где τ — коэффициент регуляризации. Действительно, максимизация KL-дивергенции между $p(t)$ и равномерным распределением, фактически, есть требование, чтобы как можно больше вероятностей в $p(t)$ приняли нулевые значения.

Этот регуляризатор можно использовать для отбора тем: после введения регуляризации вероятности для наиболее незначительных тем обнулятся, то есть такая ненужная тема будет удалена из коллекции. Это можно использовать, в том числе, для определения полного количества тем: необходимо сначала взять число тем избыточным а затем, постепенно с помощью регуляризатора удалять незначительные темы.

Интересным побочным эффектом этого регуляризатора оказалось удаление линейно-зависимых, расцепленных тем. Такие избыточные темы получаются при разделении слов одной темы в две. Этот эффект оказался нетривиальным и до сих пор не имеет теоретического обоснования, хотя хорошо работает на практике.

7.5. Мультимодальные тематические модели

7.5.1. Понятие модальности

На практике часто встречаются коллекции документов, которые включают в себя метаинформацию, связывающую каждый документ с элементами (токенами) каких-то конечных множеств (не обязательно слов). Эти конечные множества называются модальностями.

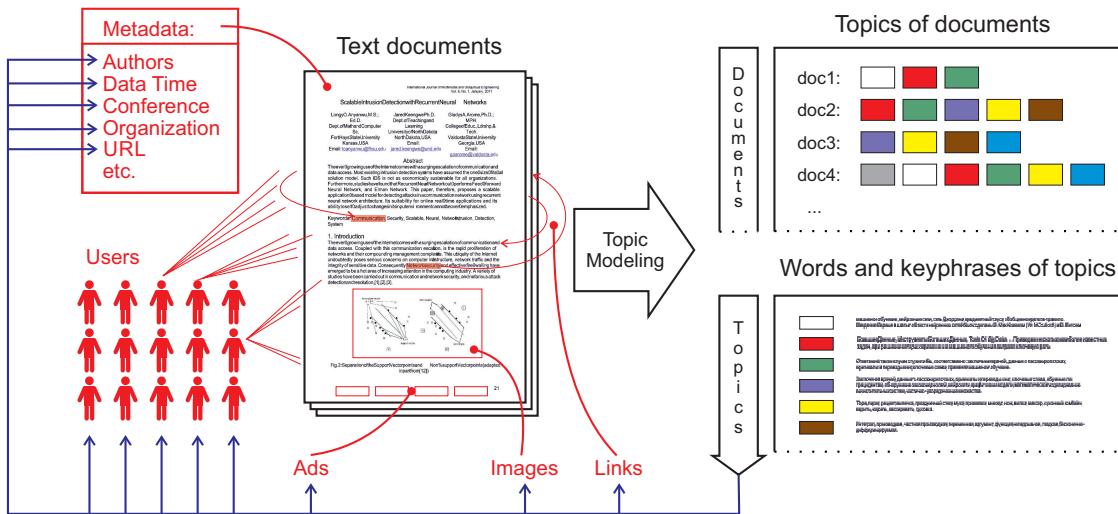


Рис. 7.4: Мультимодальная тематическая модель описывает появление элементов разных модальностей

Примеры модальностей:

- Авторы, моменты времени и так далее:** в этом случае каждому документу приписывается соответственно метка автора, временная метка и так далее.
- Элементы изображений,** содержащихся в документе. Изображение в таком случае можно мыслить как мини-документ, состоящий из псевдослов — элементов изображений.
- Множество ссылок на другие документы**, в том числе гиперссылки в сети Интернет и цитирование других статей в научных трудах.
- Множество рекламных баннеров, которые появились на данной странице**, а также **множество пользователей, которые кликнули на данные баннеры**, это два примера возможных модальностей.
- Множество пользователей, сделавших определенное действие с документом (скачал, лайкнул, поставил оценку и так далее).** После того, как операция выполнена, в системе остается запись о том, что данный пользователь сделал конкретную операцию. И поэтому можно считать, что в документ также включена и эта информация.

Чтобы иметь возможность пользоваться данными типами информации, необходимо строить тематические модели, которые описывают появление элементов разных модальностей в документе по известной тематике. Другими словами, благодаря тому, что документ относится к какой-либо теме, в нем появляются определенные слова из этой темы, на картинках изображены элементы, которые характерны для этой темы, а также его читают пользователи, которым эта тема интересна, и так далее.

7.5.2. Мультимодальная ARTM

Тематическая модель описывает появление элементов всех модальностей исходя из единого тематического профиля всего документа. Каждая модальность $m \in M$ описывается своим словарём токенов W^m , каждая тема имеет своё распределение $p(w|t)$ для каждой модальности $w \in W^m$.

Словарь всех модальностей — это объединение непересекающихся словарей отдельных модальностей:

$$W = W^1 \sqcup \cdots \sqcup W^M.$$

Можно построить вероятностную модель, как это делалось ранее, расписывая принцип максимума логарифма правдоподобия с регуляризацией (исходя из требуемых от модели свойств) отдельно для каждой модальности:

$$\sum_{m \in M} \tau_m \sum_{d \in D} \sum_{w \in W^m} n_{dw} \ln \sum_t \phi_{wt} \theta_{td} + R(\Phi, \Theta) \rightarrow \max_{\Phi, \Theta}$$

где τ_m — веса модальностей.

Такая постановка задачи оказывается не намного сложнее рассмотренной ранее и для нее легко выписывается модифицированный ЕМ-алгоритм, представляющий собой метод простой итерации для системы уравнений относительно параметров модели ϕ_{wt} , θ_{td} и вспомогательных переменных $p_{tdw} = p(t|d, w)$:

$$\begin{aligned} \text{E-шаг: } & p_{tdw} = \underset{t \in T}{\text{norm}} (\phi_{wt} \theta_{td}) \\ \text{M-шаг: } & \begin{cases} \phi_{wt} = \underset{w \in W^m}{\text{norm}} \left(\sum_{d \in D} \tau_{m(w)} n_{dw} p_{tdw} + \phi_{wt} \frac{\partial R}{\partial \phi_{wt}} \right) \\ \theta_{td} = \underset{t \in T}{\text{norm}} \left(\sum_{w \in d} \tau_{m(w)} n_{dw} p_{tdw} + \theta_{td} \frac{\partial R}{\partial \theta_{td}} \right) \end{cases} \end{aligned}$$

Всего модификации было сделано две: во-первых, частота слова в документе n_{dw} домножается на вес $\tau_{m(w)}$ соответствующей модальности $m(w)$ этого слова w . Во-вторых, при вычислении матрицы Φ она нормируется для каждой модальности по отдельности. Такое незначительное расширение математической модели позволяет решать разнообразные прикладные задачи с большим числом модальностей.

Пусть дана параллельная коллекция документов, то есть коллекция из оригинальных текстов и переводов этих текстов на другие языки. В этом случае различными модальностями будут различные языки. Если построить тематическую модель, учитывая все доступные модальности, появляется возможность делать кросс-язычный поиск, то есть получать результат на другом языке, нежели чем на котором был сделан запрос. Например, по тексту русскоязычному научной статьи можно будет в большой коллекции англоязычных текстов еще что-нибудь с похожей тематикой.

В качестве второй модальности можно использовать биграммы, то есть выделенные в тексте словосочетания. В таком случае получаются хорошо интерпретируемые модели, примеры которых были показаны в самом начале.

Урок 8

Тематическое моделирование-2

Оценки качества делятся на две большие категории: внутренние критерии качества и внешние критерии качества. Внутренние критерии — такие критерии, которые позволяют оценить качество модели по матрицам Φ и Θ , которые модель дала на выходе. Внешние критерии измеряют качество модели, глядя на то, как она решает ту конечную прикладную задачу, ради которой она и была создана.

8.1. Внутренние критерии качества тематических моделей

8.1.1. Перплексия (Perplexity)

Перплексия — известная в вычислительной лингвистике мера качества модели языка. В данном случае моделью языка является условное распределение слов в документах.

Перплексия коллекции D для языковой модели $p(w|d)$:

$$P(D) = \exp\left(-\frac{1}{n} \sum_{d \in D} \sum_{w \in d} n_{dw} \ln p(w|d)\right), \quad n = \sum_{d \in D} \sum_{w \in d} n_{dw}.$$

Эта мера качества тесно связана с правдоподобием. По сути дела это экспонента от усредненного по всем словам всех документов значения логарифма правдоподобия.

Значение перплексии можно интерпретировать следующим образом. Если подставить в качестве распределения слов в документах равномерное распределение:

$$p(w|d) = \frac{1}{|W|},$$

получится, что перплексия равна мощности словаря:

$$P = |W|.$$

Можно сказать, что это мера неопределенности или различности слов в тексте. Если распределение слов неравномерно, то перплексия уменьшается по сравнению с тем значением, которое дает равномерное распределение. Еще можно сказать, что перплексия — коэффициент ветвления текста, то есть количество ожидаемых в среднем различных слов после каждого слова в документе.

8.1.2. Hold-out perplexity

Перплексия может быть вычислена по самой коллекции, по которой построена тематическая модель, но тогда существует риск переобучения. Чтобы избежать этого и получить несмещенную оценку, вычисляют перплексию тестовой (отложенной) коллекции D' (hold-out perplexity):

$$P(D') = \exp\left(-\frac{1}{n''} \sum_{d \in D'} \sum_{w \in d''} n_{dw} \ln p(w|d)\right), \quad n'' = \sum_{d \in D'} \sum_{w \in d''} n_{dw}.$$

В отличие от предыдущего случая, параметры ϕ_{wt} оцениваются по обучающей коллекции D . Для каждого документа $d \in D$ строится случайное разбиение $d = d' \sqcup d''$ на две половины равной длины, причем параметры θ_{td} оцениваются по d' , а перплексия вычисляется по d'' .

Эксперименты на больших коллекциях показывают, что различие между перплексией на обучающей и на тестовой выборках, как правило, не существенно для цели сравнения разных моделей. Поэтому рекомендуется на очень больших выборках не считать hold-out perplexity, а считать обычную перплексию по основным данным.

8.1.3. Меры интерпретируемости тем

Перплексия ничего не говорит об интерпретируемости тем, а только говорит о том, насколько хорошо построилось матричное разложение. То есть ничего не говорит о том, насколько построенная модель будет полезна для конечных приложений. Поэтому были придуманы меры качества, которые измеряют, насколько темы хороши и понятны. Такую оценку можно сделать только при помощи экспертов.

Например, экспертам предлагается рассмотреть темы как последовательности слов, упорядоченные по вероятности встретить слово в данной теме, и оценить интерпретируемость темы по некоторой шкале (2 или 5-балльной). Обычно экспертам дают такую инструкцию: если эту тему можно кратко озаглавить или по этим словам можно сформулировать поисковый запрос и получить релевантную поисковую выдачу, то такую тему следует считать интерпретируемой. Такие оценки, естественно, являются субъективными, поэтому каждую тему оценивают несколько экспертов, чтобы понять, насколько непротиворечивы их оценки.

Другой метод, так называемый метод интрузий (intrusion), заключается в том, что в список топовых (имеющих большую вероятность) слов темы внедряется лишнее слово, которое заведомо этой теме не принадлежит. Экспертам предлагается определить, какое слово из списка лишнее, и измеряется доля ошибок при его определении. Такой способ оценки интерпретируемости существенно проще для экспертов, а, следовательно, эксперт может оценить больше тем в единицу времени.

8.1.4. Когерентность (Согласованность)

В ходе экспериментов было выявлено, что экспертные оценки хорошо коррелируют с такой мерой качества как когерентность, которая может быть вычислена полностью автоматически без участия человека. Когерентность темы — мера, которая показывает, насколько слова, встречающиеся рядом в текстах, оказываются в топах одних и тех же тем.

Когерентность темы t — средняя поточечная взаимная информация топ-слов темы (pointwise mutual information, PMI):

$$PMI_t = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i}^k PMI(w_i, w_j),$$

где w_i — i -ый термин в порядке убывания ϕ_{wk} , $k = 10$.

Поточечная взаимная информация

$$PMI(u, v) = \ln \frac{|D| N_{uv}}{N_u N_v},$$

где N_{uv} — число документов, в которых термины u и v хотя бы один раз встречаются рядом (в окне 10 слов), N_u — число документов, в которых термин u встретился хотя бы один раз. Чем выше величина поточечной взаимной информации, тем выше неслучайность того, что два слова стоят рядом.

8.2. Внешние критерии качества тематических моделей

Если внутренние критерии качества оценивают матрицы Φ и Θ построенной тематической модели, то внешние критерии качества основаны на измерении полезности построенной тематической модели для исходной прикладной задачи.

8.2.1. Примеры задач классификации и категоризации документов

Задача классификации документов является задачей машинного обучения с учителем. Типичное применение задачи классификации:

- Определение жанра документа (художественный, научный, учебный, рекламный и так далее). Это может быть необходимо для анализа потока документов, полученных поисковыми роботами из интернета.
- Определение тематики сообщения в новостном потоке (политика, экономика, наука, здоровье, спорт и так далее)
- Определение категории (рубрики), к которой относится документ (например, Наука/Физика/Большой Адронный Коллайдер или Спорт/Футбол/Чемпионат мира по футболу). Основная особенность — наличие иерархической структуры.

Обучающая информация, которая используется для построения модели классификации, является результатом ручной классификации документов экспертами. То есть имеется обучающая выборка, в которой эксперты указали, к какой категории/жанру/тематике принадлежит тот или иной документ. Эта информация может быть использована для оценивания качества тематической модели.

8.2.2. Задача классификации

Безусловно, можно применить механизм мультимодальных моделей и использовать две модальности: термины $w \in W$ и классы $c \in C$.

На этапе обучения на вход подается коллекция документов, то есть матрицы (n_{dw}) (словарный состав документов) и (n_{dc}) (принадлежность документов классам, причем документ может быть отнесен сразу к нескольким классам), а на выходе — модель коллекции $p(w|t)$, $p(c|t)$. По полученной тематической модели можно будет классифицировать документы, для которых никаких классов не известно.

На этапе классификации на вход подается документ $d : (n_{dw})$ и известна модель классификации, построенная на этапе обучения. В первую очередь определяется тематика данного документа $p(t|d)$, а затем по модели, построенной на этапе обучения, определяются условные вероятности каждого класса для данного документа:

$$p(c|d) = \sum_{t \in T} p(c|t)p(t|d).$$

На основании этого распределения определяется принадлежность документа классу или классам.

8.2.3. Критерии качества классификации или категоризации

Оценивать качество работы алгоритма, возвращающего условные вероятности классов для заданного документа, можно через:

- Число ошибок классификации (считается, что документ относится к классу, если соответствующая условная вероятность больше некоторого порогового значения). Такой способ является не очень хорошим, поскольку классы могут иметь разную важность и разную мощность.
- Критерий чувствительности и специфичности. Этот критерий подходит для работы с несбалансированными выборками.
- Критерий площади под кривой чувствительность–специфичность (AUC). Такой критерий не зависит от выбранного порога для классификации.
- Критерий точности и полноты.
- Критерий площади под кривой точность–полнота (AUC-PR).

8.2.4. Точность и полнота многоклассовой классификации

Пусть c — некоторый класс, TP_c — количество документов, которые были верно отнесены алгоритмом к классу c , FP_c — ошибочно отнесенных к классу c , FN_c — ошибочно неотнесенных к классу c . Точность (precision) — доля релевантных документов среди всех отнесенных алгоритмом к классу c , полнота (recall) — доля документов, отнесенных алгоритмом к классу c , среди всех релевантных. Точность P_c и полнота R_c для класса c выражаются так:

$$P_c = \frac{TP_c}{TP_c + FP_c}, \quad R_c = \frac{TP_c}{TP_c + FN_c}.$$

Если классов много, существуют две стратегии:

- Точность и полнота с микроусреднением:

$$P = \frac{\sum_c TP_c}{\sum_c (TP_c + FP_c)}, \quad R = \frac{\sum_c TP_c}{\sum_c (TP_c + FN_c)}.$$

- Точность и полнота с макроусреднением:

$$P = \frac{1}{|C|} \sum_c P_c, \quad R = \frac{1}{|C|} \sum_c R_c.$$

Принципиальная разница состоит в том, что макроусреднение более чувствительно к качеству классификации маломощных классов. При микроусреднении качество работы на маломощных классах слабо влияет на полное качество.

8.2.5. Тематический поиск

Тематический поиск — еще одна задача, которая часто используется для оценивания качества тематических моделей.

На этапе обучения на вход подается коллекция документов, то есть матрица слов в документах (n_{dw}), а на выходе строится тематическая модель коллекции $p(t|d)$. На этапе поиска по поисковому запросу (документу произвольной длины) $q : (n_{qw})$, используя построенную на этапе обучения модель коллекции, строится модель запроса $p(t|q)$. Также выводятся документы коллекции, ранжированные по близости к запросу.

Для этого необходимо уметь сравнивать условные распределения тем у запроса $p(t|q)$ и у документа d из коллекции $p(t|d)$.

8.2.6. Оценивание близости запроса и документа

Для сравнения вероятностных распределений тем в информационном поиске принято использовать несколько мер (которые показали себя наиболее эффективными в экспериментах на реальных задачах):

- Косинусная мера (чем больше, тем ближе):

$$\cos(q, d) = \frac{\sum_t p(t|q)p(t|d)}{\sqrt{\sum_t p(t|q)^2} \sqrt{\sum_t p(t|d)^2}}$$

- Расстояние Хеллингера (чем меньше, тем ближе):

$$H^2(q, d) = \frac{1}{2} \sum_t \left(\sqrt{p(t|q)} - \sqrt{p(t|d)} \right)^2.$$

- KL-дивергенция (чем меньше, тем ближе):

$$KL(q, d) = \sum_t p(t|d) \log \frac{p(t|q)}{p(t|d)}$$

Следует отметить, что KL-дивергенция является несимметричной мерой и имеет смысл того, насколько запрос подходит под данный документ. KL-дивергенцию имеет смысл использовать в тех случаях, когда запрос короче документов.

Еще один момент заключается в том, что на косинусную меру и расстояние Хеллингера может влиять «обрезание хвостов распределений». Этот прием используется, чтобы отбросить редкие темы, которые обычно мало интересуют. Это позволяет иногда повысить качество поиска и его скорость.

8.2.7. Критерии качества тематического поиска

Качество тематической модели по результату тематического поиска можно оценить по:

- точности первых k позиций поисковой выдачи (требуется экспертная оценка релевантности),
- средней позиции некоторого документа в поисковой выдаче при поиске по его аннотации,
- средней точности по релевантным позициям (MAP = Mean Average Precision) при поиске фрагментов документа по другим фрагментам,
- средней позиции документа при поиске по его переводу на другой язык (при кросс-язычном поиске).

8.3. Визуализация тематических моделей

Тематические модели, как правило, создаются для упрощения понимания и обеспечения навигации по большим текстовым коллекциям, поэтому важной задачей является визуализация тематических моделей. В последние годы было создано достаточно много средств визуализации, многие из которых находятся в свободном доступе. Большинство из этих инструментов ориентированы на то, чтобы визуализировать текстовые коллекции через web-интерфейсы.

8.3.1. Система TMVE

Система Topic Model Visualization Engine является одним из канонических примеров тематического навигатора с web-интерфейсом.

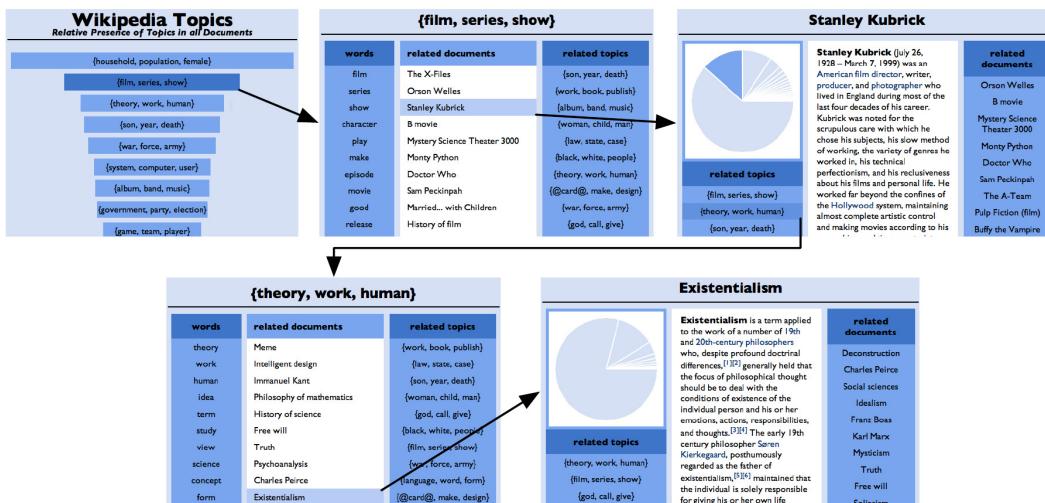


Рис. 8.1: «Wikipedia Topics», демонстрационный пример, представленный авторами TMVE.

На главной странице системы находится список тем, по каждой теме можно просмотреть документы и термины этой темы. Таким образом реализуется возможность навигации пользователя по коллекции.

8.3.2. Система TERMITE

Система TERMITE позволяет интерактивно визуализировать матрицу Φ и больше подходит для разработчиков тематической модели.

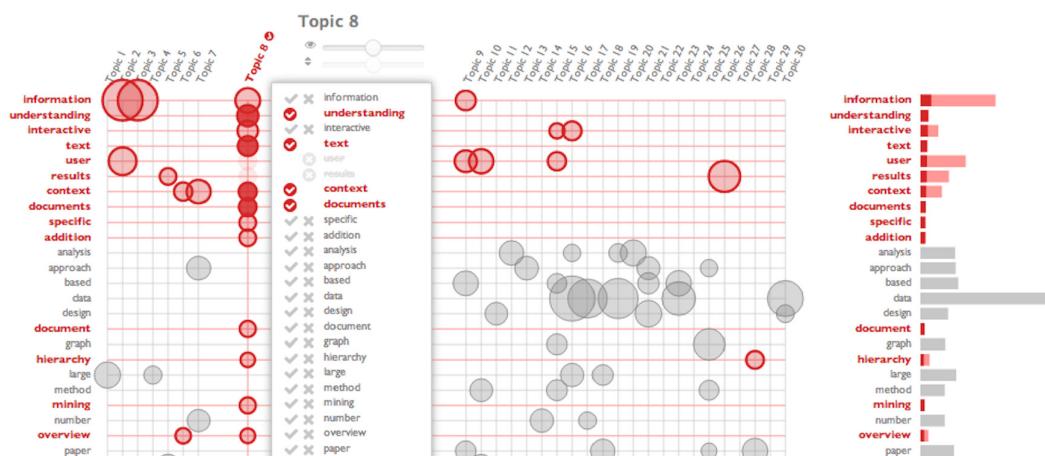


Рис. 8.2: Система TERMITE

8.3.3. Динамические модели, учитывающие время

Есть огромное количество средств визуализации для тематических моделей потоков новостей, научных статей или любых других коллекций, где каждому документу приписывается метка времени. Тогда строить тематические модели очень удобно, визуализируя их в виде графиков, на которых отображено, как развивались темы во времени, в какие моменты темы набирали популярность.

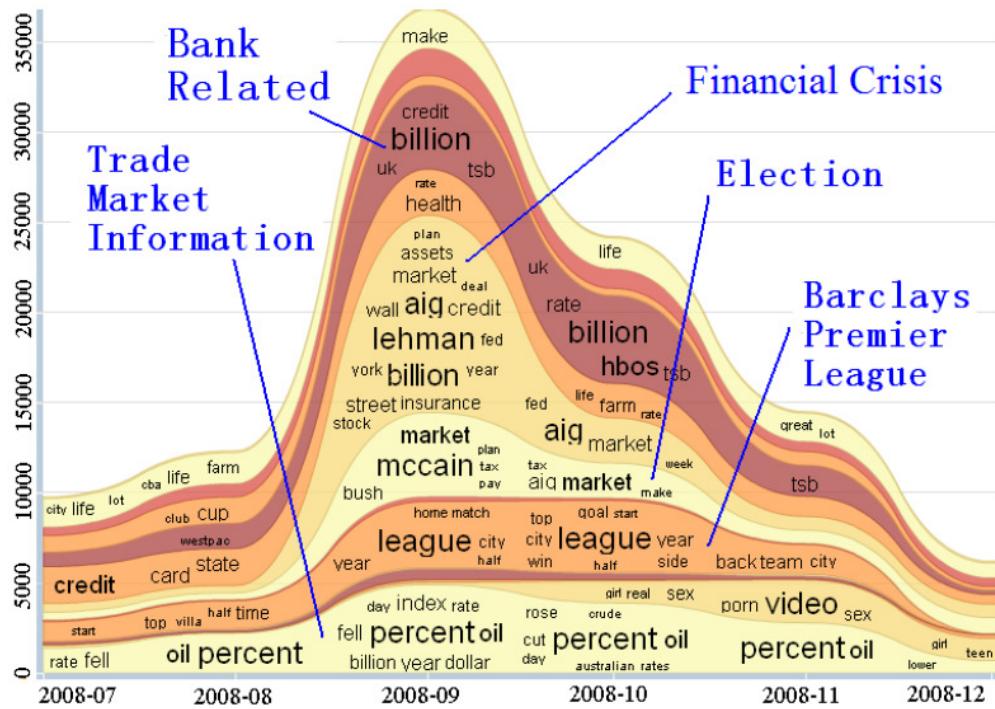


Рис. 8.3: На этом графике видны темы, которые возникли в связи с финансовым кризисом 2008.

На таких графиках можно изучать предвестники, последствия и связанные темы.

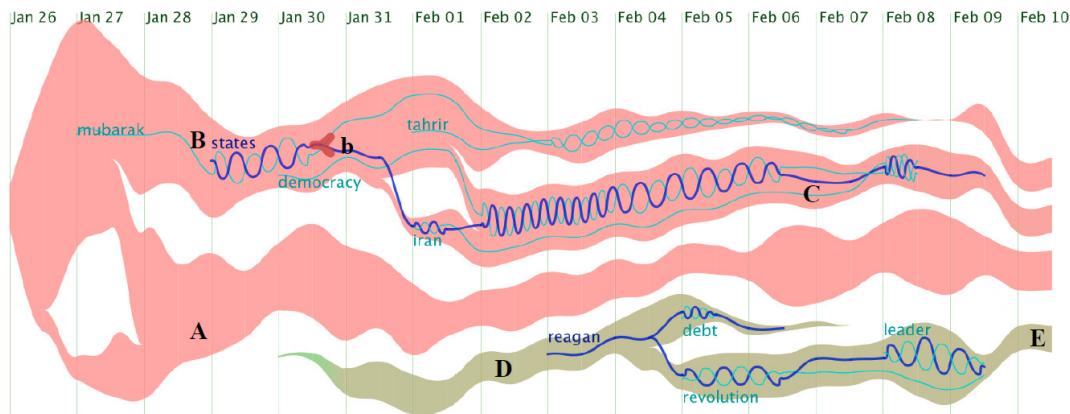
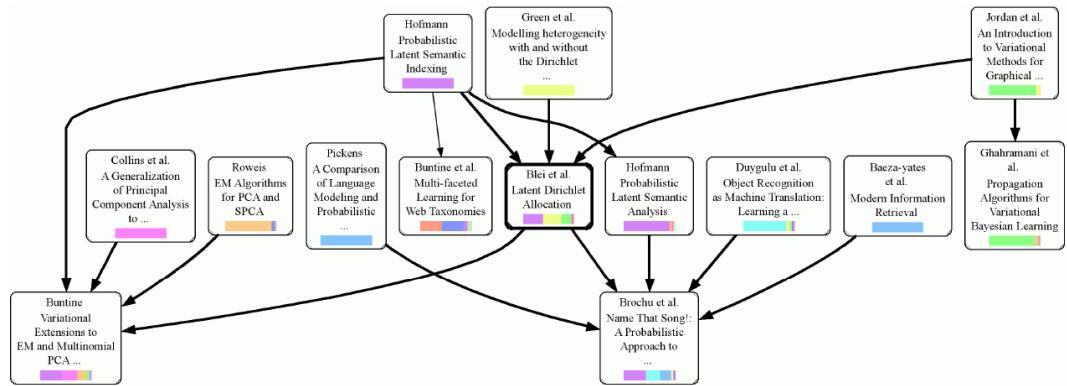


Рис. 8.4: Визуализация «река тем»: изображены моменты зарождения тем, исчезновения. Волнами изображаются траектории отдельных слов, причем часто колеблющаяся волна значит, что слово использовалось часто.

8.3.4. Динамические модели, учитывающие ссылки

Если тематическая модель учитывает не только слова, но также связи между документами, например связи цитирования между научными статьями, то можно ставить очень интересные задачи.

Например, можно попытаться ответить на вопрос, какие предшествующие работы действительно существенно повлияли на данную статью. В статье часто десятки ссылок, многие из которых чисто формальные, или дань вежливости, или же незначительные моменты, которые для данной статьи не важны. Оказывается, что это можно сделать с помощью тематической модели: выявить тематику статьи и выбрать из списка литературы те статьи, которые тоже соответствуют этой тематике.



С другой стороны, использование ссылок и цитат позволяет уточнить саму тематическую модель. Для этого предполагается, что если статья ссылается на другую, то у них есть общая тематика, и это учитывается с помощью регуляризатора.

8.3.5. Выявление взаимосвязей между темами

Оказывается, что можно выявлять связи между темами. Это особенно хорошо получается на коллекциях научных текстов. Так, например, в статье про археологию скорее появится термин из геологии, чем из генетики. Выявление таких связей между отраслями знаний представляет отдельный прикладной интерес.

Если каждую тему изображать в виде вершины графа, а ребро проводить только в том случае, когда соответствующие две темы часто появлялись в документах одновременно, то получившаяся тематическая модель будет называться кореллированной тематической моделью.

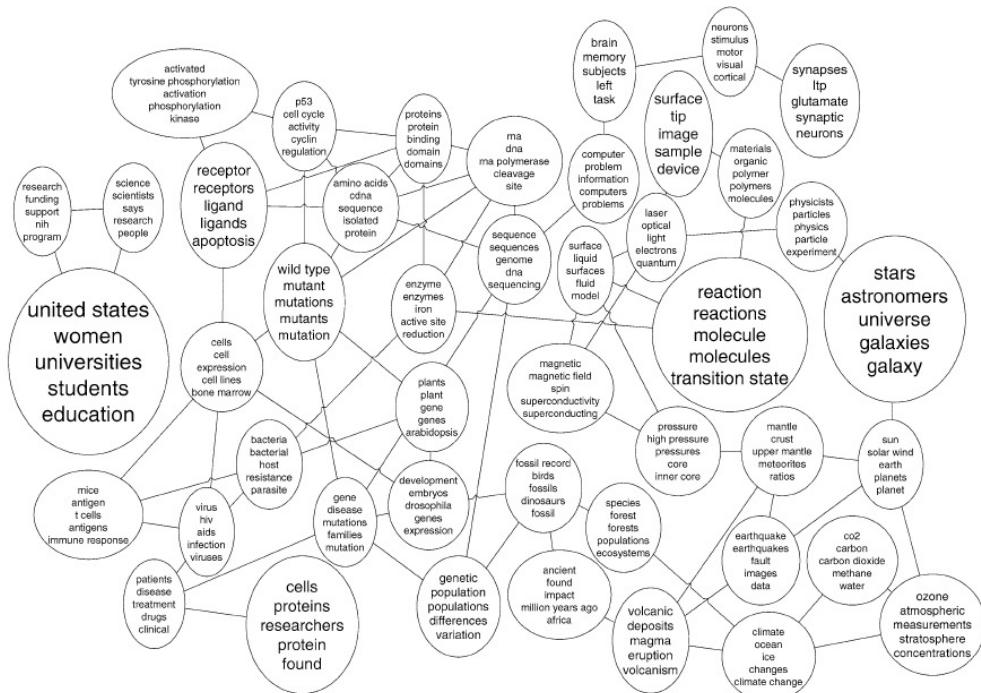


Рис. 8.5: Кореллированная тематическая модель, построенная на текстах из журнала Science.

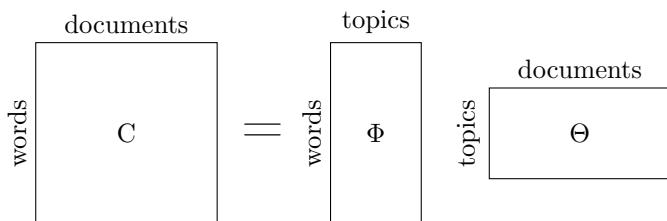
8.4. Тематические модели на практике

В этом разделе будет рассказано про практические аспекты применения тематических моделей.

8.4.1. Тематическая модель как матричное разложение

Тематические модели является еще одним видом матричного разложения, адаптированное для работу с текстом. Коллекция документов представляется в виде матрицы частот слов, причем эта матрица обычно сильно разрежена. Матричное разложение позволяет приблизить исходную матрицу в виде произведения двух матриц меньшего ранга. Причем, чем больше промежуточная размерность, тем лучше аппроксимация исходной матрицы. Разные виды матричного разложения отличаются ограничениями на матрицы и метрикой, с помощью которой оценивается мера схожести с исходной матрицей.

Тематические модели приближают исходную матрицу по псевдометрике, которая называется дивергенция Кульбака-Лейблера. Промежуточная размерность, то есть количество тем, обычно подбирается экспериментально, так как большее значение промежуточной размерности не всегда хорошо. Иногда количество тем диктуется поставленной задачей — это самый лучший вариант.



С помощью тематических моделей оказывается решенной одна из популярных проблем машинного обучения — интерпретируемость модели. Матрица Φ — матрица распределений слов в темах, а матрица Θ — матрица распределений тем в документах. По списку наиболее вероятных слов тем можно понять, о чём эта тема, и дать ей название. Тогда будет легко охарактеризовать тематический профиль документа.

8.4.2. Примеры задач тематического моделирования

Тематическое моделирование можно использовать для анализа огромного количества сообщений социальной сети. Например, коллекция документов — миллион сообщений из социальной сети Twitter. Если на этой коллекции построить тематическую модель, то можно будет смотреть не все твиты, а только на интересующую тему. Таким образом, тематическое моделирование позволяет сузить множество документов, которые необходимо просматривать. Использование других инструментов автоматического анализа текстов позволит автоматически построить навигатор по коллекции текстовых документов.

Тематические модели используются в рекомендательных системах, при медицинской диагностике и даже при распознавании изображений. На сайте [habrahabr.ru](#) доступна [статья](#), в которой разработчики рассказывали о применении тематического моделирования для задачи рекомендации web-страниц. В качестве матрицы частот слов использовалась матрица популярности страниц для пользователей, то есть в качестве документов выступали пользователи, а в качестве слов — отдельные страницы. После построения тематической модели оказалось, что статьи, которые попали в одну тему, действительно семантически похожи друг на друга.

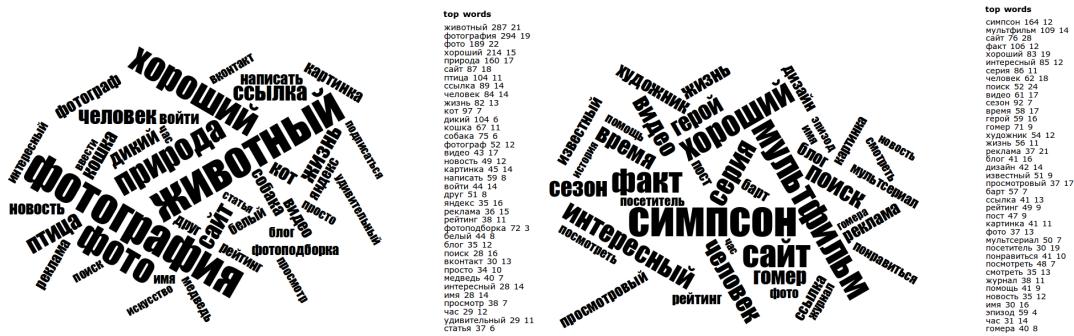
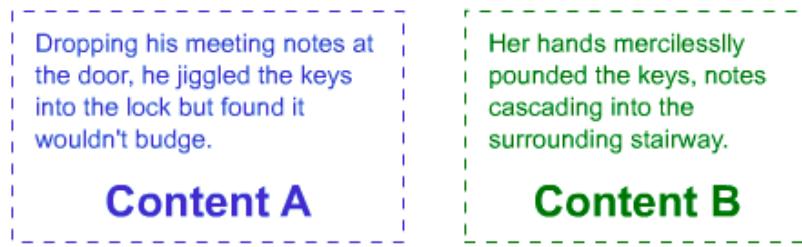


Рис. 8.6: Для статей из каждой темы были определены наиболее часто встречающиеся слова.

Еще одно применение тематических моделей — **использование в поисковых машинах**. В частности, они используются для выбора документов, которые потенциально могут быть релевантны запросу.

Search Query: Pianist



Solution: Topic Modeling

As humans reading both sentences, we can infer that **Content B** is obviously about the musical instrument - a piano - and the woman playing it. But a search engine armed with only the methods we described above will struggle since both sentences use the words "keys" and "notes," some of the only clues to the puzzle.

NOTE: We were excited to see that our LDA modeling tool correctly scored B higher than A :-)

Рис. 8.7: Тематические модели позволяют отличить релевантный документ от нерелевантного за счет построения тематического профиля.

8.4.3. Методы тематического моделирования

Одним из самых первых подходов к построению тематических моделей, Probabilistic Latent Semantic Analysis (PLSA), был предложен в 1999 году. В этом методе ставилась задача матричного разложения с дополнительными условиями на нормировку столбцов матриц (столбцы матриц должны представлять собой вероятностное распределение), которая решалась методом максимального правдоподобия, стандартным методом в статистике.

В 2003 году эта же задача была рассмотрена в байесовской постановке, в которой вместо матриц строится вероятностное распределение над матрицами. Этот подход получил название Latent Dirichlet Allocation (LDA), или латентное размещение Дирихле. Стоит отметить, что LDA — самая изученная модель тематического моделирования.

Не так давно был разработан другой подход, Additive Regularization of Topic Models (ARTM), который заключался в регуляризации PLSA с целью получения лучших моделей. Для этого предполагается ввести дополнительные критерии как регуляризаторы в модель PLSA, за счет чего модель получается более гибкой и ее можно адаптировать к большему числу задач.

LDA	ARTM
Очень популярный	Молодой
Множество модификации для различных задач	Мощный аппарат регуляризаторов для модификации модели
Для каждого усложнения нужно искать реализацию	Одна реализация для разных задач
Нужно настраивать гиперпараметры	Нужно настраивать параметры регуляризации