Шпаргалка по обучению без учителя / Концепции

Cheatsheet (XeLaTeX)

Краткий справочник

April 2, 2025

Contents

Обучение без учителя (Unsupervised Learning): Введение

В отличие от обучения с учителем, здесь у нас **нет правильных ответов** (меток) для данных. Цель — найти **скрытые структуры**, закономерности или группы в самих данных. Представь, что тебе дали кучу разных носков, и ты должен рассортировать их по парам, не зная изначально, какие носки составляют пару — ты ищешь схожесть сам.

- **Основные задачи:** Понижение размерности, кластеризация, поиск аномалий, изучение ассоциативных правил.
- **Ключевые методы для старта:** PCA (понижение размерности) и K-Means (кластеризация).

1 Метод Главных Компонент (PCA - Principal Component Analysis)

Идея: Понижение размерности

Представь, что у тебя есть сложный объект (многомерные данные), и ты хочешь сделать его "плоскую" фотографию (понизить размерность), но так, чтобы на фото сохранилось как можно больше информации о форме объекта. PCA именно это и делает — находит наиболее информативные "проекции" данных. **Цель:** Уменьшить количество признаков (столбцов) в данных, сохранив при этом максимальное количество **дисперсии** (информации, вариативности).

Как работает PCA (Интуиция)

PCA ищет новые оси (называемые **главными компонентами**) в пространстве признаков. Эти оси обладают следующими свойствами:

- **Максимизация дисперсии:** Первая главная компонента — это направление, вдоль которого данные имеют наибольший разброс (дисперсию). Вторая — следующее направление с максимальной дисперсией, но **ортогональное** (перпендикулярное) первой, и так далее.
- **Ортогональность:** Главные компоненты не коррелируют друг с другом.
- **Математика (основа):** Эти направления *математически* соответствуют **собственным векторам** (*eigenvectors*) ковариационной матрицы данных. "Важность" каждого направления (объем объясняемой дисперсии) определяется соответствующим **собственным значением** (*eigenvalue*). Чем больше собственное значение, тем больше дисперсии объясняет компонента.

Аналогия "Тени": Представь, что твои данные — это облако точек в 3D. PCA находит такую "стену" (плоскость), что тень (проекция) от облака на эту стену будет максимально "размазанной", то есть сохранит максимум информации об исходной форме облака. Эта стена и задается главными компонентами.

Когда применять PCA и Как выбрать компоненты?

- **Визуализация:** Уменьшение размерности до 2D или 3D для построения графиков.
- **Уменьшение шума:** Отбрасывание компонент с малой дисперсией может убрать шум.
- **Борьба с мультиколлинеарностью:** Главные компоненты не коррелируют, что полезно для некоторых моделей (например, линейной регрессии).
- **Ускорение обучения:** Меньше признаков -> быстрее работают другие алгоритмы ML.
- **Важно:** PCA чувствителен к **масштабу данных**. Перед применением PCA признаки обычно нужно **стандартизировать** (например, с помощью StandardScaler).
- **Выбор количества компонент:** Часто смотрят на **долю объясненной дисперсии** (explained variance ratio) для каждой компоненты. Выбирают такое количество компонент, которое объясняет достаточный процент общей дисперсии (например, 90-99%) или ищут "локоть" на графике кумулятивной объясненной дисперсии (когда добавление новой компоненты уже не дает значительного прироста объясненной дисперсии).

2 Кластеризация K-Means

Идея: Кластеризация

Разделить все объекты (точки данных) на заранее заданное число (K) групп (кластеров) так, чтобы объекты внутри одного кластера были максимально похожи друг на друга, а объекты из разных кластеров — максимально различны. **Аналогия "Почтовые отделения":** Представь, что тебе нужно открыть K почтовых отделений в городе так, чтобы суммарное расстояние от всех жителей до ближайшего к ним отделения было минимальным. K-Means решает похожую задачу: центры кластеров (**центроиды**) — это как почтовые отделения, а точки данных — жители.

Как работает K-Means (Алгоритм)

Это итеративный алгоритм:

1. **Инициализация:** Выбрать K (количество кластеров). Случайным образом разместить K **центроидов** (начальных центров кластеров).
2. **Шаг присваивания (Assignment Step):** Каждую точку данных отнести к кластеру, чей центроид к ней **ближе всего** (обычно по евклидову расстоянию).
3. **Шаг обновления (Update Step):** Пересчитать положение каждого центроида как **среднее арифметическое** всех точек, отнесенных к этому кластеру на предыдущем шаге.
4. **Повторение:** Повторять шаги 2 и 3 до тех пор, пока центроиды не перестанут значительно смещаться или не будет достигнуто максимальное число итераций.

Важно: Результат K-Means зависит от начального положения центроидов. Обычно алгоритм запускают несколько раз с разными инициализациями и выбирают лучший результат (например, с наименьшим WCSS - см. ниже). Также K-Means предполагает, что кластеры имеют примерно сферическую форму и одинаковый размер.

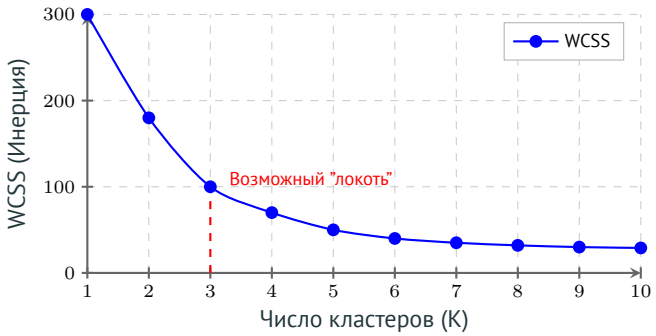
2.1 Выбор количества кластеров (K)

Как подобрать K?

Выбор оптимального K — нетривиальная задача, так как нет единственно "правильного" ответа. Популярные эвристические методы:

- **Метод Локтя (Elbow Method):** Строится график зависимости суммы квадратов расстояний от точек до центроидов их кластеров (**WCSS - Within-Cluster Sum of Squares** или инерция) от числа кластеров K. Ищется точка "изгиба" (локтя) на графике, после которой добавление нового кластера уже не дает существенного уменьшения WCSS. *Аналогия: После определенного момента добавление нового почтового отделения уже не сильно сокращает среднее расстояние для жителей.*
- **Метод Силуэта (Silhouette Method):** Запускают K-Means для разных K и выбирают то K, для которого **средний Силуэт** по всем точкам максимален (ближе к 1). Этот метод часто дает более осмысленные результаты, чем метод локтя.
- **Знание предметной области:** Иногда оптимальное количество кластеров можно определить исходя из понимания данных и бизнес-задачи (например, мы знаем, что у нас 3 типа клиентов).

Метод Локтя (Пример)



2.2 Оценка качества: Силуэт (Silhouette Score)

Интуиция Силуэта

Как понять, насколько хорошо точки сгруппировались? Метрика "Силуэт" помогает это оценить для каждой точки данных. Она показывает, насколько точка "хорошо сидит" в своем кластере по сравнению с соседними кластерами. Для каждой точки i:

- $a(i)$: Среднее расстояние от точки i до **всех других точек в её собственном** кластере (насколько она "своя"). Чем меньше $a(i)$, тем лучше.
- $b(i)$: Минимальное **среднее** расстояние от точки i до **всех точек в каком-либо другом** ("соседнем", ближайшем) кластере (насколько она далека от "чужих"). Чем больше $b(i)$, тем лучше.

Силуэт точки i:

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

Интерпретация значений $s(i)$**:**

- Близко к $+1$: Точка i хорошо соответствует своему кластеру и далека от других (идеальный случай).
- Близко к 0: Точка i находится на границе между кластерами.
- Близко к -1: Точка i, скорее всего, попала не в тот кластер.

Общий Силуэт: Часто усредняют значения $s(i)$ по всем точкам, чтобы получить общую оценку качества кластеризации для выбранного K. Чем ближе средний силуэт к 1, тем лучше разделение на кластеры. **Аналогия "Районы города":** Силуэт жителя показывает, насколько он близок к своим соседям по району ($a(i)$) и насколько он далек от жителей ближайшего другого района ($b(i)$). Высокий силуэт — четко очерченные районы, низкий — районы смешаны или житель живет "на границе".

2.3 Плюсы K-Means

Преимущества

- **Простота и скорость:** Легко понять, реализовать и он относительно быстро работает даже на больших наборах данных. Вычислительная сложность линейно зависит от числа точек.
- **Интерпретируемость:** Концепция центроидов как "представителей" кластера интуитивно понятна.

2.4 Минусы K-Means

Ограничения

- **Чувствительность к инициализации:** Разные начальные положения центроидов могут привести к разным результатам кластеризации. Решение: многократные запуски (параметр `n_init` в scikit-learn) или "умная" инициализация (K-Means++).
- **Необходимость задавать K:** Нужно заранее знать количество кластеров или использовать эвристики (метод локтя, силуэт) для его подбора, что не всегда просто.
- **Предположение о форме кластеров:** K-Means хорошо работает, когда кластеры имеют примерно сферическую (выпуклую) форму, одинаковый размер и плотность. Он плохо справляется с вытянутыми кластерами, кластерами сложной формы или разного размера. (Алгоритмы вроде DBSCAN или спектральной кластеризации лучше подходят для таких случаев).
- **Чувствительность к выбросам:** Аномальные точки могут сильно смещать положение центроидов, искажая результат кластеризации.