

Chapter 1

Лекция 7. Обучение без
учителя. Кластеризация.
Снижение размерности данных.

Глинский А.В.

Цель занятия: В этом модуле мы познакомимся с такой темой, как обучение без учителя. После прохождения модуля ученики смогут понимать основные принципы подхода обучение без учителя, и применять базовые алгоритмы кластеризации и снижения размерности данных.

Содержание

1 Лекция 7. Обучение без учителя. Кластеризация. Снижение размерности данных.	1
1.1 Обучение без учителя	4
1.1.1 Кластеризация	5
1.1.2 Снижение размерности данных	9
1.1.3 Вопросы для самопроверки	11
1.1.4 Резюме по разделу	13
1.2 K-means	14
1.2.1 Визуальная демонстрация алгоритма	14
1.2.2 Подготовка данных для алгоритма	15
1.2.3 Процесс обучения	15
1.2.4 Оценка качества алгоритма	16
1.2.5 Применение алгоритма	17
1.2.6 Плюсы и минусы алгоритма	18
1.2.7 Реализация алгоритма в Python	19
1.2.8 Вопросы для самопроверки	19
1.2.9 Резюме по разделу	20
1.3 DBSCAN	21
1.3.1 Визуальная демонстрация алгоритма	21
1.3.2 Подготовка данных для алгоритма	22
1.3.3 Процесс обучения	23
1.3.4 Оценка качества алгоритма	24
1.3.5 Применение алгоритма	24
1.3.6 Плюсы и минусы алгоритма	25
1.3.7 Реализация алгоритма в Python	26
1.3.8 Вопросы для самопроверки	27
1.3.9 Резюме по разделу	27
1.4 Агломеративная кластеризация	29
1.4.1 Визуальная демонстрация алгоритма	29
1.4.2 Подготовка данных для алгоритма	29
1.4.3 Процесс обучения	31
1.4.4 Оценка качества алгоритма	31
1.4.5 Применение алгоритма	32
1.4.6 Плюсы и минусы алгоритма	32

1.4.7	Реализация алгоритма в Python	33
1.4.8	Вопросы для самопроверки	34
1.4.9	Резюме по разделу	34
1.5	Метод главных компонент	35
1.5.1	Визуальная демонстрация алгоритма	38
1.5.2	Подготовка данных для алгоритма	42
1.5.3	Процесс обучения	42
1.5.4	Оценка качества алгоритма	43
1.5.5	Применение алгоритма	44
1.5.6	Плюсы и минусы алгоритма	45
1.5.7	Реализация алгоритма в Python	46
1.5.8	Вопросы для самопроверки	46
1.5.9	Резюме по разделу	47
1.6	t-SNE	48
1.6.1	Визуальная демонстрация алгоритма	51
1.6.2	Процесс обучения	54
1.6.3	Оценка качества алгоритма	55
1.6.4	Применение алгоритма	55
1.6.5	Плюсы и минусы алгоритма	56
1.6.6	Реализация алгоритма в Python	57
1.6.7	Вопросы для самопроверки	57
1.6.8	Резюме по разделу	58
1.7	Резюме по модулю	59

1.1 Обучение без учителя

Обучение без учителя (англ. Unsupervised learning) - это подраздел машинного обучения, который занимается извлечением информации и паттернов из неструктурированных (неразмеченных) данных. В отличие от обучения с учителем, где для модели предоставляются размеченные данные с правильными ответами, при обучении без учителя данные не имеют меток или правильных ответов. Основная цель обучения без учителя - найти скрытую структуру в данных или сгенерировать полезные представления данных без явной разметки. В результате обучения без учителя модель выявляет внутренние закономерности, группирует или кластеризует данные, находит скрытые факторы или создает низкоразмерные представления данных.

Примеры алгоритмов обучения без учителя включают в себя методы кластеризации (например, k-средних, DBSCAN), методы снижения размерности (например, PCA, t-SNE), алгоритмы генеративных моделей (например, автокодировщики, генеративные состязательные сети) и многие другие.

Обучение без учителя широко применяется в анализе данных, обнаружении аномалий, сжатии данных, генерации контента и во многих других задачах, где необходимо извлекать полезную информацию из неразмеченных данных. Например, на рисунке показано совместное применение алгоритмов PCA (снижение размерности) + k-средних (кластеризация) для визуализации датасета MNIST (Рисунок 1.1). В итоге алгоритм находит центроиды кластеров¹ и закрашивает области, наиболее близкие к каждому центроиду, определенным цветом.

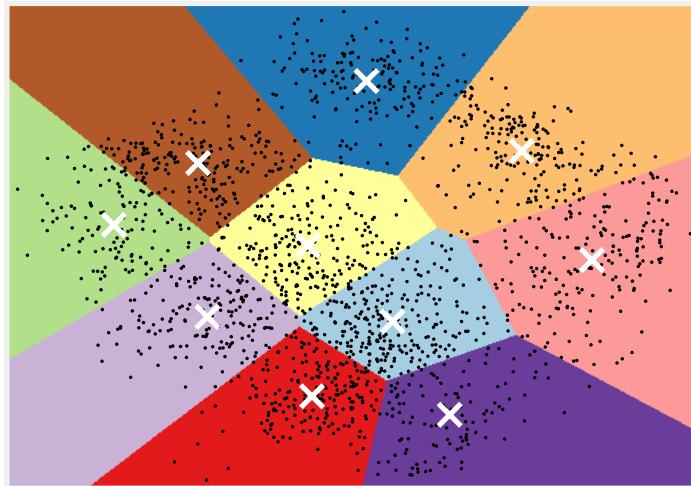


Рисунок 1.1: Визуализация датасета MNIST после применения алгоритмов снижения размерности и кластеризации

Давайте подробнее поговорим о двух классических подразделах обучения без учителя - кластеризации и снижении размерности данных.

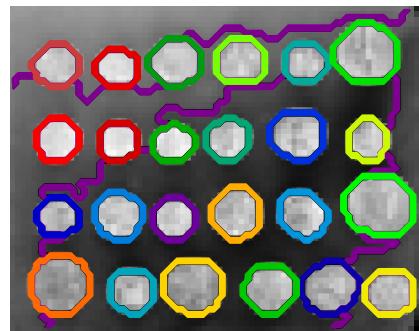
¹ Центроиды - это точки, представляющие центры каждого кластера, и они являются средними значениями точек внутри каждого кластера.

1.1.1 Кластеризация

Кластеризация в машинном обучении относится к задачам обучения без учителя и заключается в группировке объектов данных в кластеры на основе их схожести или близости друг к другу. Алгоритмы кластеризации стремятся найти внутреннюю структуру в данных, основываясь на их признаках или свойствах, без заранее известного количества кластеров или информации о принадлежности объектов к определенным кластерам. Например, на рисунке 1.2 алгоритм Agglomerative Clustering выделяет монеты на фотографии. Целью кластеризации является максимизация схожести объектов внутри кластеров и минимизация схожести между кластерами.



(1) Монеты на фотографии



(2) Кластеризация монет

Рисунок 1.2: Кластеризация с помощью Agglomerative Clustering

На рисунке 1.3 показан пример сжатия данных с помощью алгоритма K-means.



(1) Оригинальный рисунок с большим количеством цветов



(2) Обработанный рисунок с малым количеством цветов

Рисунок 1.3: Пример сжатия данных с помощью кластеризации

Распространенные примеры использования кластеризации:

- Поиск скрытых структур: Кластерный анализ помогает выявить скрытые структуры и группы в данных. Это может быть полезно для идентификации паттернов, тенденций

или сегментов, которые могут быть использованы для принятия более информированных решений.

- Сжатие данных: Путем группировки похожих объектов в кластеры можно сократить объем данных, сохраняя при этом основные характеристики и структуру. Это может упростить дальнейший анализ или обработку данных.
- Рекомендательные системы: Кластеризация может использоваться для создания групп пользователей или товаров с похожими характеристиками. Это может быть основой для разработки рекомендаций, чтобы предложить пользователям похожие товары или связать их с другими пользователями с общими интересами.
- Сегментация аудитории: Кластерный анализ может помочь в разделении аудитории на группы с общими характеристиками. Это может быть полезно для адаптации маркетинговых стратегий, персонализации коммуникации или создания целевых сообщений для каждой группы.

Метрики качества кластеризации

Кластеры представляют собой группы объектов, которые обладают схожими характеристиками или свойствами, в то время как объекты из разных кластеров сильно отличаются друг от друга. Для измерения расстояний вводится понятие центроида. Внутрикластерное расстояние и межкластерное расстояние - это две основные характеристики, которые учитываются при оценке качества кластеризации:

- Внутрикластерное расстояние (интракластерное расстояние) - это мера сходства между объектами внутри одного кластера. Чем меньше среднее внутрикластерное расстояние, тем более компактными и однородными являются кластеры (Формула 1.1).

$$\text{Среднее внутрикластерное расстояние} = \frac{1}{N} \sum_{i=1}^N \sum_{\substack{j=1 \\ j \neq i}}^N \text{расстояние}(x_i, x_j) \quad (1.1)$$

где N - общее количество объектов, x_i и x_j - объекты, а $\text{расстояние}(x_i, x_j)$ - функция, вычисляющая расстояние между объектами x_i и x_j . Обычно в качестве меры расстояния используют L2 норму. Эта формула вычисляет сумму всех парных расстояний между объектами внутри каждого кластера и затем усредняет это значение по всем кластерам. Низкое внутрикластерное расстояние указывает на то, что объекты внутри кластера находятся близко друг к другу, что является желательным свойством для качественной кластеризации. Определение низкого внутрикластерного расстояния зависит от контекста и специфики данных. Универсального значения, которое можно было бы считать низким для всех случаев, нет.

- Межкластерное расстояние - это мера различия между кластерами. Оно определяет, насколько различные кластеры удалены друг от друга. Чем больше межкластерное расстояние, тем более разделены и отделены друг от друга кластеры. Высокое межкластерное расстояние указывает на хорошую разделимость и различимость кластеров. Среднее межкластерное расстояние может быть представлено следующим образом (Формула 1.2):

$$\text{Среднее межкластерное расстояние} = \frac{1}{K(K-1)} \sum_{i=1}^K \sum_{j=1, j \neq i}^K \text{расстояние}(c_i, c_j) \quad (1.2)$$

где K - общее количество кластеров, c_i и c_j - центроиды (средние значения) кластеров, а $\text{расстояние}(c_i, c_j)$ - функция, вычисляющая расстояние между центроидами c_i и c_j . Обычно в качестве меры расстояния используют L2 норму. Эта формула вычисляет сумму всех парных расстояний между центроидами кластеров и затем усредняет это значение по всем парам кластеров.

Вот несколько основных метрик качества кластеризации, построенных с помощью учета расстояния:

- Коэффициент силуэта (англ. Silhouette Coefficient) (Формула 1.3):

$$SC = \frac{M - S}{\max(S, M)} \quad (1.3)$$

где M - среднее межкластерное расстояние, S - среднее внутрикластерное расстояние. Коэффициент силуэта принимает значения от -1 до 1, где ближе к 1 указывает на хорошую кластеризацию, а ближе к -1 - на неправильную кластеризацию. Формула называется «коэффициентом силуэта» (Silhouette Coefficient) потому, что она основана на понятии «силуэта» как визуального представления объекта или кластера.

Силуэт представляет собой меру компактности и разделенности объекта в контексте кластеризации. Визуально, силуэт объекта можно представить как его очертание, отражающее степень близости объекта к другим объектам внутри его кластера и удаленности от объектов в других кластерах. Она отражает, насколько хорошо объекты сгруппированы внутри своих кластеров, и насколько различаются между собой разные кластеры.

- Индекс Данна (англ. Dunn Index) (Формула 1.4):

$$D = \frac{\min_{1 \leq i \leq k, 1 \leq j \leq k, i \neq j} M(i, j)}{\max_{1 \leq l \leq k} d(l)} \quad (1.4)$$

где $M(i, j)$ - расстояние между кластерами C_i и C_j , k - общее количество кластеров, которые были образованы в результате кластеризации данных, а $d(l)$ - диаметр кластера C_l . Чем больше значение индекса Данна, тем лучше разделены кластеры. Это означает, что межкластерные расстояния максимизированы, а внутрикластерные расстояния минимизированы. Более четкое разделение кластеров приводит к более высокому значению индекса Данна. В отличие от коэффициента силуэта, индекс Данна не имеет фиксированного диапазона значений, и его интерпретация может быть более сложной. Обычно сравниваются несколько различных кластерных разбиений, и кластеризация с более высоким значением индекса Данна считается более оптимальной.

- Индекс Дэвиса-Боулдина (англ. Davies-Bouldin Index) (Формула 1.5):

$$DB = \frac{1}{k} \sum_{i=1}^k \max_{j \neq i} \left(\frac{S_i + S_j}{M_{ij}} \right) \quad (1.5)$$

где S_i - среднее расстояние между объектами внутри кластера C_i , M_{ij} - расстояние между центроидами кластеров C_i и C_j , k - общее количество кластеров, которые были образованы в результате кластеризации данных. Чем меньше значение индекса Дэвиса-Боулдина, тем лучше разделены кластеры. Это означает, что внутрикластерные схожести максимизированы, а межкластерные различия минимизированы. Более четкое разделение кластеров приводит к более низкому значению индекса Дэвиса-Боулдина.

Сложность кластеризации заключается в том, что на одной выборке может быть несколько различных вариантов разделения на кластеры, и не всегда ясно, какое разбиение является правильным. Формализация критериев для определения правильного разбиения на практике достаточно сложна, поэтому сама задача кластеризации не всегда имеет однозначное решение.

В отсутствии разметки и при неизвестном заранее количестве кластеров, на практике одной из лучших метрик является коэффициент силуэта. Он позволяет оценить качество кластеризации, принимая во внимание меру компактности кластеров и их отделенность друг от друга. Коэффициент силуэта позволяет выбирать наилучший вариант кластеризации в таких случаях.

Распространенные алгоритмы кластеризации

Существует множество алгоритмов кластеризации, каждый из которых имеет свои особенности и применимость в различных ситуациях. Вот наиболее распространенные:

- К-средних (K-means): Он разбивает данные на K кластеров, где K задается пользователем. Алгоритм итеративно оптимизирует положения центроидов кластеров и присваивает объекты к ближайшим центроидам. Это один из наиболее широко используемых алгоритмов кластеризации.
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise): Этот алгоритм основан на плотности данных. Он ищет плотные области в пространстве данных и формирует кластеры на основе связанных плотных областей. DBSCAN может обнаруживать кластеры произвольной формы и обрабатывать выбросы.
- Иерархическая кластеризация: Это семейство алгоритмов, которые строят иерархическую структуру кластеров. Два основных подхода в иерархической кластеризации: агломеративный и дивизионный. Агломеративные методы действуют от частного к общему: начинают с отдельных объектов и последовательно объединяют их в кластеры, пока не будет достигнуто заданное условие остановки. Дивизионные методы действуют от общего к частному: начинают с одного крупного кластера и разделяют его на более мелкие, пока не будут получены отдельные кластеры.
 - Агломеративная кластеризация (Agglomerative Hierarchical Clustering): Это вариант иерархической кластеризации, где начинают с отдельных объектов и последовательно объединяют их на основе среднего расстояния между кластерами. Алгоритм объединяет кластеры, пока не будет достигнуто заданное количество кластеров.
- GMM (Gaussian Mixture Model): Этот алгоритм моделирует данные с использованием смеси нормальных (гауссовских) распределений. Каждый компонент смеси соответствует одному кластеру, и модель пытается найти наиболее вероятную смесь, описывающую данные.

Ниже представлено сравнение приведенных алгоритмов в форме таблицы (Таблица 1.1):

В этой таблице представлены основные характеристики каждого метода, их применимость в различных ситуациях и преимущества. Выбор алгоритма зависит от характеристик данных, размерности пространства, предполагаемого числа кластеров и других факторов.

Также приведем визуальное сравнение работы алгоритмов кластеризации с сайта sklearn (Рисунок 1.4).

На рисунке 1.4 представлена работа алгоритмов с разными датасетами: вложенные окружности, наборы данных в виде дуг (подков), три близко друг к другу расположенных кластера,

Метод	Основа алгоритма	Входные данные	Фиксированное количество кластеров	Форма кластеров	Обнаружение выбросов
К-средних	Расстояние между объектами и центроидами	Фактические наблюдения	Да	Сферические кластеры	Нет
DBSCAN	Плотность регионов в данных	Фактические наблюдения или попарные расстояния между наблюдениями	Нет	Произвольная	Да
Иерархическая кластеризация	Расстояние между объектами	Попарные расстояния между наблюдениями	Нет	Произвольная	Нет
Агломеративная кластеризация	Расстояние между объектами	Попарные расстояния между наблюдениями	Нет	Произвольная	Нет
GMM (Gaussian Mixture Model)	Смесь гауссовых распределений	Фактические наблюдения	Да	Сферические кластеры с различными ковариационными структурами	Да

Таблица 1.1: Сравнение основных алгоритмов кластеризации

три параллельных кластера, три отдаленных друг от друга кластера, один кластер. Качество работы алгоритма измеряется тем, насколько корректно он кластеризует эти данные и тем, сколько времени он потратит на это (информация о скорости работы алгоритма представлена в левом нижнем углу). Графическое сравнение алгоритмов на рисунке 1.4 показывает, что наиболее качественные результат кластеризации показывают DBSCAN и OPTICS, чуть менее качественные - агломеративная кластеризация. Наиболее сложные датасеты (строка один, два и шесть) кластеризуются лучше с помощью DBSCAN и OPTICS, однако DBSCAN более оптимален с точки зрения скорости работы.

Алгоритмы K-means, DBSCAN и агломеративная кластеризация далее будут рассмотрены более подробно.

1.1.2 Снижение размерности данных

Снижение размерности данных - это процесс уменьшения числа признаков (измерений) в наборе данных. Оно выполняется с целью упрощения анализа данных, устранения шума,

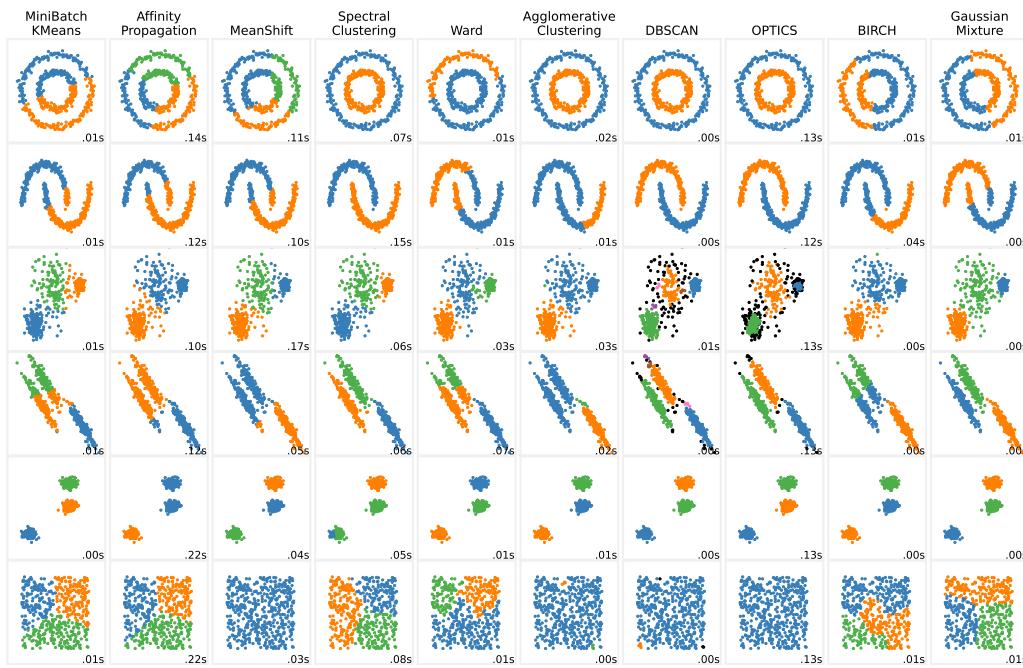


Рисунок 1.4: Сравнение качества работы алгоритмов кластеризации с помощью графиков

избавления от избыточной информации или подготовки данных для последующего использования в задачах машинного обучения.

Алгоритмы снижения размерности данных применяются в различных задачах анализа данных и машинного обучения. Вот несколько примеров задач, для которых широко используются методы снижения размерности:

- Визуализация данных: Снижение размерности помогает визуализировать данные высокой размерности в двух или трех измерениях для лучшего понимания структуры данных, обнаружения паттернов или визуального анализа.
- Отбор признаков: Методы снижения размерности могут использоваться для извлечения наиболее информативных признаков из исходных данных. Снижение размерности может быть применено перед построением моделей машинного обучения, чтобы уменьшить размерность входных данных и удалить избыточные или неинформативные признаки. Это может улучшить производительность моделей и сократить вычислительную сложность.
- Устранение шума: Методы снижения размерности могут помочь устраниить шум в данных, фильтруя или игнорируя менее значимые компоненты.
- Сжатие данных для уменьшения расхода памяти.

Это некоторые типы задач, для которых применяются алгоритмы снижения размерности данных. Выбор конкретного метода зависит от конкретной задачи, типа данных и требований анализа.

Распространенные алгоритмы снижения размерности данных

Существует множество методов для снижения размерности данных. Рассмотрим некоторые из них:

- Метод главных компонент (PCA): Он выполняет линейное преобразование данных, чтобы получить новые некоррелированные переменные, называемые главными компонентами. PCA — наиболее популярный метод снижения размерности.
- Метод t-SNE (t-Distributed Stochastic Neighbor Embedding): t-SNE позволяет визуализировать данные высокой размерности, сохраняя относительные расстояния между объектами. Он стремится сохранить соседство объектов в исходном пространстве и проектировать их на низкоразмерное пространство для визуализации.
- Метод линейного дискриминантного анализа (англ. Linear Discriminant Analysis, LDA): Линейный дискриминантный анализ - это статистический метод, используемый для анализа и классификации данных. Он относится к области обучения с учителем, где каждому образцу данных присваивается определенная метка класса. Основная цель LDA состоит в том, чтобы найти линейные комбинации признаков, которые наилучшим образом разделяют различные классы данных. Это достигается путем максимизации разделения между классами и минимизации разброса внутри каждого класса.
- Автоэнкодеры (Autoencoders): Автоэнкодеры являются нейронными сетями, которые обучаются восстанавливать входные данные на выходе. Внутри модели создаются представления данных меньшей размерности, которые модель затем пробует восстановить до исходной размерности. Автоэнкодеры могут быть использованы для снижения размерности и извлечения наиболее информативных признаков из данных.
- Корреляционный анализ: Корреляционный анализ - это статистический метод, который используется для измерения степени связи между двумя или более переменными. Он позволяет определить, насколько тесно связаны две переменные и какие типы связей между ними существуют. В корреляционном анализе используется коэффициент корреляции, который измеряет степень линейной зависимости между переменными.
- Информативный признаковый отбор (Information Gain) в алгоритмах на деревьях: Информативный признаковый отбор, или Information Gain, является методом выбора наиболее информативных признаков при построении деревьев решений или других алгоритмов, основанных на деревьях, таких как случайный лес.
- Алгоритм регуляризации L1: Применение L1 регуляризации приводит к решению, при котором некоторые коэффициенты модели становятся равными нулю. Это позволяет выполнить отбор признаков, исключая несущественные признаки из модели. Признаки с нулевыми коэффициентами считаются неинформативными или слабо влияющими на целевую переменную.

Ниже представлено сравнение приведенных алгоритмов в форме таблицы (Таблица 1.2):

В данной таблице представлены основные характеристики методов снижения размерности данных: PCA, t-SNE, LDA и Autoencoders. Каждый метод обладает своими особенностями, применимостью и преимуществами. Это позволяет выбрать подходящий в зависимости от целей и требований конкретной задачи снижения размерности данных. Пример визуализации датасета Iris с помощью алгоритмов снижения размерности PCA и LDA представлен на рисунке 1.5.

Алгоритмы PCA и t-SNE далее будут рассмотрены более подробно.

1.1.3 Вопросы для самопроверки

Для чего могут использоваться алгоритмы кластеризации данных (три ответа)?

1. Поиск скрытых структур

Метод	Описание	Преимущества	Недостатки
PCA	находит новое пространство признаков, состоящее из главных компонент исходных данных.	Простота и высокая скорость вычислений	Потеря информации о классификации
t-SNE	строит карту точек данных в низкоразмерном пространстве, сохраняя локальную структуру.	Сохранение локальной структуры данных	Вычислительно сложен и может быть медленным для больших наборов данных
LDA	находит новое пространство признаков, максимизируя разделение классов данных.	Учет информации о классификации	Могут возникнуть проблемы с переобучением, когда классов данных много или данные несбалансированы
Autoencoders	нейронные сети, которые обучаются восстанавливать входные данные с помощью сжатого представления.	Гибкость в моделировании сложных нелинейных отображений данных	Обучение требует больше вычислительных ресурсов и времени в сравнении с другими методами снижения размерности. Могут возникнуть проблемы с переобучением.

Таблица 1.2: Сравнение основных алгоритмов снижения размерности данных

2. Задача регрессии
3. Сжатие данных
4. Рекомендательные системы

Правильные ответы: 1, 3, 4

Какие из приведенных алгоритмов относятся к алгоритмам кластеризации (два ответа)?

1. kNN
2. K-средних
3. DBSCAN
4. AdaBoost

Правильные ответы: 2, 3

Какие из приведенных алгоритмов относятся к алгоритмам снижения размерности (два ответа)?

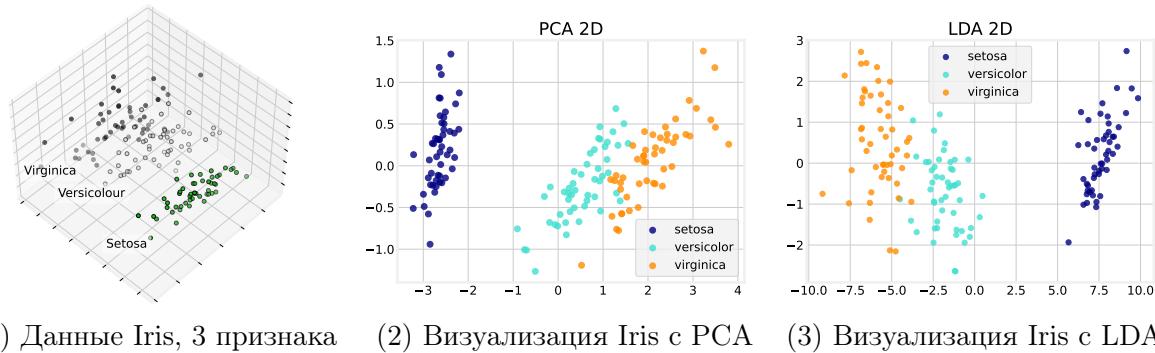


Рисунок 1.5: Визуализации датасета Iris с помощью алгоритмов снижения размерности PCA и LDA

1. GMM
2. t-SNE
3. DBSCAN
4. PCA

Правильные ответы: 2, 4

1.1.4 Резюме по разделу

Обучение без учителя (англ. Unsupervised Learning) - это раздел машинного обучения, в котором модель обучается на неразмеченных данных, то есть данных, не имеющих заранее определенных меток или целевых переменных. В отличие от обучения с учителем, где модель обучается на размеченных данных с известными метками классов или целевыми значениями, обучение без учителя направлено на обнаружение скрытых структур, закономерностей и интересных паттернов в данных.

Кластеризация и снижение размерности данных являются классическими подразделами обучения без учителя.

1. Кластеризация:

- Кластеризация является методом группировки данных в кластеры с похожими объектами.
- Распространенные метрики качества для оценки эффективности кластеризации: среднее внутрикластерное расстояние, среднее межкластерное расстояние, коэффициент силуэта, индекс Данна, индекс Дэвиса-Боулдина
- Распространенные алгоритмы кластеризации включают K-средних, DBSCAN, иерархическую кластеризацию и GMM.

2. Снижение размерности данных:

- Снижение размерности данных - это процесс уменьшения количества признаков в данных.
- Распространенные алгоритмы снижения размерности данных включают PCA (метод главных компонент), t-SNE (t-distributed stochastic neighbor embedding), LDA (линейный дискриминантный анализ) и автокодировщики (autoencoders).

1.2 K-means

Цель занятия: ученик может применить алгоритм K-means для решения задач кластеризации на подготовленных и неподготовленных данных.

План занятия:

- Визуальная демонстрация алгоритма
- Подготовка данных для алгоритма
- Процесс обучения
- Оценка качества алгоритма
- Применение алгоритма
- Плюсы и минусы алгоритма
- Реализация алгоритма в Python

1.2.1 Визуальная демонстрация алгоритма

К-средних (k-means) - это один из наиболее распространенных методов кластеризации, используемых в машинном обучении и анализе данных. Он представляет собой алгоритм, который разбивает набор данных на заранее заданное количество кластеров, где каждый кластер представляет группу объектов, которые схожи между собой и отличаются от объектов в других кластерах (Рисунок 1.6).

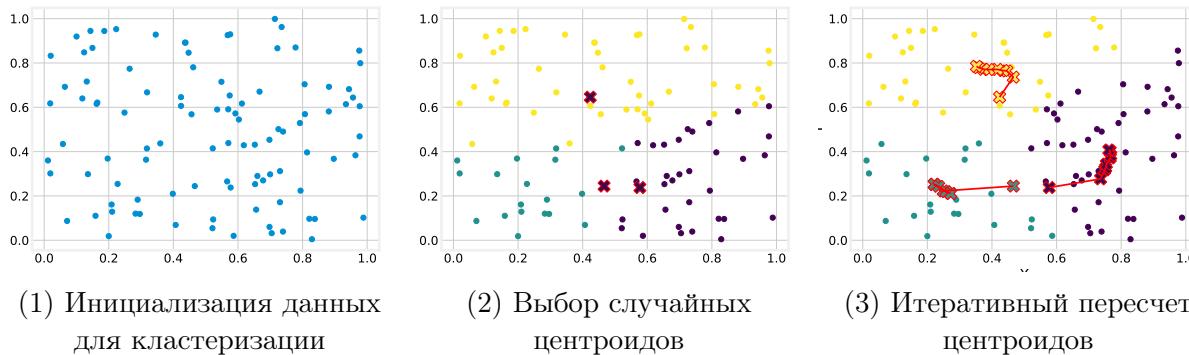


Рисунок 1.6: Процесс обучения K-means

На начальном этапе мы выбираем случайные центры для каждого кластера. Затем мы относим каждую точку к ближайшему кластеру на основе расстояния до центров. После этого мы пересчитываем центры кластеров, используя среднее арифметическое точек, принадлежащих кластеру.

Теперь, когда центры кластеров обновлены, мы повторяем процесс перераспределения точек по кластерам и обновления центров. Этот итеративный процесс повторяется до тех пор, пока центры кластеров и распределение точек не стабилизируются или достигнутся заранее заданные условия остановки.

Алгоритм k-means пытается минимизировать сумму квадратов расстояний между точками и центрами кластеров, что приводит к тому, что точки внутри каждого кластера становятся более близкими друг к другу, а точки между разными кластерами остаются далекими друг от друга.

1.2.2 Подготовка данных для алгоритма

Подготовка данных для алгоритма К-средних включает несколько шагов. Вот общий процесс подготовки данных для использования с алгоритмом К-средних:

- Масштабирование данных: Рекомендуется масштабировать признаки входных данных перед применением К-средних, особенно если признаки имеют различные шкалы или единицы измерения. Обычно используется стандартизация или нормализация данных для приведения их к общей шкале.
- Удаление выбросов: Выбросы могут негативно влиять на процесс кластеризации. Перед применением алгоритма К-средних рекомендуется удалить или корректировать выбросы в данных.
- Работа с категориальными переменными: Если у вас есть категориальные переменные в данных, требуется их преобразование в числовую формат. Вы можете использовать методы, такие как One-Hot Encoding или Label Encoding.
- Обработка пропущенных значений: Если в ваших данных есть пропущенные значения, нужно решить, как с ними поступить. Вы можете удалить строки или столбцы с пропущенными значениями или заполнить их средними или медианными значениями в зависимости от контекста данных.
- Отбор признаков: При наличии большого количества признаков важно выбрать наиболее релевантные и информативные признаки для кластеризации. Вы можете использовать методы отбора признаков или снижения размерности данных, такие как анализ главных компонент (PCA) или методы выбора признаков на основе значимости.
- Обработка корреляций: Если у вас есть сильные корреляции между признаками, это может влиять на работу алгоритма К-средних. Рекомендуется выполнить анализ корреляций и решить, какие признаки оставить, исключить или объединить в один признак.
- Учет особенностей данных: Если в ваших данных есть особенности, такие как выборки с несбалансированным размером кластеров или наличие шума, следует учесть эти особенности при выборе количества кластеров и интерпретации результатов.

1.2.3 Процесс обучения

Основная идея алгоритма k-means заключается в том, чтобы минимизировать сумму квадратов ошибок (Sum of Squared Errors), то есть отклонения между объектами внутри кластеров и их центроидами. Сумма квадратов ошибок представлена следующим образом (Формула 1.6):

$$SSE = \sum_{i=1}^n \sum_{j=1}^k (x_{ij} - c_j)^2 \quad (1.6)$$

где n - общее количество точек данных, k - общее количество кластеров, x_{ij} - j -я компонента i -й точки данных, и c_j - j -й центроид кластера. В этом уравнении мы вычисляем квадрат расстояния между каждой точкой данных x_{ij} и соответствующим ей центроидом c_j для всех точек данных и кластеров. Затем мы суммируем эти квадраты расстояний, чтобы получить SSE.

Выбор оптимального количества кластеров в алгоритме K-Means может быть сложной задачей. Вот некоторые методы, которые можно использовать для определения оптимального числа кластеров:

- Метод локтя (elbow method): Этот метод заключается в вычислении суммы квадратов ошибок (SSE) для различных значений числа кластеров и выборе значения, при котором увеличение числа кластеров не приводит к существенному снижению SSE. График зависимости SSE от числа кластеров будет иметь форму локтя, и оптимальное число кластеров будет соответствовать точке, где SSE перестает значительно уменьшаться.
- Коэффициент силуэта (silhouette coefficient): Этот коэффициент используется для оценки качества кластеризации. Он учитывает среднее расстояние между объектами внутри кластера и среднее расстояние до ближайшего соседнего кластера. Значение коэффициента силуэта находится в диапазоне от -1 до 1, где более близкое к 1 значение указывает на лучшую кластеризацию. Оптимальное количество кластеров можно выбрать на основе максимального значения коэффициента силуэта.
- Метод gap statistic: Этот метод сравнивает значения SSE для фактических данных с ожидаемыми значениями SSE в случайной выборке. Чем больше разница между фактическими и ожидаемыми значениями SSE, тем лучше модель кластеризации. Оптимальное количество кластеров выбирается на основе максимального значения гап между фактическими и ожидаемыми значениями SSE.
- Экспертные знания: Иногда экспертные знания о предметной области могут помочь в выборе оптимального количества кластеров. Если вы знакомы с данными и ожидаете определенное количество групп или паттернов, вы можете использовать это знание для выбора числа кластеров.

Важно отметить, что выбор оптимального числа кластеров является относительным и может зависеть от конкретной задачи и данных. Рекомендуется использовать несколько методов оценки и сравнить результаты.

Алгоритм выполняет такие шаги до сходимости:

1. Инициализация: Выбираются случайные центроиды для каждого из кластеров. Центроид представляет собой точку в пространстве признаков, которая является центром кластера.
2. Присваивание кластера: Каждый объект из набора данных присваивается к ближайшему центроиду. Близость обычно измеряется с использованием евклидового расстояния, но можно использовать и другие метрики.
3. Обновление центроидов: Расчет нового центроида для каждого кластера путем вычисления среднего значения всех объектов, принадлежащих к данному кластеру.

Шаги 2-3 повторяются до тех пор, пока не будет достигнут критерий сходимости, например, пока изменение центроидов становится незначительным или количество итераций достигает заранее определенного предела.

1.2.4 Оценка качества алгоритма

Оценка качества работы алгоритма К-средних (K-means) может быть выполнена с использованием нескольких метрик и методов. Вот некоторые распространенные способы оценки:

- Сумма квадратов ошибок (SSE): Это метрика, которая измеряет сумму квадратов расстояний между каждой точкой данных и центроидом ее кластера. Меньшее значение SSE указывает на лучшую кластеризацию. Однако SSE не является нормализованной метрикой и может зависеть от выбранного числа кластеров.

- Коэффициент силуэта (Silhouette coefficient): Это метрика, которая оценивает, насколько каждая точка данных хорошо соответствует своему собственному кластеру по сравнению с другими кластерами. Коэффициент силуэта принимает значения от -1 до 1, где ближе к 1 указывает на хорошую кластеризацию, а ближе к -1 - на неправильную кластеризацию.
- Индекс Дэвиса-Боулдина (Davies-Bouldin Index): Это индекс, который измеряет среднюю схожесть между кластерами и различие между кластерами. Меньшее значение индекса указывает на лучшую кластеризацию.
- Внутрикластерное расстояние и межкластерное расстояние: Можно также посмотреть на внутрикластерное расстояние (в среднем расстояние между точками внутри кластера) и межкластерное расстояние (среднее расстояние между центроидами кластеров). Хорошая кластеризация должна иметь маленькое внутрикластерное расстояние и большое межкластерное расстояние.
- Визуализация результатов: Визуализация кластеров может помочь визуально оценить качество работы алгоритма К-средних. Например, можно построить диаграмму рассеяния (англ. Scatter plot) и отобразить точки данных в пространстве признаков, окрашивая их в соответствии с принадлежностью кластерам. Пример визуального анализа с помощью диаграмм рассеяния приведен на рисунке 1.4.

1.2.5 Применение алгоритма

Алгоритм К-средних (K-means) имеет широкий спектр применений в различных областях. Некоторые из них включают:

- Кластерный анализ и сегментация данных: К-средних широко используется для кластеризации данных и сегментации на группы схожих объектов. Это может быть применено в маркетинге для сегментации клиентов на основе их предпочтений и поведения, в анализе данных для группировки схожих наблюдений и в биоинформатике для классификации геномных данных.
- Обработка изображений: К-средних может использоваться для сегментации изображений, разделения пикселей на различные группы в соответствии с их цветом или текстурой. Это может быть полезно, например, в распознавании образов, компьютерном зрении или сжатии изображений.
- Рекомендательные системы: В области рекомендательных систем К-средних может применяться для группировки пользователей или элементов в схожие кластеры. Это позволяет создавать персонализированные рекомендации, исходя из поведения или предпочтений пользователей.
- Анализ текстовых данных: К-средних может использоваться для анализа текстовых данных, например, для кластеризации документов по схожести содержания или группировки слов в тематические кластеры.
- Геоинформационные системы: В геоинформационных системах К-средних может быть применен для кластеризации пространственных данных, таких как точки на карте или географические регионы. Это может быть полезно для анализа распределения объектов, обнаружения паттернов или планирования маршрутов.
- Обнаружение аномалий: К-средних может использоваться для обнаружения аномальных наблюдений в данных. Аномальные точки могут быть удалены или выделены в

отдельные кластеры, что помогает в идентификации необычных событий или аномалий в различных областях, таких как финансы, кибербезопасность или медицинская диагностика.

Это только несколько примеров областей применения алгоритма K-средних, и он может быть использован во многих других сферах, где требуется кластеризация.

1.2.6 Плюсы и минусы алгоритма

Плюсы:

- Простота реализации и понимания: K-средних является простым и интуитивно понятным алгоритмом. Он основан на принципе минимизации суммы квадратов ошибок (SSE) и легко реализуется с помощью итеративного процесса.
- Высокая эффективность: K-средних имеет высокую вычислительную эффективность, особенно при больших объемах данных. Он может обрабатывать большие наборы данных относительно быстро и масштабируется для работы с большим количеством точек данных.
- Масштабируемость: Алгоритм K-средних хорошо масштабируется с увеличением количества кластеров или размера данных. Он может быть применен для различных задач кластеризации, начиная от небольшого числа кластеров до большого количества.
- Применимость к широкому спектру данных: K-средних может быть использован для различных типов данных, включая числовые данные, категориальные данные и даже текстовые данные. Он не зависит от распределения данных и может работать с разными типами признаков.
- Интерпретируемость результатов: Результаты K-средних могут быть легко интерпретированы и визуализированы. Кластеры, сформированные алгоритмом, представляют собой группы схожих объектов, что помогает в понимании структуры данных и выявлении паттернов.

Минусы:

- Чувствительность к выбору начальных центроидов: Результаты K-средних могут сильно зависеть от исходного выбора центроидов. Неправильная инициализация может привести к сходимости к локальному оптимуму, а не к глобальному оптимальному решению. Это может требовать множественного запуска алгоритма с разными начальными условиями.
- Зависимость от количества кластеров: Пользователь должен заранее определить количество кластеров K, что может быть сложной задачей. Неправильный выбор значения K может привести к неправильной кластеризации или объединению несхожих групп в один кластер.
- Предположение о выпуклости кластеров: Алгоритм K-средних предполагает, что кластеры являются выпуклыми. Выпуклость кластера является свойством структуры кластеризации данных. Она описывает, насколько кластер представляет собой выпуклую форму. Кластер считается выпуклым, если каждая точка внутри кластера находится внутри выпуклой оболочки кластера. Иначе говоря, в выпуклом кластере вы можете провести прямую линию из любой точки кластера в любую другую точку кластера, не покидая кластер. В случае, если данные имеют сложную форму или кластеры имеют различные формы и размеры, K-средних может давать неправильные результаты.

- Чувствительность к выбросам: К-средних чувствителен к выбросам в данных. Одиночные точки-выбросы могут сильно повлиять на положение центроидов и привести к искажению кластеризации.
- Нет гарантии глобального оптимума: Алгоритм К-средних может сойтись к локальному оптимуму, особенно при сложной структуре данных или плохой инициализации. В некоторых случаях может потребоваться использование более сложных алгоритмов кластеризации для достижения лучших результатов.

1.2.7 Реализация алгоритма в Python

K-means реализован в библиотеке sklearn. Для использования KMeans из библиотеки scikit-learn необходимо предварительно импортировать класс KMeans из модуля sklearn.cluster. Для задачи кластеризации с использованием K-means в библиотеке sklearn существует класс **KMeans**. Основные параметры модели:

- **n_clusters**: количество кластеров, которое требуется найти. Это обязательный параметр, необходимо его указать при создании объекта KMeans.
- **init**: метод инициализации начальных центроидов. Может принимать значения «*k-means++*», «*random*» или массив начальных центроидов. По умолчанию используется метод «*k-means++*». *K-means++* позволяет более равномерно распределить начальные центроиды по набору данных, что помогает избежать попадания в локальные минимумы и улучшает качество кластеризации.
- **n_init**: количество запусков алгоритма с различными начальными центроидами. По умолчанию *n_init*=10.
- **max_iter**: максимальное количество итераций для сходимости алгоритма. По умолчанию *max_iter*=300.
- **tol**: порог сходимости алгоритма. Если изменение SSE (суммы квадратов ошибок) между двумя последовательными итерациями меньше *tol*, алгоритм прекращает работу. По умолчанию *tol*=1e-4.

Класс KMeans также имеет методы *fit(X)* для обучения модели на данных X, *predict(X)* для присвоения меток кластеров новым данным X и *transform(X)* для преобразования данных X в матрицу расстояний до центроидов.

1.2.8 Вопросы для самопроверки

Какое предположение делает алгоритм K-Means о форме и размере кластеров?

1. Кластеры должны быть выпуклыми и иметь одинаковую дисперсию.
2. Кластеры могут иметь произвольную форму и размер.
3. Кластеры должны быть сферическими и иметь одинаковый радиус.
4. Кластеры должны быть линейно разделимыми.

Правильные ответы: 1

Какой параметр влияет на чувствительность алгоритма K-Means к выбросам?

1. Количество итераций (*max_iter*).

2. Количество кластеров ($n_clusters$).
3. Метод инициализации начальных центроидов ($init$).
4. Порог сходимости алгоритма (tol).

Правильные ответы: 4

Как выбрать оптимальное количество кластеров в алгоритме K-Means?

1. Выбрать количество кластеров, равное количеству уникальных меток в данных.
2. Выбрать количество кластеров, равное среднему значению внутрикластерной дисперсии.
3. Использовать метод локтя (elbow method), оценивая изменение суммы квадратов ошибок при различном числе кластеров.
4. Определить количество кластеров на основе экспертных знаний о данных.

Правильные ответы: 3, 4

1.2.9 Резюме по разделу

K-means - это простой и популярный алгоритм кластеризации, который позволяет разделить набор данных на группы (кластеры) на основе их схожести. Краткое резюме по K-means:

- K-means итеративно присваивает точки к ближайшим центроидам и обновляет центроиды до сходимости.
- Алгоритм стремится минимизировать внутрикластерную сумму квадратов ошибок (SSE), которая измеряет сумму квадратов расстояний между точками и соответствующими центроидами.
- K-means требует заранее заданное количество кластеров (K), и выбор оптимального значения K является важной задачей при использовании K-means.
- Качество работы алгоритма может быть оценено с помощью метрик, таких как коэффициент силуэта или индекс Дэвиса-Боулдина.
- K-means может быть применен в различных областях, таких как анализ данных, сегментация клиентов, обработка изображений и многое другое. Однако K-means имеет свои ограничения, такие как чувствительность к начальной инициализации и форме кластеров, а также предположение о выпуклых кластерах и равной дисперсии.

1.3 DBSCAN

Цель занятия: ученик может применить алгоритм DBSCAN для решения задач кластеризации на подготовленных и неподготовленных данных.

План занятия:

- Визуальная демонстрация алгоритма
- Подготовка данных для алгоритма
- Процесс обучения
- Оценка качества алгоритма
- Применение алгоритма
- Плюсы и минусы алгоритма
- Реализация алгоритма в Python

1.3.1 Визуальная демонстрация алгоритма

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) - это алгоритм кластеризации данных, который определяет кластеры на основе плотности точек в пространстве данных. В отличие от других алгоритмов кластеризации, таких как k-средних или иерархическая кластеризация, DBSCAN может обнаруживать кластеры произвольной формы и способен обрабатывать шумовые точки.

Принцип работы DBSCAN заключается в следующем:

1. Определение окрестности точки: Для каждой точки в пространстве данных определяется окрестность, состоящая из всех точек, находящихся в заданном радиусе от данной точки.
2. Определение основной точки: Если в окрестности точки находится не менее min_samples точек (включая саму точку), то эта точка считается основной точкой. На рисунке 1.7.1 основные точки обозначены красным цветом.
3. Формирование кластеров: Начиная с основных точек, алгоритм идентифицирует связанные основные точки и объединяет их в кластеры. Две точки считаются связанными, если есть путь от одной точки к другой через серию соседних точек, и каждая точка на этом пути также является основной.
4. Обработка граничных точек: Граничные точки находятся в окрестности основной точки, но сами не являются основными. Они могут принадлежать кластеру, если они связаны с основной точкой, но могут также оставаться неразмеченными. На рисунке 1.7.1 граничные точки обозначены синим цветом.
5. Идентификация шумовых точек: Точки, которые не являются ни основными, ни граничными, считаются шумовыми и не принадлежат ни к одному кластеру. На рисунке 1.7.1 шумовые точки обозначены темным цветом.

DBSCAN является одним из популярных алгоритмов кластеризации, особенно при работе с большими наборами данных и кластерами произвольной формы.

Визуальная демонстрация алгоритма представлена на рисунке 1.7.

На рисунке 1.7.1 красным выделены основные точки и их окрестности с параметром $\text{min_samples}=3$. На рисунке 1.7.2 инициализируется выборка. На рисунке 1.7.3 алгоритм делит выборку на кластеры, отмечая основные точки кругами большего радиуса, граничные точки - кругами меньшего радиуса и шумовые точки - кругами темного цвета. На рисунке 1.7.4 показана возможность ведения алгоритмом DBSCAN кластеров сложной формы.

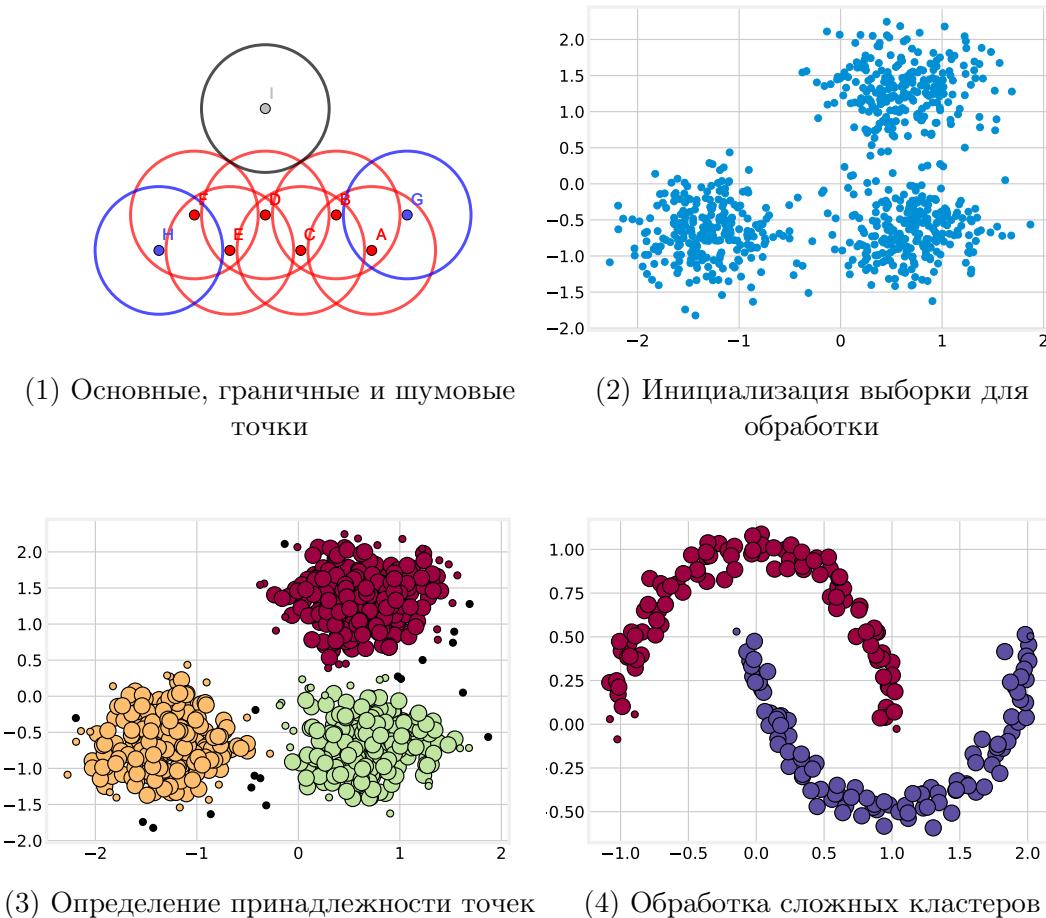


Рисунок 1.7: Демонстрация принципа работы алгоритма DBSCAN

1.3.2 Подготовка данных для алгоритма

Общий процесс подготовки данных для использования с алгоритмом DBSCAN:

- **Масштабирование данных:** Рекомендуется масштабировать признаки входных данных перед применением DBSCAN, особенно если признаки имеют различные шкалы или единицы измерения. Обычно используется стандартизация или нормализация данных для приведения их к общей шкале.
- **Удаление выбросов:** Выбросы могут оказывать влияние на результаты DBSCAN, поскольку алгоритм опирается на плотность точек. Перед применением DBSCAN рекомендуется удалить или корректировать выбросы в данных.
- **Работа с категориальными переменными:** Если у вас есть категориальные переменные в данных, требуется их преобразование в числовую формат. Вы можете использовать методы, такие как One-Hot Encoding или Label Encoding, чтобы закодировать категориальные переменные перед применением DBSCAN.
- **Обработка пропущенных значений:** Если в ваших данных есть пропущенные значения, нужно решить, как с ними поступить. Вы можете удалить строки или столбцы с пропущенными значениями или заполнить их средними или медианными значениями в зависимости от контекста данных.

- Отбор признаков: При наличии большого количества признаков важно выбрать наиболее релевантные и информативные признаки для кластеризации с помощью DBSCAN. Вы можете использовать методы отбора признаков или снижения размерности данных, такие как анализ главных компонент (PCA) или методы выбора признаков на основе значимости.
- Обработка корреляций: Если у вас есть сильные корреляции между признаками, это может влиять на работу DBSCAN. Рекомендуется выполнить анализ корреляций и решить, какие признаки оставить, исключить или объединить в один признак.
- Учет особенностей данных: Если в ваших данных есть особенности, такие как выборки с несбалансированным размером кластеров или наличие шума, следует учесть эти особенности при выборе параметров DBSCAN и интерпретации результатов.

1.3.3 Процесс обучения

Процесс обучения алгоритма DBSCAN состоит из следующих шагов:

1. Задание параметров: Прежде чем приступить к обучению DBSCAN, необходимо задать параметры алгоритма. Главными параметрами являются:
 - eps (epsilon): радиус окрестности, в пределах которой точки считаются соседними.
 - min_samples: минимальное количество точек в окрестности, необходимое для определения основной точки.Значения этих параметров должны быть выбраны в соответствии с характеристиками данных и требуемым уровнем плотности точек.
2. Вычисление плотности: Алгоритм DBSCAN анализирует плотность точек в пространстве данных. Для каждой точки вычисляется число соседей, находящихся в пределах заданного радиуса eps.
3. Определение типов точек: В зависимости от числа соседей каждая точка может быть классифицирована как:
 - Основная точка (core point): точка, для которой число соседей в окрестности eps превышает или равно min_samples.
 - Границчная точка (border point): точка, для которой число соседей в окрестности eps меньше min_samples, но находится в окрестности основной точки.
 - Шумовая точка (noise point): точка, для которой число соседей в окрестности eps меньше min_samples и не находится в окрестности основной точки.
4. Формирование кластеров: DBSCAN идентифицирует связанные основные точки и объединяет их в кластеры. Две точки считаются связанными, если есть путь от одной точки к другой через серию соседних точек, и каждая точка на этом пути также является основной. В результате формируются кластеры различной формы и размеров.
5. Маркировка граничных точек: Граничные точки, которые не принадлежат ни одному кластеру, могут быть отмечены соответствующим образом или оставаться без метки.

Важно отметить, что DBSCAN не требует явного обучения или итераций, как в случае с алгоритмами, основанными на центроидах, например, K-средних. Он основывается на анализе плотности точек и обнаружении связанных областей в данных. Поэтому процесс работы DBSCAN не включает фазы обучения и тестирования, а только параметризацию и применение алгоритма к данным.

1.3.4 Оценка качества алгоритма

Оценка качества работы алгоритма DBSCAN может быть выполнена с использованием следующих метрик и методов:

- Коэффициент силуэта (Silhouette coefficient): Как и в случае с алгоритмом K-средних, коэффициент силуэта может использоваться для оценки качества кластеризации с помощью DBSCAN. Он измеряет, насколько каждая точка данных хорошо соответствует своему кластеру по сравнению с другими кластерами. Значения коэффициента силуэта варьируются от -1 до 1, где ближе к 1 указывает на хорошую кластеризацию, ближе к -1 - на неправильную кластеризацию, а близость к 0 означает, что точки могут находиться на границе между кластерами.
- Количество обнаруженных кластеров: DBSCAN не требует задания числа кластеров заранее. Однако количество обнаруженных кластеров может быть использовано в качестве метрики для оценки работы алгоритма. Слишком большое или слишком маленькое количество кластеров может быть признаком неправильной кластеризации или недостаточной гибкости алгоритма.
- Визуализация результатов: Визуализация кластеров, полученных с помощью DBSCAN, может быть полезным способом оценки их качества. Аналогично K-средним, можно построить диаграмму рассеяния и окрасить точки данных в соответствии с принадлежностью кластерам. Визуальное исследование может помочь определить, насколько хорошо алгоритм обнаруживает плотные области данных и как обрабатывает шумовые точки.
- Отношение шума к кластерам: DBSCAN также может быть оценен по отношению шумовых точек к обнаруженным кластерам. Слишком большое количество шумовых точек может быть признаком плохой кластеризации или неправильно выбранных параметров алгоритма.

Важно отметить, что DBSCAN, в отличие от K-средних, может обнаруживать выбросы как шумовые точки, и его оценка может отличаться в зависимости от природы данных и параметров алгоритма.

1.3.5 Применение алгоритма

Алгоритм DBSCAN имеет широкий спектр применений в различных областях. Некоторые из них включают:

- Кластерный анализ и сегментация данных: DBSCAN используется для кластеризации данных и сегментации на группы схожих объектов. Он позволяет обнаруживать плотные области в данных и отделять их от разреженных областей. Применяется, например, для сегментации географических данных, обнаружения сообществ в социальных сетях или анализа покупательского поведения.
- Обнаружение выбросов и аномалий: DBSCAN может использоваться для обнаружения выбросов и аномалий в данных. Шумовые точки, которые не принадлежат ни к одному кластеру, могут быть идентифицированы как выбросы или аномалии. Применяется, например, для обнаружения аномальных транзакций в финансовых данных или необычных медицинских записей.
- Геоинформационные системы: В геоинформационных системах DBSCAN может быть использован для кластеризации пространственных данных, таких как точки на карте,

географические регионы или географические маршруты. Применяется, например, для анализа распределения объектов на карте, выявления географических паттернов или планирования оптимальных маршрутов.

- Сегментация изображений: DBSCAN может быть применен для сегментации изображений, разделения пикселей на группы в соответствии с их пространственным расположением. Применяется, например, в компьютерном зрении для выделения объектов на изображении или сегментации текстурных областей.
- Рекомендательные системы: DBSCAN может использоваться в рекомендательных системах для группировки пользователей или элементов на основе их пространственной близости или схожести. Это позволяет создавать персонализированные рекомендации или группировать похожие элементы в рекомендациях.
- Анализ сетей и связей: DBSCAN может быть применен для анализа структуры сетей или связей между объектами, например, он позволяет выявлять сообщества в социальных сетях.

Это только несколько примеров областей применения алгоритма DBSCAN, и он может быть использован во многих других сферах, где требуется кластеризация.

1.3.6 Плюсы и минусы алгоритма

Плюсы:

- Способность обнаруживать кластеры произвольной формы: DBSCAN не требует заранее заданного числа кластеров и способен обнаруживать кластеры произвольной формы. Он основывается на плотности точек данных и может определить плотные области в данных, независимо от их формы.
- Работа с шумом и выбросами: DBSCAN может эффективно обрабатывать шумовые точки и выбросы в данных. Он идентифицирует точки, которые не принадлежат ни к одному кластеру, и относит их к категории шума. Это делает DBSCAN устойчивым к выбросам и способным обрабатывать данные с неопределенными или неструктурированными областями.
- Не требует предварительной спецификации числа кластеров: DBSCAN не требует заранее заданного числа кластеров, в отличие от некоторых других алгоритмов кластеризации. Это позволяет ему автоматически определять число кластеров на основе плотностной структуры данных.
- Гибкость в определении плотности: DBSCAN позволяет гибко настраивать параметры, определяющие плотность кластеров. Можно задать радиус окрестности точки (*epsilon*) и минимальное количество точек в окрестности (*min_samples*), чтобы определить, что точка является ядром кластера. Это позволяет адаптировать DBSCAN под различные плотностные характеристики данных.
- Эффективность для больших объемов данных: DBSCAN может быть эффективным для обработки больших объемов данных. Он использует индексацию и структуры данных, такие как деревья, для ускорения поиска ближайших соседей и определения плотных областей.
- Способность к обнаружению выбросов и аномалий: Благодаря способности DBSCAN идентифицировать шумовые точки, он может быть использован для обнаружения выбросов и аномалий в данных. Это полезно во многих областях, таких как обнаружение мошенничества, анализ аномалий или медицинская диагностика.

Минусы:

- Чувствительность к выбору параметров: DBSCAN требует настройки двух основных параметров - радиуса окрестности (`epsilon`) и минимального количества точек в окрестности (`min_samples`). Выбор подходящих значений этих параметров может быть нетривиальной задачей и требует предварительного анализа данных или итеративного подбора. Неправильный выбор параметров может привести к неправильной классификации или объединению кластеров.
- Сложность работы с данными разной плотности: DBSCAN может иметь проблемы с кластеризацией данных, содержащих кластеры разной плотности. В случае, когда кластеры имеют значительно различающуюся плотность, может возникнуть сложность в правильном определении радиуса окрестности (`epsilon`) и минимального количества точек в окрестности (`min_samples`) для каждого кластера.
- Зависимость от плотности данных: DBSCAN может столкнуться с проблемой в случае данных с неравномерной плотностью. В областях, где плотность данных существенно меняется, алгоритм может иметь трудности с корректным определением границ кластеров.
- Сложности при работе с высокоразмерными данными: DBSCAN имеет сложности при работе с высокоразмерными данными, так как пространство высокой размерности может быть разреженным и иметь проблемы с определением плотных областей. Это может привести к неправильной кластеризации или потере информации.
- Отсутствие поддержки иерархической кластеризации: DBSCAN не поддерживает иерархическую кластеризацию, то есть невозможно получить иерархию кластеров с различными уровнями детализации.

1.3.7 Реализация алгоритма в Python

Алгоритм DBSCAN реализован в библиотеке scikit-learn и доступен через класс `DBSCAN` из модуля `sklearn.cluster`. Для использования `DBSCAN` в `scikit-learn` необходимо предварительно импортировать этот класс.

Вот основные параметры модели `DBSCAN`:

- **`eps`**: радиус окрестности, в пределах которой должно находиться минимальное количество точек, чтобы рассматриваемая точка была считана основной (core point). Это обязательный параметр, который следует выбирать с учетом особенностей данных.
- **`min_samples`**: минимальное количество точек, которые должны находиться в радиусе окрестности (`eps`), чтобы рассматриваемая точка была считана основной (core point). По умолчанию `min_samples=5`.
- **`metric`**: используемая метрика для измерения расстояния между точками. Может принимать различные значения, такие как «`euclidean`» (евклидово расстояние), «`manhattan`» (манхэттенское расстояние) и другие. По умолчанию `metric=«euclidean»`.
- **`algorithm`**: используемый алгоритм для выполнения `DBSCAN`. Может принимать значения «`auto`», «`ball_tree`», «`kd_tree`» или «`brute`». По умолчанию `algorithm=«auto»`, что позволяет автоматически выбрать наиболее эффективный алгоритм на основе входных данных.
- **`leaf_size`**: размер листа, используемый в алгоритмах «`ball_tree`» или «`kd_tree`». Этот параметр влияет на скорость построения дерева и использование памяти. По умолчанию `leaf_size=30`.

- **metric_params:** дополнительные параметры, которые могут быть переданы используемой метрике. Например, для метрики Минковского (Minkowski) можно указать значение параметра p через `metric_params='p': 2`.

Класс DBSCAN также имеет методы `fit(X)` для обучения модели на данных X , `fit_predict(X)` для кластеризации данных и присвоения меток кластеров, а также методы `fit_transform(X)` и `transform(X)` для преобразования данных X в матрицу расстояний до основных точек и кластеризации соответственно.

1.3.8 Вопросы для самопроверки

Какие типы точек присутствуют в алгоритме DBSCAN (один ответ)?

1. Основные (core points), граничные (border points) и шумовые (outliers)
2. Центроиды (centroids), границы (boundaries) и фоновые (background)
3. Критические (critical points), периферийные (peripheral points) и шумовые (outliers)
4. Произвольные (arbitrary points), отдельные (isolated points) и фоновые (background)

Правильные ответы: 1

Какие основные параметры нужно задать для алгоритма DBSCAN (один ответ)?

1. Количество кластеров и метод инициализации центроидов
2. Радиус окрестности и минимальное количество точек в окрестности
3. Максимальное количество итераций и порог сходимости
4. Метрика расстояния и размер листа для алгоритма кластеризации

Правильные ответы: 2

Плюсы алгоритма DBSCAN (четыре ответа)?

1. Гибкость в определении плотности
2. Зависимость от плотности данных
3. Отсутствие поддержки иерархической кластеризации
4. Работа с шумом и выбросами
5. Сложности при работе с высокоразмерными данными
6. Сложность работы с данными разной плотности
7. Способность к обнаружению выбросов и аномалий
8. Способность обнаруживать кластеры произвольной формы

Правильные ответы: 1, 4, 7, 8

1.3.9 Резюме по разделу

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) - это алгоритм кластеризации, который основывается на плотности данных. Вместо предварительно заданного числа кластеров, DBSCAN автоматически определяет количество и форму кластеров, исходя из плотности точек данных.

Основные преимущества DBSCAN:

- Способность обнаруживать кластеры произвольной формы: DBSCAN может успешно обрабатывать кластеры произвольной формы, включая кластеры с несферическими или неравными размерами.
- Не нужно указывать количество кластеров: DBSCAN автоматически определяет количество кластеров на основе плотности данных, что делает его особенно полезным при отсутствии заранее известной информации о количестве кластеров.
- Устойчивость к выбросам: DBSCAN способен обнаруживать и отделять выбросы (точки, не принадлежащие к какому-либо кластеру) от кластеров, что помогает в обработке шумных данных.

Некоторые ограничения DBSCAN:

- Зависимость от параметров: DBSCAN требует задания двух параметров - радиуса окрестности (eps) и минимального числа точек в окрестности (min_samples). Неправильные выбранные значения этих параметров могут привести к нежелательным результатам.
- Чувствительность к плотности данных: DBSCAN может столкнуться с трудностями, когда в данных присутствует значительная разница в плотности между кластерами. В таких случаях, определение подходящих значений параметров может быть сложным.
- Высокая вычислительная сложность: Алгоритм DBSCAN может быть вычислительно требовательным при работе с большими объемами данных, особенно если данные не были предварительно отфильтрованы или преобразованы.

В целом, DBSCAN является мощным инструментом для кластеризации данных, особенно в случаях, когда форма и количество кластеров неизвестны заранее, и когда важно выделение выбросов и обработка шумных данных. Однако правильный выбор параметров и управление вычислительной сложностью могут потребовать дополнительных усилий и экспериментов.

1.4 Агломеративная кластеризация

Цель занятия: ученик может применить алгоритм агломеративной кластеризации для решения задач кластеризации на подготовленных и неподготовленных данных.

План занятия:

- Визуальная демонстрация алгоритма
- Подготовка данных для алгоритма
- Процесс обучения
- Оценка качества алгоритма
- Применение алгоритма
- Плюсы и минусы алгоритма
- Реализация алгоритма в Python

1.4.1 Визуальная демонстрация алгоритма

Агломеративная кластеризация — это один из методов кластерного анализа, который используется для группировки объектов в иерархическую структуру. Он начинается с того, что каждый объект представляется отдельным кластером, а затем последовательно объединяет наиболее похожие кластеры до тех пор, пока не будет получен единственный кластер, содержащий все объекты. Визуально этот процесс можно представить следующим образом (Рисунок 1.8). На каждом шаге мы сливаем два наиболее близких кластера, а также добавляем новый уровень в дендрограмму.

Дендрограмма - это графическое представление результатов иерархической кластеризации. Она представляет собой дерево-структуру, где каждый узел представляет кластер или группу объектов, а ветви указывают на близость или расстояние между кластерами.

На дендрограмме ось Y представляет меру расстояния или сходства между кластерами. Чем выше на оси Y находится точка слияния кластеров, тем дальше они друг от друга или менее похожи. Можно провести горизонтальную линию через дендрограмму, чтобы определить, какие кластеры объединяются на каждом уровне.

Ось X дендрограммы представляет собой список объектов или кластеров. Каждая точка на оси X соответствует отдельному объекту или кластеру. Путем анализа дендрограммы можно определить, какие объекты или кластеры объединяются на каждом уровне иерархии.

Дендрограммы широко используются в кластерном анализе для визуализации и интерпретации результатов. Они помогают исследователям определить оптимальное количество кластеров или иерархическую структуру данных. Также дендрограммы могут быть полезны при принятии решений о разделении кластеров на разных уровнях детализации, а также для анализа подобия или сходства между кластерами.

После того, как построен единственный кластер и дендрограмма, по дендрограмме можно выбрать количество кластеров, выбрав соответствующий уровень дендрограммы для разбиения на кластеры (на рисунке 1.8.8 один из вариантов разбиения выделен красным пунктиром).

1.4.2 Подготовка данных для алгоритма

Ниже представлен общий процесс подготовки данных для использования с алгоритмом агломеративной кластеризации:

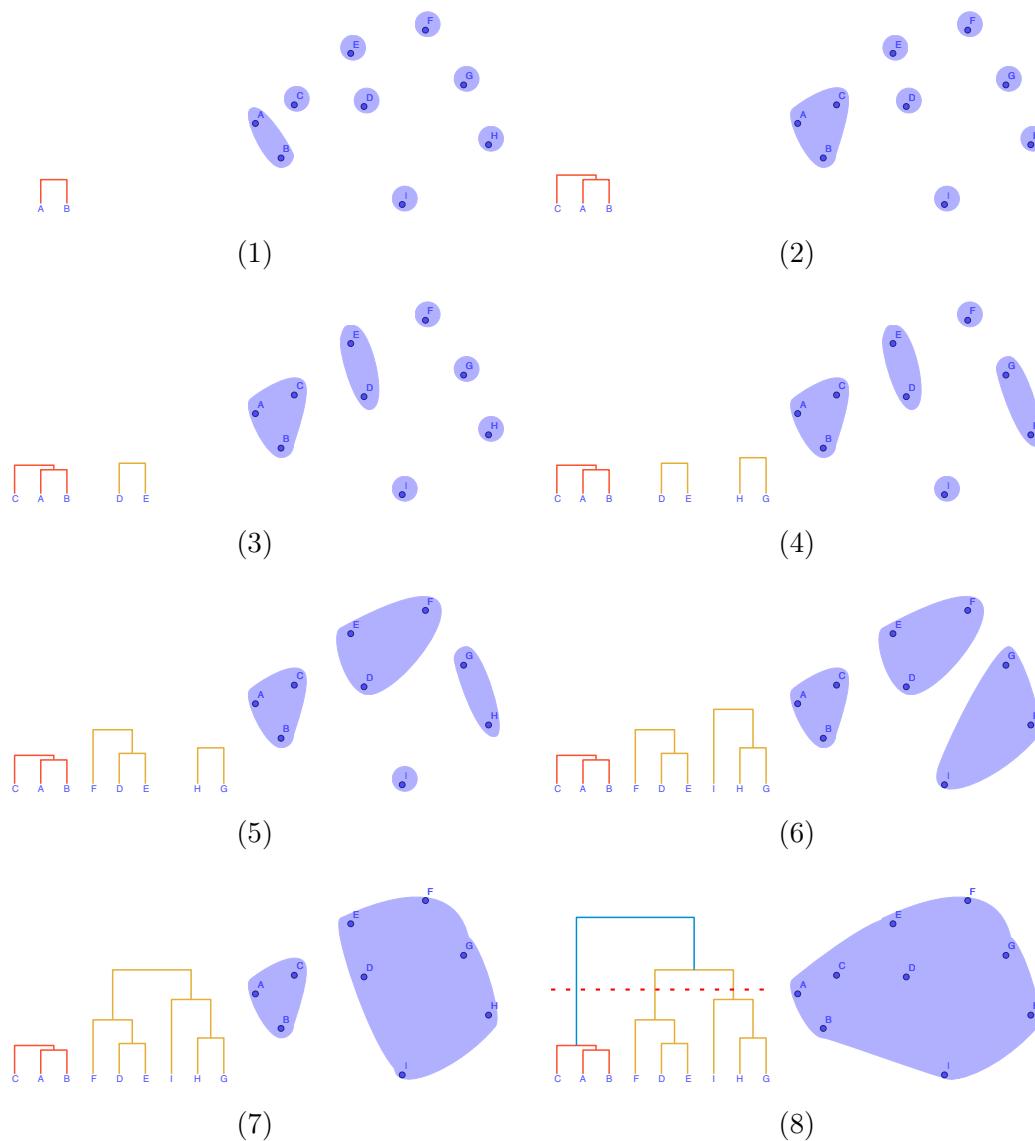


Рисунок 1.8: Пошаговый процесс построения дендрограммы и единственного кластера для агломеративной кластеризации

- **Масштабирование данных:** Рекомендуется масштабировать признаки входных данных перед применением агломеративной кластеризации, особенно если признаки имеют различные шкалы или единицы измерения. Обычно используется стандартизация или нормализация данных для приведения их к общей шкале.
- **Обработка категориальных переменных:** Если у вас есть категориальные переменные в данных, требуется их преобразование в числовую формат. Вы можете использовать методы, такие как One-Hot Encoding или Label Encoding, чтобы закодировать категориальные переменные перед применением агломеративной кластеризации.
- **Обработка пропущенных значений:** Если в ваших данных есть пропущенные значения, нужно решить, как с ними поступить. Вы можете удалить строки или столбцы с пропущенными значениями или заполнить их средними или медианными значениями в зависимости от контекста данных.

- Отбор признаков: При наличии большого количества признаков важно выбрать наиболее релевантные и информативные признаки для кластеризации с помощью агломеративной кластеризации. Вы можете использовать методы отбора признаков или снижения размерности данных, такие как анализ главных компонент (PCA) или методы выбора признаков на основе значимости.
- Обработка корреляций: Если у вас есть сильные корреляции между признаками, это может влиять на работу агломеративной кластеризации. Рекомендуется выполнить анализ корреляций и решить, какие признаки оставить, исключить или объединить в один признак.
- Учет особенностей данных: Если в ваших данных есть особенности, такие как выборки с несбалансированным размером кластеров или наличие шума, следует учесть эти особенности при выборе параметров агломеративной кластеризации и интерпретации результатов.

1.4.3 Процесс обучения

Вот основные шаги агломеративной кластеризации:

1. Инициализация: Каждый объект представляется в отдельном кластере.
2. Вычисление матрицы расстояний: Вычисляется матрица расстояний между всеми парами кластеров. Расстояние может быть определено различными способами, например, евклидово расстояние или корреляция.
3. Объединение кластеров: Наиболее похожие кластеры объединяются в один кластер на основе определенного критерия объединения. Распространенными критериями являются минимальное расстояние (single linkage), максимальное расстояние (complete linkage) или среднее расстояние (average linkage) между кластерами.
4. Обновление матрицы расстояний: Матрица расстояний обновляется, чтобы отразить новые расстояния между объединенными кластерами.
5. Повторение шагов 3 и 4: Шаги объединения кластеров и обновления матрицы расстояний повторяются до тех пор, пока все объекты не будут объединены в один кластер.
6. Формирование дендрограммы: В результате агломеративной кластеризации можно получить дендрограмму, которая представляет собой дерево объединений кластеров. По оси X дендрограммы отображаются объекты или кластеры, а по оси Y отображается мера расстояния или сходства.

Агломеративная кластеризация не требует заранее заданного числа кластеров и позволяет исследовать структуру данных на разных уровнях детализации. Этот метод широко используется в различных областях, включая анализ данных, биоинформатику и т. д.

1.4.4 Оценка качества алгоритма

Оценка качества алгоритма агломеративной кластеризации может быть выполнена с использованием различных метрик и методов. Вот некоторые распространенные меры, которые могут быть использованы для оценки качества агломеративной кластеризации:

- Внутрикластерное расстояние: Эта метрика измеряет среднее расстояние между объектами внутри каждого кластера. Чем меньше значение внутrikластерного расстояния, тем лучше.

- Межклластерное расстояние: Эта метрика измеряет расстояние между центроидами или центрами масс различных кластеров. Чем больше значение межклластерного расстояния, тем лучше.
- Коэффициент силуэта: Коэффициент силуэта оценивает качество кластеризации, учитывая как близость объектов внутри своего кластера, так и удаленность от соседних кластеров. Значение коэффициента силуэта находится в диапазоне от -1 до 1, где значения ближе к 1 указывают на хорошую кластеризацию.

Могут использоваться и другие метрики в соответствии с контекстом оценки качества алгоритма агломеративной кластеризации.

1.4.5 Применение алгоритма

Алгоритм агломеративной кластеризации широко применяется в различных областях анализа данных и машинного обучения. Вот некоторые области, где алгоритм агломеративной кластеризации может быть полезным:

- Маркетинг и сегментация потребителей: Агломеративная кластеризация может использоваться для идентификации групп потребителей с похожими предпочтениями, поведением или демографическими характеристиками. Это позволяет компаниям создавать более целевые маркетинговые стратегии и персонализированные предложения.
- Биология и генетика: Агломеративная кластеризация может быть применена для анализа генетических данных и выявления генетических паттернов или групп, связанных с определенными заболеваниями или фенотипами. Это помогает в понимании генетической структуры и классификации образцов.
- Обработка естественного языка: В анализе текстовых данных агломеративная кластеризация может использоваться для группировки документов или текстовых фрагментов по сходству содержания. Это может быть полезно, например, для создания тематических кластеров или выявления паттернов в текстовых данных.
- Обработка изображений и компьютерное зрение: Агломеративная кластеризация может применяться для сегментации изображений по сходству пикселей или объектов на изображении. Это используется, например, для распознавания образов или выделения регионов интереса на изображении.
- Социальные сети и анализ социальных данных: Агломеративная кластеризация может помочь в анализе социальных сетей и идентификации групп пользователей с похожими интересами, связями или поведением. Это полезно для рекомендательных систем, персонализации контента и анализа влияния в социальных сетях.

Это только несколько примеров областей применения агломеративной кластеризации. Алгоритм может быть использован во многих других областях, где требуется группировка и классификация данных на основе их сходства.

1.4.6 Плюсы и минусы алгоритма

Плюсы

- Простота реализации: Агломеративная кластеризация является относительно простым алгоритмом, который легко реализовать и понять. Он основывается на простых принципах объединения ближайших точек или кластеров на каждом шаге.

- Гибкость и масштабируемость: Алгоритм агломеративной кластеризации может быть применен к различным типам данных и может обрабатывать большие объемы данных. Он может работать с различными мерами расстояния и сходства, что позволяет адаптировать его к различным задачам.
- Иерархическая структура: Агломеративная кластеризация создает иерархическую структуру кластеров, которая позволяет анализировать данные на разных уровнях детализации. Это позволяет получить информацию о более общих группах и более специфичных подгруппах данных.
- Способность обнаруживать кластеры различных форм и размеров: Алгоритм агломеративной кластеризации способен обнаруживать кластеры различных форм, размеров и плотности. Он может обрабатывать кластеры с произвольной формой и даже с неоднородной плотностью.
- Возможность визуализации: Иерархическая структура, созданная алгоритмом агломеративной кластеризации, может быть визуализирована в виде дендрограммы. Дендрограмма представляет собой графическое представление иерархии кластеров, что упрощает интерпретацию результатов.

Минусы

- Высокая вычислительная сложность: Агломеративная кластеризация имеет квадратичную сложность по времени, что делает ее менее эффективной для больших наборов данных. Вычисление расстояний между всеми парами точек может быть ресурсоемкой операцией.
- Чувствительность к выбору меры расстояния: Выбор подходящей меры расстояния является критическим для успешной работы агломеративной кластеризации. Разные меры могут давать различные результаты, и некорректный выбор меры может привести к искажению кластерной структуры.
- Отсутствие контроля над размером кластеров: Алгоритм агломеративной кластеризации не предоставляет явного способа контроля над размером получаемых кластеров. Размеры кластеров могут существенно отличаться, и в некоторых случаях это может быть проблемой.

1.4.7 Реализация алгоритма в Python

Агломеративная кластеризация реализована в библиотеке scikit-learn и доступна через класс AgglomerativeClustering из модуля sklearn.cluster.

Основные параметры модели AgglomerativeClustering:

- **n_clusters:** количество кластеров, которые требуется сформировать. Это обязательный параметр.
- **affinity:** метрика, используемая для вычисления расстояния между точками. Может принимать различные значения, такие как «euclidean» (евклидово расстояние), «manhattan» (манхэттенское расстояние), «cosine» (косинусное расстояние) и другие.
- **linkage:** метод объединения кластеров. Может принимать значения «ward», «complete», «average» и «single».
- **distance_threshold:** пороговое расстояние, при котором прекращается объединение кластеров. Если не указано, то все кластеры будут объединены.

Класс AgglomerativeClustering также имеет методы fit(X) для обучения модели на данных X и fit_predict(X) для кластеризации данных и присвоения меток кластеров.

1.4.8 Вопросы для самопроверки

Какова основная идея агломеративной кластеризации (один ответ)?

1. Разделение данных на заранее заданное число кластеров.
2. Объединение ближайших элементов в иерархическую структуру.
3. Поиск наиболее отдаленных элементов для формирования кластеров.
4. Применение деревьев решений для расчета расстояния между элементами.

Правильные ответы: 2

Какая метрика используется для вычисления расстояния между точками в агломеративной кластеризации?

1. Евклидово расстояние
2. Манхэттенское расстояние
3. Косинусное расстояние
4. Все вышеперечисленные

Правильные ответы: 4

Выберите плюсы агломеративной кластеризации (четыре ответа):

- Возможность визуализации
- Высокая вычислительная сложность
- Гибкость и масштабируемость
- Простота реализации
- Способность обнаруживать кластеры различных форм и размеров
- Трудность интерпретации результатов

Правильные ответы: 1, 3, 4, 5

1.4.9 Резюме по разделу

Агломеративная кластеризация - это метод машинного обучения, который используется для группировки объектов на основе сходства между ними. Этот метод начинает с каждого объекта, рассматриваемого как отдельный кластер, и объединяет их по мере того, как они становятся более похожими друг на друга. Для этого используются различные меры сходства, такие как евклидово расстояние или корреляция. Результатом агломеративной кластеризации является дерево, называемое дендрограммой, которое показывает, как объекты были объединены в кластеры.

1.5 Метод главных компонент

Цель занятия: ученик может применить метод главных компонент (PCA) для решения задач снижения размерности на подготовленных и неподготовленных данных.

План занятия:

- Визуальная демонстрация алгоритма
- Подготовка данных для алгоритма
- Процесс обучения
- Оценка качества алгоритма
- Применение алгоритма
- Плюсы и минусы алгоритма
- Реализация алгоритма в Python

Метод главных компонент (англ. Principal Component Analysis, PCA) — это статистический метод, используемый для уменьшения размерности данных и извлечения наиболее значимых информационных характеристик из множества переменных. Он позволяет сжать исходный набор данных, сохраняя при этом наибольшее количество информации. Он основан на линейном преобразовании данных с целью найти новые оси (главные компоненты), вдоль которых данные имеют наибольшую изменчивость. Главные компоненты являются линейными комбинациями исходных признаков и ортогональны (перпендикулярны) друг другу. Они упорядочены по убыванию объясненной ими дисперсии данных.

Для объяснения принципов работы PCA необходимо рассмотреть некоторые понятия.

Дисперсия - это мера разброса или изменчивости случайной величины. Она показывает, насколько сильно значения случайной величины отклоняются от ее среднего значения.

Для случайной величины X с функцией вероятности (или функцией плотности вероятности) $f(x)$ и средним значением μ , дисперсия σ^2 определяется следующим образом:

$$\sigma^2 = E[(X - \mu)^2] = \int (x - \mu)^2 f(x) dx$$

где $E[\cdot]$ обозначает математическое ожидание, а интеграл берется по всем возможным значениям x случайной величины.

Дисперсия является положительным числом и представляет собой среднюю квадратическую разницу между значениями случайной величины и ее средним значением. Чем больше дисперсия, тем больше вариативность или разброс значений случайной величины.

В контексте PCA, объясненная дисперсия относится к количеству дисперсии в исходных данных, которая объясняется каждой главной компонентой.

Вычисление Среднее_{PC1} (среднего значения первой главной компоненты) осуществляется путем суммирования всех значений первой главной компоненты и деления этой суммы на общее количество наблюдений.

Математически, Среднее_{PC1} можно вычислить следующим образом:

$$\text{Среднее}_{\text{PC1}} = \frac{\sum_{i=1}^n \text{PC1}_i}{n}$$

где PC1_i - значения первой главной компоненты для каждого наблюдения, а n - общее количество наблюдений.

Объясненная дисперсия первой главной компоненты может быть записана как:

$$\text{Дисперсия}_{\text{PC}1} = \frac{\sum_{i=1}^n (\text{PC}1_i - \text{Среднее}_{\text{PC}1})^2}{n}$$

где $\text{PC}1_i$ - значения первой главной компоненты для каждого наблюдения, $\text{Среднее}_{\text{PC}1}$ - среднее значение первой главной компоненты, и n - общее количество наблюдений.

Аналогично может быть вычислена объясненная дисперсия для других главных компонент.

Объясненная дисперсия позволяет определить, какую долю дисперсии данных объясняет каждая главная компонента. Чем больше объясненная дисперсия, тем больше информации оригинальных данных сохраняется в соответствующей главной компоненте.

PCA находит главные компоненты путем вычисления собственных значений и собственных векторов ковариационной матрицы данных.

Ковариация — это мера статистической зависимости между двумя случайными переменными. Она измеряет, насколько две переменные изменяются вместе. Ковариация может быть положительной, если две переменные сильно положительно коррелируют (увеличение одной переменной связано с увеличением другой), отрицательной, если они сильно отрицательно коррелируют (увеличение одной переменной связано с уменьшением другой), или близкой к нулю, если между ними нет линейной зависимости (Рисунок 1.9).

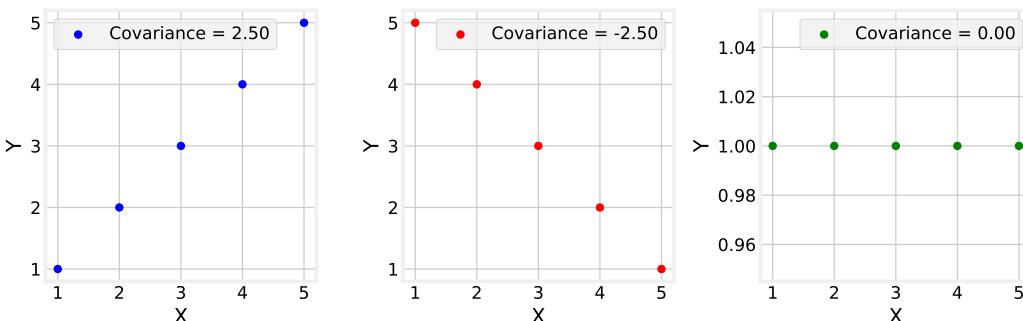


Рисунок 1.9: Положительная, отрицательная и нулевая ковариация переменных X и Y

Ковариация двух случайных переменных X и Y может быть записана следующим образом:

$$\text{cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$$

где $E[X]$ и $E[Y]$ обозначают математические ожидания (средние значения) переменных X и Y соответственно.

Ковариационная матрица - это квадратная матрица, которая содержит ковариации между парами случайных переменных. Для n случайных переменных X_1, X_2, \dots, X_n , ковариационная матрица C определяется следующим образом:

$$C = \begin{bmatrix} \text{cov}(X_1, X_1) & \text{cov}(X_1, X_2) & \dots & \text{cov}(X_1, X_n) \\ \text{cov}(X_2, X_1) & \text{cov}(X_2, X_2) & \dots & \text{cov}(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(X_n, X_1) & \text{cov}(X_n, X_2) & \dots & \text{cov}(X_n, X_n) \end{bmatrix}$$

где $\text{cov}(X_i, X_j)$ представляет собой ковариацию между X_i и X_j .

Ковариационная матрица является симметричной. В общем случае ковариация между двумя случайными переменными X_1 и X_2 является симметричной, поэтому $\text{cov}(X_1, X_2) = \text{cov}(X_2, X_1)$. Это свойство происходит из определения ковариации и не зависит от конкретных значений переменных. Таким образом, порядок указания переменных в ковариации не имеет значения, и результат будет одинаковым независимо от того, какую пару переменных мы рассматриваем.

Собственные векторы матрицы - это векторы, которые при умножении на эту матрицу остаются коллинеарными (линейно зависимыми) сами себе, изменяясь только в масштабе.

Визуальное представление о собственных векторах и значениях представлено на рисунке 1.10.

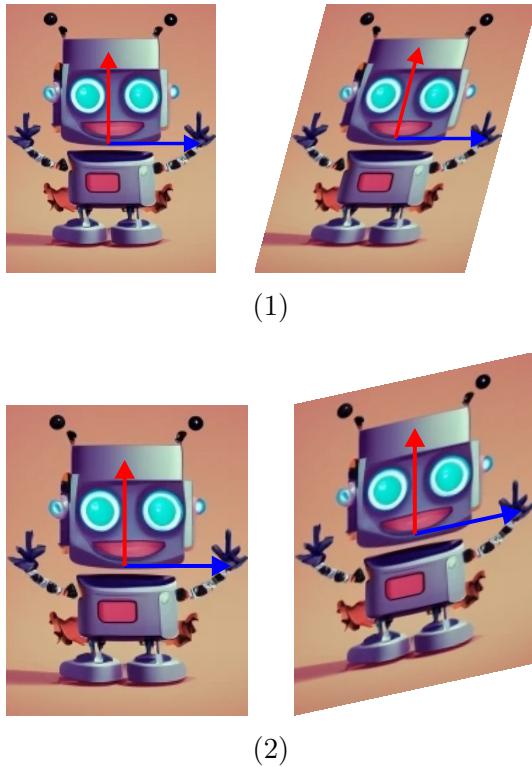


Рисунок 1.10: Визуальное представление о собственных векторах и значениях

Формально, собственные векторы определяются как ненулевые векторы \mathbf{v} , для которых выполняется следующее уравнение:

$$\mathbf{A} \cdot \mathbf{v} = \lambda \mathbf{v},$$

где \mathbf{A} - исходная матрица, λ - собственное значение (скаляр), а \mathbf{v} - собственный вектор.

Таким образом, умножение матрицы \mathbf{A} на собственный вектор \mathbf{v} приводит к получению нового вектора, который параллелен \mathbf{v} и масштабирован в λ раз. Собственные векторы матрицы играют важную роль в линейной алгебре. Они позволяют анализировать свойства и поведение матрицы, например, определять направления наибольшей и наименьшей изменчивости (главные компоненты) или решать системы линейных дифференциальных уравнений.

Собственные значения матрицы - это значения, которые удовлетворяют уравнению $\mathbf{A} \cdot \mathbf{v} = \lambda \mathbf{v}$, где \mathbf{A} - исходная матрица, λ - собственное значение (скаляр), а \mathbf{v} - собственный вектор.

Иными словами, собственные значения матрицы являются корнями уравнения $\det(\mathbf{A} - \lambda \mathbf{I}) = 0$, где \mathbf{I} - единичная матрица, а $\det(\cdot)$ обозначает определитель матрицы.

Собственный вектор, обозначенный синим цветом на рисунке 1.10.1, остается параллельным самому себе при деформации или преобразовании. Это означает, что его направление не меняется, а только масштабируется (его длина изменяется). Такой вектор является собственным вектором преобразования, соответствующим определенному собственному значению λ (в данном случае $\lambda = 1$), так как его направление сохраняется. Важно отметить, что любой вектор, параллельный синему собственному вектору, также будет собственным вектором с собственным значением λ .

Синий вектор меняет направление при деформации или преобразовании, в то время как красный вектор сохраняет свое направление. Поэтому красный вектор является собственным вектором, а синий - нет. Красный вектор не растягивается и не сжимается, его длина остается неизменной, поэтому соответствующее собственное значение равно единице, как показано на рисунке 1.10.2.

Собственные значения и собственные векторы играют важную роль в различных областях, таких как линейная алгебра, теория графов, физика, машинное обучение и другие. Они позволяют анализировать свойства и поведение матрицы, а также решать различные задачи, связанные с линейными операциями над данными.

Метод главных компонент и сингулярное разложение матриц (англ. Singular Value Decomposition, SVD) - это два связанных понятия в анализе данных и линейной алгебре. SVD - это разложение матрицы на три компонента: U , Σ и V . SVD широко используется в линейной алгебре и анализе данных. SVD может быть применено к матрице данных, где каждая строка представляет собой наблюдение, а каждый столбец - признак. Разложение SVD может быть использовано для реализации PCA.

Связь между PCA и SVD состоит в том, что PCA может быть реализован с использованием SVD разложения. Если X - это матрица данных, то SVD разложение X даёт U , Σ и V . Главные компоненты, полученные с помощью PCA, могут быть вычислены как линейная комбинация столбцов матрицы U , связанной с SVD разложением. Σ содержит сингулярные значения, которые представляют собой важность каждой главной компоненты.

Таким образом, SVD является математическим инструментом, который может быть использован для вычисления PCA и уменьшения размерности данных.

1.5.1 Визуальная демонстрация алгоритма

Объекты в наборах данных могут быть представлены разным количеством признаков. Мы можем визуализировать только те наборы данных, в которых не более 3 признаков (Рисунок 1.11).

Алгоритм PCA позволяет снизить количество признаков для визуализации или снижение размера датасета. В общем случае, количество признаков может быть снижено весьма существенно (например, для визуализации набора данных MNIST происходит снижение размерности с 784 до 2), однако в учебных целях мы будем преобразовывать данные в двумерном пространстве, произведя центрирование данных и смену осей.

Визуальная демонстрация алгоритма PCA может быть представлена в виде набора следующих шагов. Сначала мы читаем текущие данные и находим средние значения по каждому признаку, получая таким образом новый центр координат (Рисунок 1.12).

Затем мы перемещаем точку начала координат в этот новый усредненный центр нашего набора данных (Рисунок 1.13). Взаимное расположение объектов при этом сохраняется.

Затем мы находим линию (на рисунке обозначена красным пунктиром), проходящую через новый центр координат, сумма расстояний от точек данных до которой минимальна. Для

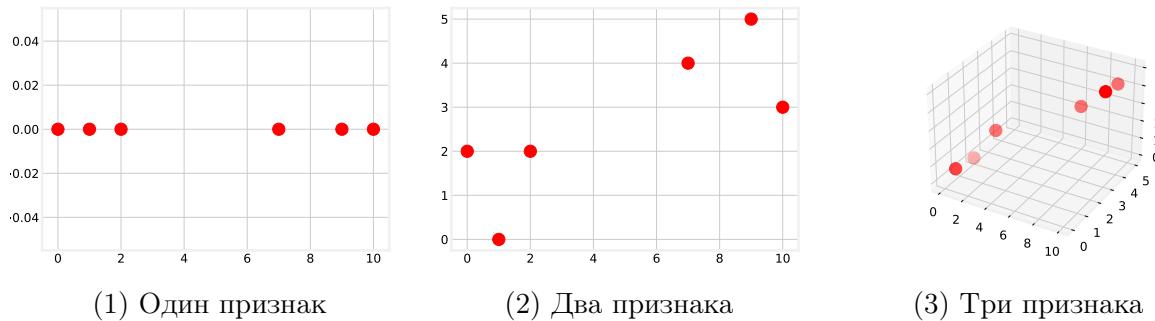


Рисунок 1.11: Разное количество признаков в датасете

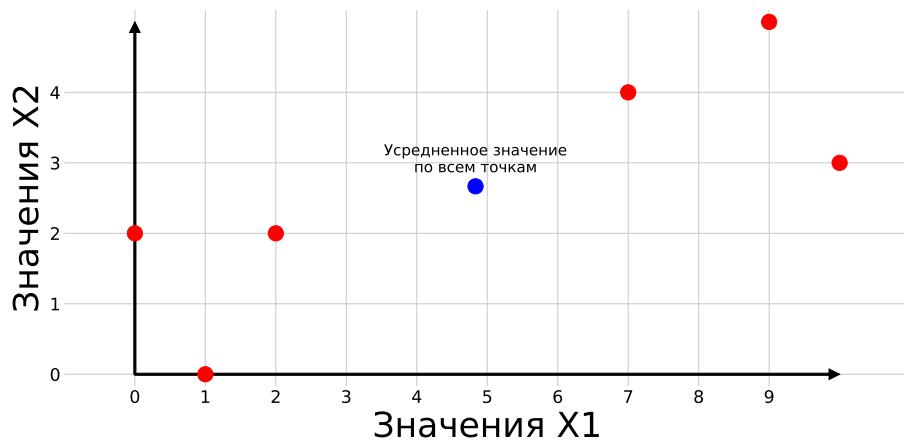


Рисунок 1.12: Нахождение среднего значения по признакам

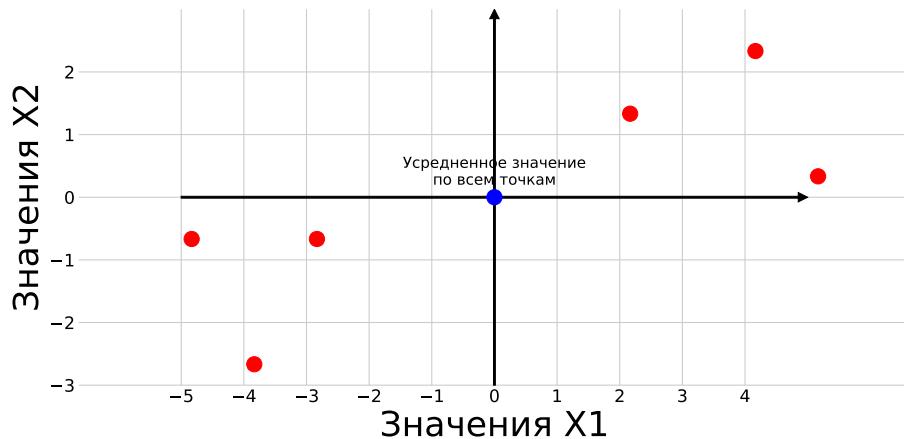


Рисунок 1.13: Центрирование данных

наглядности используется итеративный алгоритм (Рисунок 1.14), на практике используются другие техники, например, ковариационная матрица.

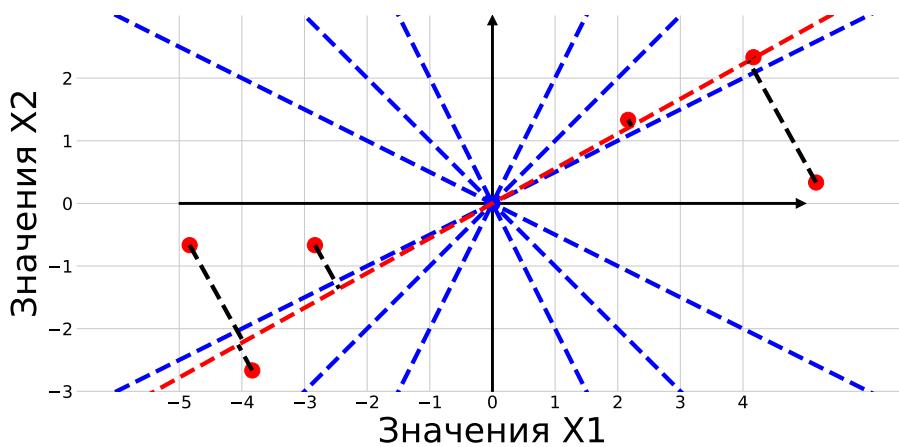


Рисунок 1.14: Нахождение линии, сумма расстояний от точек данных до которой минимальна

После этого, зная коэффициент наклона найденной прямой, находится линейная комбинация исходных признаков, которая определяет вектор, сонаправленный найденной линии, сумма расстояний от точек данных до которой минимальна. Нормировав все элементы линейной комбинации на длину этого вектора, мы получим собственный вектор первой главной компоненты длины 1 (на рисунке 1.15 синего цвета).

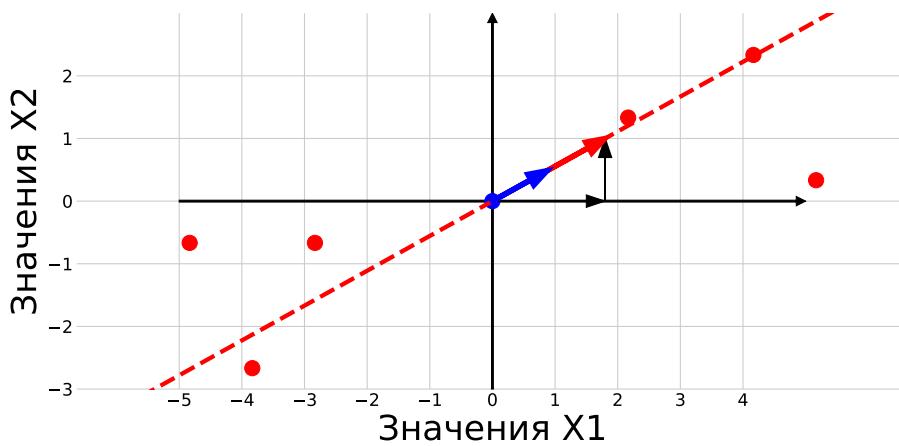


Рисунок 1.15: Нахождение линии, сумма расстояний от точек данных до которой минимальна

Также можно найти собственные значения главной компоненты. Каждое собственное значение соответствует одной главной компоненте и показывает, какую часть дисперсии в исходных данных объясняет соответствующая компонента. Сумма всех собственных значений равна общей дисперсии исходных данных.

Собственные значения используются для выбора количества главных компонент, которые нужно оставить после снижения размерности с помощью РСА. Чем больше собственное значение, тем больше доля дисперсии объясняется соответствующей главной компонентой. Обычно выбирают те главные компоненты, которые объясняют большую часть общей дисперсии (например, 95% или 99%) и игнорируют остальные компоненты, чтобы сократить размерность данных и сохранить наиболее информативные характеристики.

В наглядном примере формула для вычисления собственного значения главной компоненты равна:

$$\lambda = \frac{\sum_{i=1}^n \text{расстояние}^2 \text{ от } i\text{-го примера до линии главной компоненты}}{n - 1}$$

где n - количество объектов в наборе данных.

На практике формула для вычисления собственных значений в методе главных компонент (PCA) основана на ковариационной матрице данных.

Пусть C - ковариационная матрица размерности $d \times d$ для исходных данных, где d - количество признаков.

Собственные значения λ можно получить как корни характеристического уравнения:

$$\det(C - \lambda I) = 0$$

где I - единичная матрица размерности $d \times d$.

Решив это уравнение, можно получить собственные значения $\lambda_1, \lambda_2, \dots, \lambda_d$.

Обратите внимание, что собственные значения обычно упорядочиваются по убыванию, что позволяет выбрать наиболее информативные главные компоненты, которые объясняют наибольшую часть дисперсии в данных.

Вторая главная компонента будет перпендикулярна первой. Аналогично, зная угловой коэффициент линии, перпендикулярной первой главной компоненте, мы можем найти линейные комбинации признаков, которые формируют вектор второй главной компоненты. Отнормировав все элементы линейной комбинации на длину полученного вектора, мы получим собственный вектор второй главной компоненты (Рисунок 1.16).

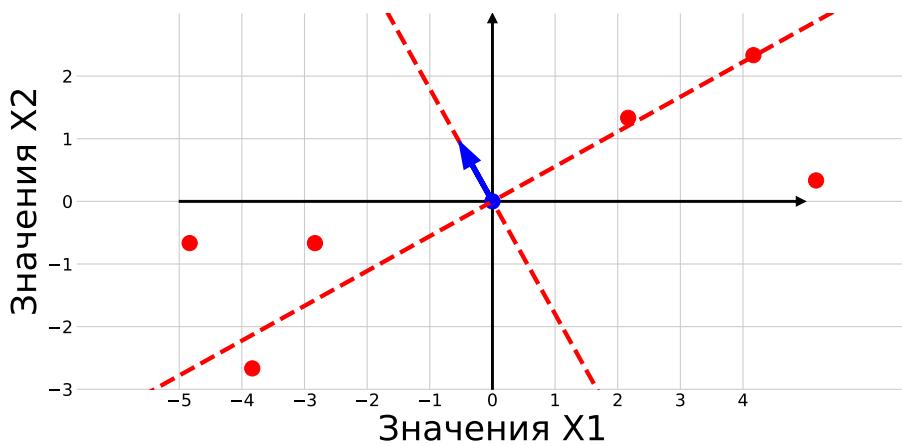


Рисунок 1.16: Нахождение собственного вектора второй главной компоненты

Если есть необходимость получить дальнейшие компоненты, процесс их поиска будет идти аналогичным образом.

В нашем случае мы остановимся после нахождения первых двух главных компонент. После вычисления этих компонент необходимо повернуть график таким образом, чтобы первая главная компонента стала новой осью обсцисс (осью X) (Рисунок 1.17). Соответственно, главные компоненты 1 и 2 становятся новыми осями координат.

Необходимо помнить, что каждая следующая главная компонента будет ортогональной предыдущим главным компонентам (Рисунок 1.18).

Также необходимо помнить, что чем выше собственные значения первых главных компонент, тем лучше они сохраняют информацию об исходном датасете (Рисунок 1.19). Если доля

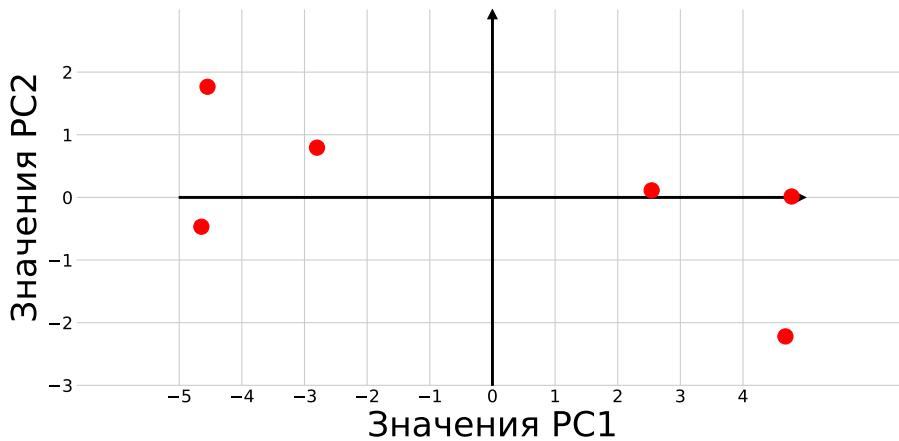


Рисунок 1.17: Поворот осей относительно начала координат

объясненной дисперсии среди первых компонент недостаточно высока, это может привести к снижения качества признакового описания, полученного в результате работы PCA.

1.5.2 Подготовка данных для алгоритма

Ниже представлен общий процесс подготовки данных для использования с методом главных компонент (PCA):

- Масштабирование данных: Рекомендуется масштабировать признаки входных данных перед применением PCA, особенно если признаки имеют различные шкалы или единицы измерения. Обычно используется стандартизация или нормализация данных для приведения их к общей шкале.
- Обработка категориальных переменных: Если у вас есть категориальные переменные в данных, требуется их преобразование в числовую формат. Вы можете использовать методы, такие как One-Hot Encoding или Label Encoding, чтобы закодировать категориальные переменные перед применением PCA.
- Обработка пропущенных значений: Если в ваших данных есть пропущенные значения, нужно решить, как с ними поступить. Вы можете удалить строки или столбцы с пропущенными значениями или заполнить их средними или медианными значениями в зависимости от контекста данных.

1.5.3 Процесс обучения

Шаги обучения метода главных компонент (PCA) с использованием сингулярного разложения (SVD) включают следующие действия:

1. Подготовка данных: Подготовьте данные, убедитесь, что они числовые и центрированы (имеют среднее значение равное нулю).
2. Вычисление матрицы ковариации: Вычислите матрицу ковариации для центрированных данных. Матрица ковариации имеет размерность $d \times d$, где d - количество признаков.

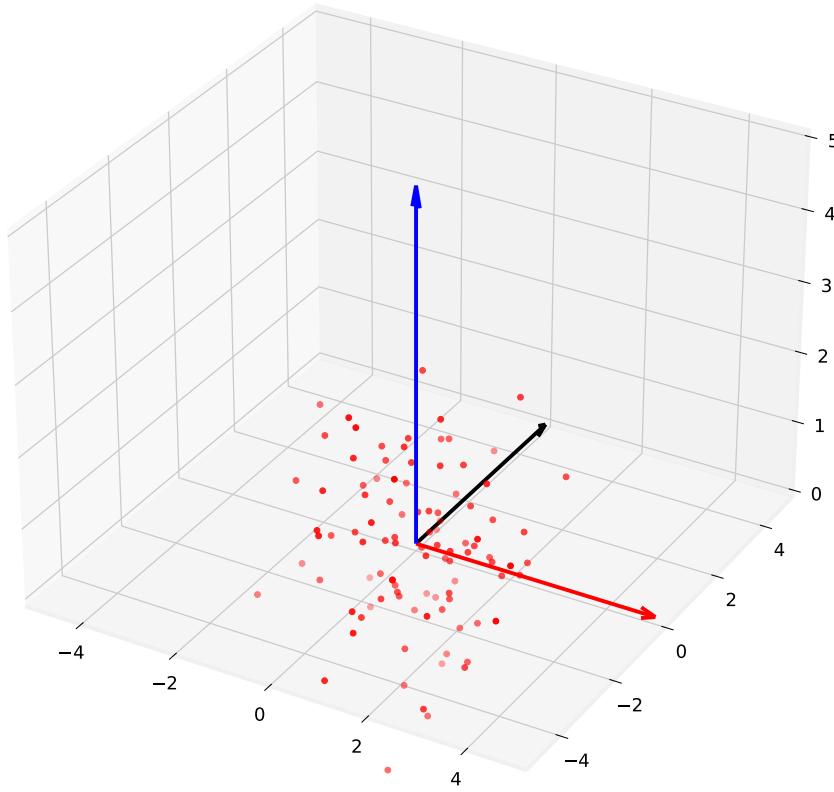


Рисунок 1.18: Ортогональность (перпендикулярность главных компонент)

3. SVD разложение: Примените сингулярное разложение (SVD) к матрице ковариации. SVD разложение представляет матрицу как произведение трех матриц: $C = USV^T$, где U - ортогональная матрица размерности $d \times d$, S - диагональная матрица размерности $d \times d$ с сингулярными значениями на диагонали, и V - ортогональная матрица размерности $d \times d$.
4. Выбор компонент: Определите, сколько главных компонент нужно оставить, выбирая сингулярные значения, которые объясняют большую часть общей дисперсии данных. Это можно сделать, например, с помощью критерия сохранения определенного процента дисперсии (например, 95% или 99%).
5. Проецирование на новое пространство: Проецируйте центрированные данные на пространство главных компонент. Для этого умножьте матрицу данных на матрицу главных компонент U_k размерности $d \times k$, где k - количество выбранных главных компонент.

1.5.4 Оценка качества алгоритма

Оценка качества алгоритма РСА (метода главных компонент) может быть выполнена с использованием нескольких метрик и методов. Некоторые из них включают:

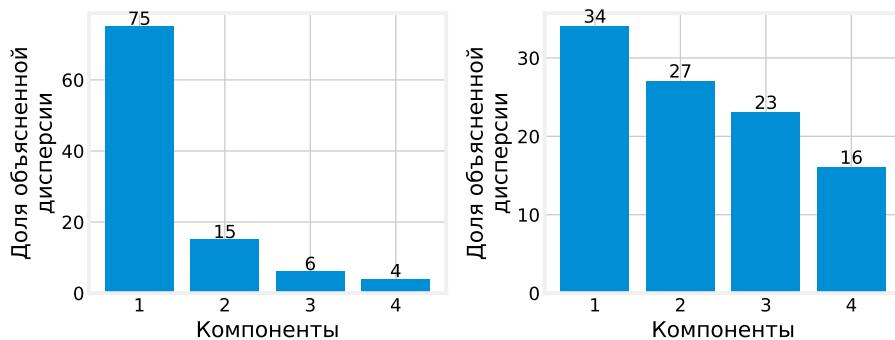


Рисунок 1.19: Высокая доля объясненной дисперсии среди первых главных компонент (график слева) и относительно низкая доля объясненной дисперсии среди первых главных компонент (график справа)

- Объясненная дисперсия: Оценка объясненной дисперсии позволяет определить, какую часть общей дисперсии данных объясняют выбранные главные компоненты. Эта метрика позволяет оценить, насколько успешно PCA сжимает информацию о данных.
- Кумулятивная объясненная дисперсия: Кумулятивная объясненная дисперсия показывает, сколько общей дисперсии данных объяснено суммарно выбранными главными компонентами. Она может быть полезна для определения оптимального числа главных компонент для использования.
- Восстановление данных: Оценка качества алгоритма PCA может быть выполнена путем восстановления данных из пространства главных компонент и сравнения восстановленных данных с исходными. Можно использовать различные метрики, такие как среднеквадратическая ошибка (MSE) или средняя абсолютная ошибка (MAE), чтобы оценить точность восстановления данных.
- Визуализация: Визуальное представление главных компонент и их влияния на данные может помочь в оценке качества алгоритма PCA. Построение графиков, например, графика объясненной дисперсии по числу главных компонент или графика сжатых данных в пространстве главных компонент, может помочь понять эффективность PCA.
- Задача: Наконец, оценка качества алгоритма PCA может быть выполнена с учетом конкретной задачи или цели. Например, в задаче классификации можно оценить, как PCA влияет на точность классификации модели, или какие признаки являются наиболее информативными после применения PCA.

1.5.5 Применение алгоритма

Алгоритм PCA имеет широкий спектр применений и может быть использован во многих областях. Вот некоторые из них:

- Снижение размерности данных: Одним из основных применений PCA является снижение размерности данных. Это позволяет представить многомерные данные в более компактной форме, удалив менее информативные признаки и выделив наиболее важные компоненты. Снижение размерности может быть полезно для улучшения эффективности вычислений, улучшения визуализации данных и борьбы с проклятием размерности.

- Визуализация данных: PCA может использоваться для визуализации данных в двух или трех измерениях. Применение PCA к исходным данным позволяет преобразовать их в новое пространство с меньшей размерностью, где можно визуализировать данные на плоскости или в пространстве. Это может помочь выявить структуру и зависимости между объектами данных.
- Удаление корреляций: PCA может использоваться для удаления или уменьшения корреляции между признаками в данных. Это может быть полезно в случаях, когда сильная корреляция между признаками мешает моделированию или приводит к мультиколлинеарности.
- Анализ главных компонент: PCA позволяет выделить главные компоненты, которые объясняют наибольшую долю дисперсии в данных. Это может помочь идентифицировать наиболее важные факторы или признаки, влияющие на данные, и понять их вклад в общую вариацию.
- Сжатие данных: PCA может использоваться для сжатия данных, сохраняя важную информацию и уменьшая объем памяти, необходимый для хранения данных. Это может быть полезно в задачах с ограниченными ресурсами или при передаче данных по сети.
- Предобработка данных: PCA может быть использован для предварительной обработки данных перед применением других алгоритмов машинного обучения. Он может помочь улучшить производительность моделей и избежать проблемы мультиколлинеарности.

Области применения PCA включают финансы, экономику, биоинформатику, обработку изображений, распознавание образов, геологию, маркетинг и многие другие.

1.5.6 Плюсы и минусы алгоритма

Плюсы

- Снижение размерности: Одним из основных преимуществ PCA является его способность снижать размерность данных. Он позволяет представить многомерные данные в пространстве меньшей размерности, сохранив при этом наибольшую долю информации. Это позволяет улучшить эффективность вычислений, уменьшить сложность моделей и улучшить визуализацию данных.
- Выделение наиболее информативных признаков: PCA позволяет определить наиболее важные компоненты данных, которые объясняют наибольшую долю вариации. Это помогает выделить наиболее информативные признаки и факторы, которые влияют на данные. Таким образом, PCA может быть использован для отбора признаков и уменьшения размерности без потери существенной информации.
- Устранение корреляций: PCA может использоваться для устранения или уменьшения корреляций между признаками. Это особенно полезно, когда сильная корреляция между признаками мешает моделированию или приводит к проблемам мультиколлинеарности.
- Визуализация данных: Применение PCA позволяет визуализировать данные в пространстве с меньшей размерностью. Это может помочь визуально исследовать структуру данных, выявить паттерны и зависимости, а также обнаружить аномалии или выбросы.

- Сжатие данных: PCA может быть использован для сжатия данных, уменьшая объем памяти, необходимый для хранения данных. При этом сохраняется основная структура и вариация данных. Сжатие данных может быть особенно полезным в задачах с ограниченными ресурсами или при передаче данных по сети.
- Предобработка данных: PCA может использоваться в качестве предварительной обработки данных перед применением других алгоритмов машинного обучения. Он может помочь улучшить производительность моделей, уменьшить влияние шума и избежать проблемы мультиколлинеарности.

Минусы

- Потеря интерпретируемости: При снижении размерности данных с помощью PCA, исходные признаки объединяются в новые компоненты, которые являются линейными комбинациями исходных признаков. Это может привести к потере интерпретируемости, поскольку новые компоненты не всегда имеют прямую связь с исходными признаками.
- Зависимость от линейности: PCA предполагает линейные зависимости между признаками. Если данные имеют сложные нелинейные зависимости, PCA может быть неэффективным и не удастся захватить важную информацию, содержащуюся в данных.
- Чувствительность к выбросам: Алгоритм PCA может быть чувствительным к выбросам в данных. Выбросы могут сильно влиять на главные компоненты и искажать результаты анализа.
- Расчет вычислительно сложен: Вычисление собственных значений и собственных векторов для больших наборов данных может быть вычислительно сложной задачей. Это особенно актуально, когда количество признаков или объектов велико.
- Не учитывает контекст и доменные знания: PCA основывается на математических принципах и не учитывает контекст и доменные знания данных. Это может ограничивать его способность уловить специфические характеристики и зависимости в данных.

1.5.7 Реализация алгоритма в Python

PCA (Principal Component Analysis) в библиотеке scikit-learn реализована через класс PCA из модуля sklearn.decomposition. Основные параметры модели PCA:

- **n_components:** количество главных компонент, которые требуется извлечь из данных. Это обязательный параметр.
- **svd_solver:** метод, используемый для вычисления PCA. Может принимать значения «auto», «full», «arpack» или «randomized». По умолчанию используется «auto», который выбирает наиболее подходящий метод автоматически.

Класс PCA также имеет методы fit(X) для обучения модели на данных X, transform(X) для преобразования данных X с использованием изученных компонент, и fit_transform(X) для комбинированного обучения и преобразования данных.

1.5.8 Вопросы для самопроверки

Какое утверждение о PCA верно (один ответ)?

1. PCA используется только для уменьшения размерности данных.
2. PCA используется только для визуализации данных.

3. РСА позволяет уменьшить размерность данных и сохранить наибольшую долю информации.
4. РСА применяется только к категориальным данным.

Правильные ответы: 3

Какие компоненты в РСА имеют наибольшее собственное значение?

1. Последние компоненты.
2. Произвольные компоненты.
3. Первые компоненты.
4. Компоненты с нулевым собственным значением.

Правильные ответы: 3

Какие преимуществами обладает РСА (шесть ответов)?

- Визуализация данных
- Выделение наиболее информативных признаков
- Зависимость от линейности
- Потеря интерпретируемости
- Предобработка данных
- Расчет вычислительно сложен
- Сжатие данных
- Снижение размерности
- Устранение корреляций
- Чувствительность к выбросам

Правильные ответы: 1, 2, 5, 7, 8, 9

1.5.9 Резюме по разделу

PCA (Principal Component Analysis) - это метод анализа данных, используемый для уменьшения размерности исходного набора данных. Основная идея РСА заключается в поиске новых независимых переменных, называемых главными компонентами, которые представляют наибольшую долю вариации в данных.

Основные преимущества РСА:

- Уменьшение размерности данных, что упрощает анализ и визуализацию.
- Сокращение шума и выбросов в данных.
- Сохранение наибольшей доли информации при уменьшении размерности данных.

РСА является мощным инструментом для анализа данных и широко применяется в области машинного обучения, статистики, визуализации данных и других областях, где требуется снижение размерности данных и выделение наиболее информативных признаков.

1.6 t-SNE

Цель занятия: ученик может применить алгоритм t-SNE для решения задач снижения размерности на подготовленных и неподготовленных данных.

План занятия:

- Визуальная демонстрация алгоритма
- Подготовка данных для алгоритма
- Процесс обучения
- Оценка качества алгоритма
- Применение алгоритма
- Плюсы и минусы алгоритма
- Реализация алгоритма в Python

t-SNE (t-Distributed Stochastic Neighbor Embedding) — это алгоритм машинного обучения, используемый для визуализации и снижения размерности данных. Он был разработан для отображения сложных многомерных данных в двух- или трехмерное пространство, сохраняя при этом сходства между точками.

Основная идея t-SNE заключается в том, чтобы представить исходные данные в новом пространстве, где близкие объекты остаются близкими, а далекие объекты разделяются. Алгоритм строит вероятностное распределение для пар объектов в исходном пространстве и в новом пространстве таким образом, чтобы минимизировать разницу между ними. Также t-SNE позволяет учитывать локальные и глобальные структуры данных.

t-SNE широко используется в визуализации данных, особенно при работе с высокоразмерными и сложными наборами данных, такими как изображения или тексты. Он помогает обнаруживать скрытые паттерны, кластеры и взаимосвязи между объектами в данных.

Перед тем, как рассмотреть алгоритм t-SNE, необходимо освежить в памяти некоторые понятия. Нормальное распределение, также известное как распределение Гаусса, является одним из наиболее распространенных и важных вероятностных распределений. Оно характеризуется плотностью вероятности, которая имеет форму колокола или симметричного колоколообразного графика (Рисунок 1.20).



Рисунок 1.20: График функции плотности вероятности нормального распределения

Формула плотности вероятности нормального распределения выглядит следующим образом:

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

где: x - случайная переменная, μ - математическое ожидание (среднее значение) распределения, σ^2 - дисперсия (квадрат стандартного отклонения) распределения.

В этой формуле параметры μ и σ^2 определяют положение и форму колокола нормального распределения. Математическое ожидание μ указывает на центр колокола, а дисперсия σ^2 определяет его ширину. Распределение Стьюдента (t -распределение) - это вероятностное распределение, которое широко используется при оценке параметров и проведении статистических тестов в малых выборках, когда исходная генеральная совокупность имеет нормальное распределение, но значение дисперсии неизвестно.

Распределение Стьюдента с параметром ν , обозначаемое как $t(\nu)$, имеет форму колокола с симметричным графиком вокруг нуля. Число степеней свободы ν определяет форму распределения и влияет на его хвостатость. Чем больше ν , тем ближе распределение Стьюдента приближается к стандартному нормальному распределению. (Рисунок 1.21).



Рисунок 1.21: График функции плотности вероятности распределения Стьюдента

Формула плотности вероятности распределения Стьюдента выглядит следующим образом:

$$f(x) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi\Gamma\left(\frac{\nu}{2}\right)}} \left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

где: x - случайная переменная, ν - число степеней свободы (degrees of freedom) распределения, Γ - функция Гамма, которая вычисляется для соответствующих аргументов.

Обратите внимание, что для вычисления плотности вероятности распределения Стьюдента требуется использование функции Гамма, которая может быть определена отдельно.

Гауссовское ядро (или ядро Гаусса) - это одно из распространенных ядерных функций, используемых в алгоритмах машинного обучения и обработки сигналов. Оно часто применяется в задачах сглаживания и фильтрации данных. Гауссовское ядро представляет собой функцию, которая зависит от расстояния между двумя точками и имеет форму колокола с пиком в нуле. Оно симметрично и уменьшается с расстоянием от пика. Гауссовское ядро используется для вычисления весовых коэффициентов в методе ядерного сглаживания (kernel smoothing). В этом методе, каждая точка данных взвешивается с помощью функции

гауссовского ядра в зависимости от её расстояния от интересующей нас точки. Ближние точки получают больший вес, тогда как дальние точки получают меньший вес (Рисунок 1.22). Формула Гауссовского ядра выглядит следующим образом:



Рисунок 1.22: Гауссовское ядро

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

где: - x и x' - две точки, между которыми мы вычисляем ядро, - $\|x - x'\|$ - расстояние между точками x и x' , - σ - параметр ширины ядра (стандартное отклонение).

В формуле используется евклидово расстояние между точками x и x' , и оно возведено в квадрат. Параметр σ определяет ширину ядра и влияет на «размытость» и «распределение массы» вокруг каждой точки.

Гауссовское ядро широко применяется в алгоритмах машинного обучения, таких как метод опорных векторов (SVM) и метод гауссовых процессов (Gaussian Processes), а также в различных методах сглаживания и фильтрации данных. Дивергенция Кульбака-Лейблера (Kullback-Leibler divergence), также известная как относительная энтропия, является мерой различия между двумя вероятностными распределениями. Она широко применяется в области информационной теории и статистики для измерения расстояния между двумя распределениями. Дивергенция Кульбака-Лейблера между двумя вероятностными распределениями P и Q измеряет, насколько сильно распределение P отличается от распределения Q . Она не является метрикой расстояния, так как она не обладает свойством симметричности, то есть KL -дивергенция между распределением P и распределением Q не равна KL -дивергенции между распределением Q и распределением P .

Формула дивергенции Кульбака-Лейблера выглядит следующим образом:

$$D_{KL}(P||Q) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right)$$

где: - D_{KL} - дивергенция Кульбака-Лейблера, - $P(i)$ и $Q(i)$ - вероятности событий i в распределениях P и Q соответственно, - \log - натуральный логарифм.

Обратите внимание, что дивергенция Кульбака-Лейблера может быть определена только в случае, когда все значения $P(i)$ равны нулю, если $Q(i)$ равно нулю.

Дивергенция Кульбака-Лейблера является положительной и неотрицательной, и равна нулю только в случае, когда два распределения P и Q совпадают.

1.6.1 Визуальная демонстрация алгоритма

Визуальная демонстрация алгоритма t-SNE представлена на рисунке ???. Необходимо загрузить в память исходную выборку с большим количеством признаков и случайным образом сгенерировать её отображение в признаковое пространство меньшей размерности (Рисунок 1.23).

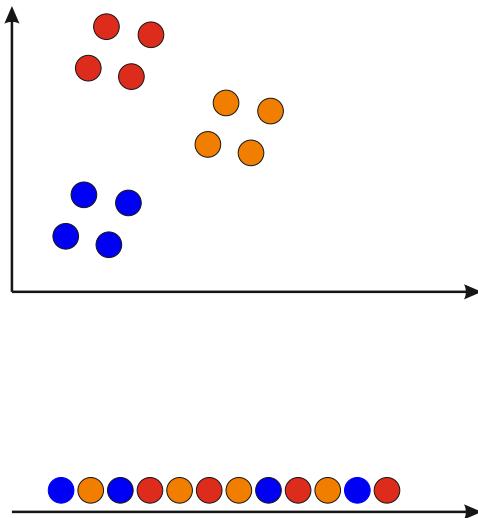


Рисунок 1.23

Основная идея алгоритма t-SNE заключается в том, чтобы сохранить близость объектов из исходного пространства в результирующем пространстве (Рисунок 1.24).

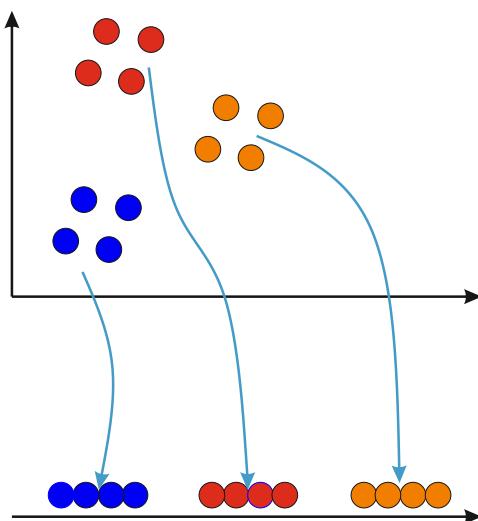


Рисунок 1.24

Сначала вычисляется попарная сходственность (affinity) между всеми парами объектов в исходном пространстве. Обычно используется гауссово ядро (гауссовое распределение) для оценки сходства между объектами. Сходство выражается числовыми значениями, которые показывают степень близости между объектами. Затем вычисляются условные вероятности сходства между парами объектов на основе попарных сходств. Эти вероятности вычисляются как отношение попарных сходств к сумме всех сходств, взятых с учетом попарных расстояний между объектами (Рисунок 1.25).

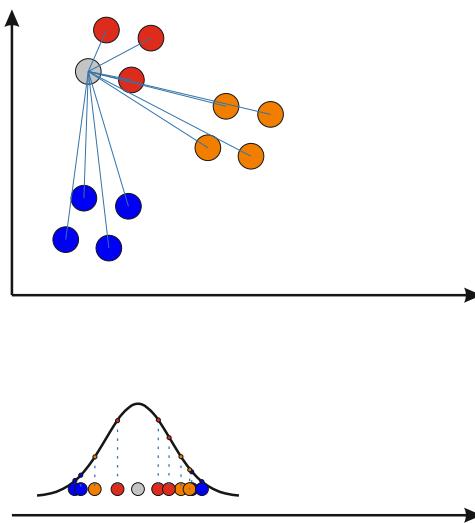


Рисунок 1.25

Симметричность расстояний от точки i до точки j и перплексия - два важных фактора для правильного функционирования алгоритма t-SNE и получения высококачественных вложений. Симметричность расстояний для метрики означает, что расстояние от точки i до точки j и наоборот равны. Изначально в дивергенции Кульбака-Лейблера это не так. В алгоритме t-SNE производится усреднение расстояний от точки i до точки j и наоборот (то есть симметризация расстояний) для достижения более устойчивых и сбалансированных вложений в новом пространстве. Перплексия (perplexity) в теории информации - это мера сложности или неопределенности распределения вероятностей. Она используется в различных контекстах, включая машинное обучение и статистику.

Математически, перплексия определяется как экспонента энтропии распределения вероятностей. Для дискретного случая, перплексия вычисляется следующим образом:

$$\text{Perplexity}(P) = 2^{-\sum_i P(i) \log_2 P(i)}$$

где P - распределение вероятностей.

Для непрерывного случая, перплексия может быть определена как:

$$\text{Perplexity}(p(x)) = \exp \left(- \int p(x) \log_2 p(x) dx \right)$$

где $p(x)$ - плотность вероятности.

Перплексия в алгоритме t-SNE используется для контроля числа ближайших соседей, которые учитываются при вычислении условных вероятностей сходства. Высокое значение перплексии увеличивает влияние ближайших соседей, тогда как низкое значение перплексии распространяет влияние на более широкий набор объектов. Выбор оптимального значения перплексии зависит от особенностей данных и требуемой структуры вложений. Подбор правильной перплексии позволяет достичь более точного отображения локальных сходств и сохранения структуры данных в новом пространстве (Рисунок 1.26).

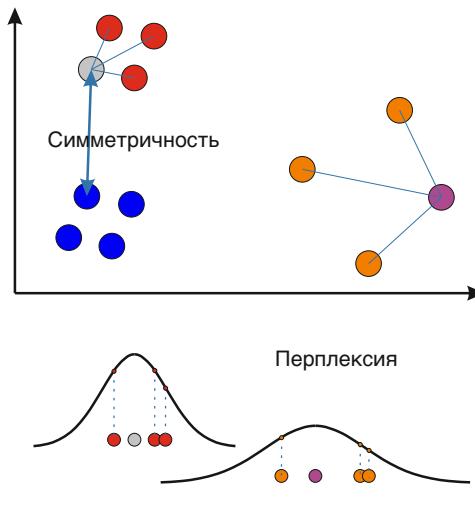


Рисунок 1.26

Симметричность расстояний и перплексия взаимодействуют, чтобы учесть локальные отношения между объектами и сохранить структуру данных в новом пространстве.

На каждой итерации алгоритма t-SNE вычисляется условная вероятность сходства объектов. Для каждой пары объектов i и j вычисляется условная вероятность p_{ij} , которая отражает вероятность выбрать объект j как соседа объекта i при условии их исходного расположения в пространстве высокой размерности. Затем эти условные вероятности p_{ij} сохраняются в матрице P , где элемент P_{ij} представляет собой условную вероятность сходства между объектами i и j .

Также, на следующем шаге текущей итерации алгоритма t-SNE, вычисляется условная вероятность в новом пространстве низкой размерности. Для каждой пары объектов в новом пространстве i' и j' , вычисляется условная вероятность q_{ij} , которая отражает вероятность выбрать объект j' как соседа объекта i' . Эти условные вероятности q_{ij} также сохраняются в матрице Q . Сохранение вероятностей сходства в матрицах P и Q является важным для дальнейшего вычисления расстояний и оптимизации вложений в пространстве низкой размерности.

Далее происходит оптимизация расположения объектов в результирующем пространстве. Алгоритм стремится минимизировать расхождение между условными вероятностями сходства в исходном пространстве и результирующем пространстве с помощью дивергенции Кульбака-Лейблера. Оптимизация выполняется с использованием метода градиентного спуска. В каждой итерации оптимизации происходит обновление вложений (координат) объ-

ектов в результирующем пространстве. Обновление выполняется путем расчета градиента и изменения координат объектов в направлении, которое минимизирует расхождение между вероятностями сходства объектов, сохраненными в соответствующих матрицах. Алгоритм повторяется до достижения сходимости. Обычно определенное число итераций задается заранее или алгоритм останавливается, когда изменение вложений становится незначительным. В результате работы алгоритма t-SNE получается новое представление данных в пространстве меньшей размерности, которое учитывает локальные сходства между объектами. Это позволяет визуализировать данные и обнаруживать структуры и паттерны, которые могут быть неочевидны в исходном пространстве высокой размерности (Рисунок 1.27).

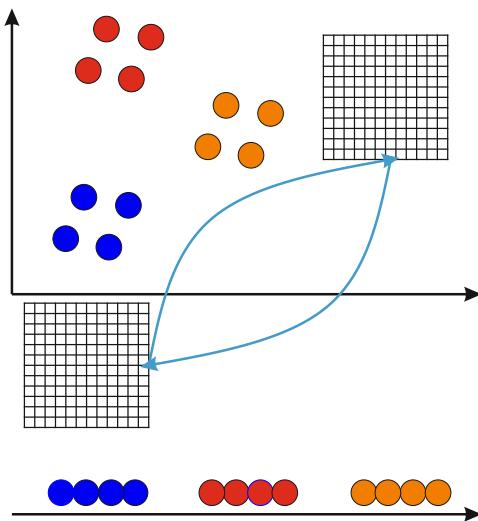


Рисунок 1.27

1.6.2 Процесс обучения

Шаги алгоритма t-SNE:

1. Вычисление схожести: Сначала алгоритм вычисляет меру схожести между парами объектов в исходном пространстве. Обычно это делается с использованием гауссовского ядра или расстояния между точками, такого как евклидово расстояние.
2. Создание вероятностного распределения: Для каждой точки в исходном пространстве алгоритм вычисляет вероятностное распределение, отражающее схожесть между этой точкой и другими точками. Более похожим точкам присваивается более высокая вероятность.
3. Вычисление схожести в пространстве назначения: Затем алгоритм переходит в пространство назначения (обычно двух- или трехмерное), где он создает аналогичное вероятностное распределение. Он старается разместить точки таким образом, чтобы более похожие точки были ближе друг к другу.
4. Минимизация дивергенции Кульбака-Лейблера: Целью t-SNE является минимизация дивергенции Кульбака-Лейблера между вероятностными распределениями в исходном

пространстве и пространстве назначения. Это достигается путем настройки позиций точек в пространстве назначения таким образом, чтобы минимизировать разницу между этими распределениями.

5. Градиентный спуск: Для минимизации дивергенции Кульбака-Лейблера алгоритм использует градиентный спуск. Он вычисляет градиент функционала ошибки и обновляет позиции точек, чтобы уменьшить ошибку. Обновление происходит итеративно, пока не будет достигнута сходимость.

1.6.3 Оценка качества алгоритма

Для оценки качества работы алгоритма t-SNE можно использовать несколько метрик. Вот некоторые из них:

- Визуализация: Одним из наиболее популярных способов оценки качества t-SNE является визуальный анализ результатов. Просмотрите полученное низкоразмерное представление данных и оцените, насколько хорошо алгоритм сохраняет структуру и относительные расстояния между объектами. Убедитесь, что близкие объекты находятся близко друг к другу, а разные кластеры отделены друг от друга.
- Поддержка кластеризации: Если у вас есть информация о настоящих кластерах в данных, вы можете оценить, насколько хорошо t-SNE справляется с их выделением. Используйте метрики, такие как Adjusted Rand Index (ARI), Normalized Mutual Information (NMI) или Silhouette Score, чтобы сравнить полученные кластеры с эталонными.
- Сохранение расстояний: t-SNE должен сохранять относительные расстояния между объектами в исходном пространстве при их проецировании на низкоразмерное пространство. Вы можете вычислить попарные расстояния между объектами в исходном и низкоразмерном пространствах и сравнить их. Можно использовать метрики, такие как Pearson correlation coefficient или Spearman rank correlation coefficient для оценки сходства между расстояниями.
- Stochastic Neighbor Embedding (SNE) cost function: t-SNE оптимизирует функцию стоимости, основанную на SNE. Вы можете отслеживать значения функции стоимости на каждой итерации и проверить, сходится ли алгоритм. Если значения функции стоимости продолжают уменьшаться, это может быть индикатором успешной работы алгоритма.

Важно отметить, что t-SNE является эвристическим алгоритмом, и не существует универсальных метрик, которые идеально оценивают его качество. Оценка качества t-SNE должна основываться на комбинации различных метрик и визуальной интерпретации результатов.

1.6.4 Применение алгоритма

Алгоритм t-SNE является мощным инструментом для визуализации исходных данных в низкоразмерном пространстве. Он может использоваться в различных областях, где требуется визуализация и анализ многомерных данных. Вот некоторые примеры применения алгоритма t-SNE:

- Визуализация данных: Основное применение t-SNE заключается в визуализации сложных многомерных данных в двух или трех измерениях. Это позволяет исследовать и понять структуру данных, выявить кластеры или группы объектов, обнаружить выбросы

или аномалии и обнаружить скрытые паттерны или зависимости. Примеры включают визуализацию геномных данных, изображений, текстовых данных и т.д.

- Кластеризация: t-SNE может использоваться для кластеризации данных на основе их визуализации в низкоразмерном пространстве. После проецирования данных на плоскость с помощью t-SNE, можно применить алгоритмы кластеризации, такие как k-means или DBSCAN, для выделения кластеров и их анализа.
- Обнаружение аномалий: t-SNE может помочь в обнаружении аномалий или выбросов в данных. Объекты, которые отображаются далеко от основной структуры данных в низкоразмерном пространстве, могут считаться потенциальными аномалиями или интересными случаями, которые заслуживают дополнительного исследования.
- Предварительная обработка данных: t-SNE может использоваться как этап предварительной обработки данных для последующего применения других алгоритмов машинного обучения. Проецирование данных на низкоразмерное пространство с помощью t-SNE может улучшить эффективность и качество работы других алгоритмов, таких как классификация, регрессия или кластеризация.
-

Важно отметить, что t-SNE является вычислительно сложным алгоритмом, особенно для больших наборов данных. Поэтому для его применения к большим данным может потребоваться распараллеливание вычислений или использование других методов.

1.6.5 Плюсы и минусы алгоритма

Плюсы Алгоритм t-SNE имеет несколько преимуществ, которые делают его популярным инструментом для визуализации и анализа данных. Вот некоторые из плюсов алгоритма t-SNE:

- Сохранение локальной структуры: t-SNE стремится сохранить относительные расстояния между близкими объектами в исходном пространстве. Это означает, что объекты, которые находятся близко друг к другу в исходных данных, будут отображены близко друг к другу в низкоразмерном пространстве. Это позволяет сохранить локальную структуру данных и обнаружить кластеры или группы объектов.
- Устойчивость к выбросам: t-SNE обычно хорошо справляется с выбросами или аномалиями в данных. Из-за своего вероятностного подхода и использования тяжелых хвостов распределения t-Student, t-SNE не так сильно подвержен влиянию выбросов и может помочь обнаружить их.
- Визуальная интерпретация: Одним из основных преимуществ t-SNE является его способность визуализировать высокоразмерные данные в двух или трех измерениях. Это позволяет исследовать данные, выявлять паттерны и структуры, а также делать выводы и принимать решения на основе визуальной интерпретации результатов.

Минусы Наряду с преимуществами, алгоритм t-SNE также имеет некоторые ограничения и потенциальные недостатки. Вот некоторые из них:

- Вычислительная сложность: Алгоритм t-SNE требует вычисления попарных расстояний между всеми парами точек в исходном пространстве. Это делает его вычислительно требовательным, особенно для больших наборов данных. Время выполнения может значительно возрастать с увеличением размерности исходных данных или числа объектов.

- Недетерминированность: t-SNE является стохастическим алгоритмом, что означает, что результаты могут немного различаться при каждом запуске. Это может затруднить воспроизводимость результатов и требует выполнения нескольких запусков для проверки стабильности и надежности результатов.
- Потеря глобальной структуры: t-SNE сконцентрирован на сохранении локальной структуры данных и обнаружении кластеров. Однако, из-за своей природы, алгоритм может снижать различия между удаленными объектами в низкоразмерном пространстве, что может привести к потере глобальной структуры данных.

1.6.6 Реализация алгоритма в Python

В библиотеке scikit-learn, алгоритм t-SNE (t-Distributed Stochastic Neighbor Embedding) реализован через класс TSNE из модуля sklearn.manifold. Описание основных параметров класса TSNE:

- **n_components:** количество компонентов в низкоразмерном пространстве, в которое будет проецироваться исходное пространство данных. Обычно это 2 или 3 для визуализации. Это обязательный параметр.
- **perplexity:** мера сложности глобальной структуры данных. Определяет баланс между сохранением локальной и глобальной структуры. Значения perplexity должны быть в диапазоне от 5 до 50, но часто используется значение около 30.
- **learning_rate:** скорость обучения алгоритма. Определяет шаг изменения в низкоразмерном пространстве. Маленькое значение learning_rate может привести к более долгой сходимости, но лучшей качеству визуализации. Значения learning_rate обычно варьируются от 10 до 1000.
- **n_iter:** количество итераций оптимизации алгоритма. Определяет количество шагов, которые алгоритм выполняет для достижения оптимального результата. Значение по умолчанию - 1000.

В классе TSNE также имеются методы **fit(X)** для обучения модели на данных X и **transform(X)** для преобразования данных X в низкоразмерное пространство с использованием изученных компонент.

1.6.7 Вопросы для самопроверки

Какое утверждение о t-SNE верно (один ответ)?

1. t-SNE используется только для уменьшения размерности данных.
2. t-SNE используется только для визуализации данных.
3. t-SNE основывается на вероятностной модели, которая стремится сохранить схожесть точек в исходном пространстве при их проецировании в низкоразмерное пространство.
4. t-SNE применяется для задач классификации.

Правильные ответы: 3

Какие параметры нужно установить для работы t-SNE (два ответа)?

1. Число соседей исходных данных
2. Размерность входных данных

3. Число итераций алгоритма
4. perplexity и learning rate

Правильные ответы: 3, 4

Какие преимуществами обладает t-SNE (три ответа)?

- Визуальная интерпретация
- Вычислительная сложность
- Недетерминированность
- Потеря глобальной структуры
- Сохранение локальной структуры
- Устойчивость к выбросам

Правильные ответы: 1, 5, 6

1.6.8 Резюме по разделу

t-SNE (t-Distributed Stochastic Neighbor Embedding) - это алгоритм снижения размерности данных, который часто используется для отображения высокоразмерных данных на двух- или трехмерное пространство. Он широко применяется для анализа и визуализации сложных наборов данных, таких как текстовые, аудио, графовые и другие.

Основная идея алгоритма t-SNE заключается в том, чтобы сохранить близость объектов из исходного пространства в результирующем пространстве. Алгоритм строит вероятностную модель, где объекты в исходном пространстве и в пространстве визуализации представлены вероятностными распределениями. Он пытается минимизировать расхождение между этими распределениями, чтобы сохранить связи и соседство между объектами.

Для работы алгоритма t-SNE необходимо настроить параметры, такие как perplexity (параметр, определяющий баланс между сохранением локальной и глобальной структуры данных) и learning rate (параметр, регулирующий скорость обучения). Оптимальный выбор этих параметров может существенно влиять на качество визуализации.

1.7 Резюме по модулю

Обучение без учителя (англ. Unsupervised learning) - это подраздел машинного обучения, в котором модель обучается на неразмеченных данных без наличия явно заданных целевых переменных или меток. В отличие от обучения с учителем, где модель обучается на парах «объект-ответ», в задачах обучения без учителя нет явно определенных целей или правильных ответов. Основная цель обучения без учителя состоит в извлечении скрытых структур, паттернов или информации из неразмеченных данных.

Основные задачи обучения без учителя включают:

- Кластеризация: Задача заключается в разделении набора данных на группы или кластеры, где объекты внутри каждого кластера схожи между собой, а объекты из разных кластеров различаются. Кластеризация помогает выявить скрытую структуру в данных и понять, как объекты могут быть группированы на основе их сходства.
- Снижение размерности: Задача состоит в уменьшении размерности данных путем проецирования их на пространство меньшей размерности. Целью является сохранение важных характеристик и структуры данных, одновременно устранив шум и избыточность. Понижение размерности упрощает визуализацию и анализ данных, а также может улучшить производительность моделей машинного обучения.

Основные алгоритмы кластеризации:

- К-средних (K-means): Это один из наиболее популярных и простых алгоритмов кластеризации. Он разбивает набор данных на K кластеров, где каждый объект присваивается к ближайшему центроиду. Центроиды пересчитываются на каждой итерации до достижения сходимости.
- Иерархическая кластеризация: Этот алгоритм строит иерархическую структуру кластеров, которая может быть представлена в виде дендрограммы. Он может быть агломеративным, начиная с каждого объекта в отдельном кластере и объединяя их постепенно, или дивизивным, начиная с одного кластера и разделяя его на подкластеры.
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise): Этот алгоритм основан на плотности данных. Он определяет кластеры, исходя из плотности объектов в их окрестности. DBSCAN может обнаруживать кластеры произвольной формы и имеет возможность обнаруживать выбросы.
- GMM (Gaussian Mixture Model): Это вероятностная модель, которая моделирует данные как смесь нескольких гауссовых распределений. GMM предполагает, что каждый кластер имеет свое гауссово распределение, и объекты внутри кластера генерируются из этого распределения. В рамках этого курса мы его не рассматриваем.

Основные алгоритмы снижения размерности:

- PCA (Principal Component Analysis): Это один из наиболее распространенных алгоритмов снижения размерности. PCA находит линейные комбинации исходных признаков, называемые главными компонентами, которые объясняют наибольшую дисперсию в данных. После этого можно выбрать первые k главных компонент для сокращения размерности данных.
- t-SNE (t-Distributed Stochastic Neighbor Embedding): Этот алгоритм используется для визуализации и снижения размерности данных. Он стремится сохранить локальные сходства объектов, перенося их на пространство меньшей размерности. t-SNE обычно

хорошо работает для сохранения глобальных структур данных, но может иметь проблемы с сохранением глобальных расстояний.

- UMAP (Uniform Manifold Approximation and Projection): Этот алгоритм является относительно новым методом снижения размерности, который также используется для визуализации данных. UMAP стремится сохранить глобальную структуру данных и их локальные сходства. Он основан на построении графа соседей и оптимизации распределения объектов на меньшую размерность с сохранением сходства соседей. В рамках этого курса мы его не рассматриваем.