Шпаргалка по обучению без учителя / Концепции Cheatsheet (XeLaTeX)

Краткий справочник

Содержание

Обучение без учителя (Unsupervised Learning): Введение

В отличие от обучения с учителем, здесь **нет правильных ответов** (меток) для данных. Цель — найти **скрытые структуры**, закономерности или группы в самих данных.

Пример и Задачи

Пример: 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/DBSCAN (кластеризация).

1 Метод Главных Компонент (PCA - Principal Component Analysis)

Идея: Понижение размерности

PCA — это метод понижения размерности, который находит новые, самые информативные "проекции" или "ракурсы" данных (главные компоненты). **Цель:** Уменьшить количество признаков (столбцов), сохранив при этом максимальное количество **вариативности (дисперсии)** исходных данных.

Почему Дисперсия

В контексте PCA предполагается, что направление, вдоль которого данные сильнее всего "разбросаны" (имеют большую дисперсию), несет больше всего информации об их различиях. Если вдоль какого-то направления все точки почти одинаковы (малая дисперсия), это направление мало что добавляет к пониманию структуры данных.

Механика PCA: Шаги

PCA находит новые оси (главные компоненты, PC) в пространстве исходных признаков. Алгоритм включает следующие шаги:

1. **Стандартизация данных:** *Критически важно!* Вычитаем среднее и делим на стандартное отклонение для **каждого** признака. PCA чувствителен к масштабу; без стандартизации признаки с большими значениями будут доминировать.
2. **Расчет ковариационной матрицы:** Строится матрица, показывающая, как стандартизированные признаки изменяются *совместно*.
3. **Нахождение собственных векторов и значений (λ):** Для ковариационной матрицы вычисляются её собственные векторы (*eigenvectors*) и значения (*eigenvalues*).
   - **Собственный вектор:** Указывает **направление** главной компоненты.
   - **Собственное значение λ:** Показывает, **сколько дисперсии** объясняется вдоль этого направления.
4. **Сортировка и выбор компонент:** Собственные векторы сортируются по убыванию их λ. PC1 соответствует наибольшему λ, PC2 — второму по величине λ (и ортогональна PC1), и т.д. Компоненты ортогональны друг другу.
5. **Проецирование:** Исходные (стандартизированные) данные проецируются на выбранные k главных компонент (с наибольшими λ). Результат — новый набор данных с k признаками.

Аналогия "Эллипс и Тени"

Представь облако точек данных как эллипс. Главные компоненты — это оси этого эллипса (самая длинная — PC1, следующая перпендикулярная ей — PC2 и т.д.). PCA находит эти оси, поворачивает данные и отбрасывает оси с наименьшей дисперсией, оставляя проекцию на самые важные.

Применения PCA и Чувствительность к Масштабу

Применения:

- Визуализация данных (понижение до 2D/3D).
- Уменьшение шума.
- Борьба с мультиколлинеарностью.
- Ускорение обучения других моделей.
- Сжатие данных.

Чувствительность к масштабу: Всегда стандартизируйте данные перед PCA! Иначе признаки с большим масштабом "перетянут" всю дисперсию на себя.

Выбор количества компонент (k) в PCA

Как определить оптимальное число компонент k:

- **Доля объясненной дисперсии:** Смотрят на кумулятивную (накопленную) долю $\sum_{i=1}^{k} \lambda_i / \sum_{j=1}^{N} \lambda_j$. Выбирают k так, чтобы объяснить достаточный процент (например, 90-99%) общей дисперсии.
- **Метод Локтя (*Scree Plot*):** Ищут "изгиб" на графике кумулятивной объясненной дисперсии или на графике самих собственных значений (λ_i от i). Точка изгиба показывает, где добавление новой компоненты перестает давать существенный прирост информации.
- **Исходя из задачи:** Для визуализации $k = 2$ или $k = 3$. Для других задач можно подбирать k по кросс-валидации для *конечной* ML-модели (например, классификатора, который будет использовать PCA-признаки).

2 Кластеризация K-Means

Идея: Центроидная Кластеризация

Разделить все объекты на заранее заданное число (K) групп (кластеров) так, чтобы объекты внутри кластера были максимально похожи (близки) друг на друга, а объекты из разных кластеров — максимально различны. **Цель:** Минимизировать суммарное квадратичное расстояние от точек до центров их кластеров (WCSS - Within-Cluster Sum of Squares).

Аналогия "Почтовые отделения"

Открыть K почтовых отделений (центроидов) в городе так, чтобы суммарное *квадратичное* расстояние от всех жителей (точек данных) до ближайшего к ним отделения было минимальным.

Алгоритм K-Means

Это итеративный алгоритм:

1. **Инициализация:** Выбрать K. Разместить K **центроидов**. Рекомендуется "умная" инициализация **K-Means++** (размещает центроиды подальше друг от друга).
2. **Шаг присваивания (Assignment):** Для каждой точки найти ближайший центроид (обычно по евклидову расстоянию) и присвоить точку его кластеру.
3. **Шаг обновления (Update):** Для каждого кластера пересчитать положение его центроида как центр масс (среднее) всех точек, попавших в этот кластер.
4. **Повторение:** Повторять шаги 2 и 3 до сходимости (центроиды почти не смещаются, или точки не меняют кластеры).

Метрики расстояния в K-Means

Выбор метрики зависит от данных и задачи:

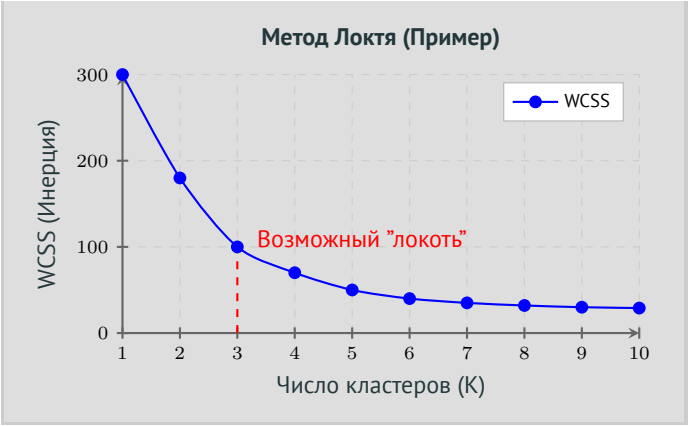
- **Евклидово:** $\sqrt{\sum(x_i - y_i)^2}$. Стандартный выбор, хорошо для сферических кластеров.
- **Манхэттенское:** $\sum |x_i - y_i|$. "Расстояние городских кварталов".
- **Косинусное:** $1 - \cos(\theta) = 1 - \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}$. Измеряет угол между векторами, полезно для текстов (TF-IDF).

2.1. Выбор количества кластеров (K)

Методы подбора K

Выбор оптимального K — нетривиальная задача. Популярные методы:

- **Метод Локтя (Elbow Method):** Строится график WCSS от K. Ищется точка "изгиба" (локтя), после которой WCSS убывает значительно медленнее. (См. график ниже).
- **Метод Силуэта (Silhouette Method):** Вычисляют средний Силуэт для разных K. Выбирают K с максимальным средним Силуэтом. (См. следующий раздел).
- **Знание предметной области:** Иногда K можно определить из контекста задачи.



2.2. Оценка качества: Силуэт (Silhouette Score)

Интуиция Силуэта

Метрика "Силуэт" оценивает, насколько хорошо точка "сидит" в своем кластере по сравнению с соседними. Показывает качество разделения.

Расчет Силуэта для точки i

- $a(i)$: Среднее расстояние от точки i до **всех других точек в её собственном** кластере. *(Внутрикластерная схожесть. Меньше = лучше).*
- $b(i)$: **Минимальное** из средних расстояний от точки i до **всех точек в каждом из других** ("соседних") кластеров. *(Межкластерное различие. Больше = лучше).*

Формула Силуэта точки i:

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

Интерпретация и Аналогия Силуэта

Значения $s(i)$: От -1 до +1.

- $s(i) \approx +1$: Точка плотно сидит в своем кластере, далеко от других (отлично).
- $s(i) \approx 0$: Точка на границе между кластерами.
- $s(i) \approx -1$: Точка, вероятно, попала не в тот кластер (плохо).

Общий Силуэт: Усредняют $s(i)$ по всем точкам. Чем ближе средний силуэт к 1, тем лучше разделение на кластеры для данного K. **Аналогия "Районы города":** Высокий силуэт жителя ($s(i) \approx 1$) — живет в центре четко очерченного района. Низкий ($s(i) \approx 0$) — на границе. Отрицательный ($s(i) \approx -1$) — ближе к центру соседнего района.

2.3. Плюсы и Минусы K-Means

Преимущества K-Means

- **Простота и скорость:** Легко понять, реализовать, относительно быстро работает на средних данных.
- **Масштабируемость:** Существуют вариации (MiniBatch K-Means) для больших данных.
- **Интерпретируемость:** Концепция центроидов как "представителей" кластера понятна.

Ограничения K-Means

- **Чувствительность к инициализации:** Результат зависит от начального положения центроидов. **Решение:** Использовать K-Means++ и многократные запуски (`n_init` в scikit-learn).
- **Необходимость задавать K:** Нужно знать K заранее или подбирать эвристиками.
- **Предположение о форме кластеров:** Ищет **выпуклые, сферические** кластеры примерно одинакового размера. Плохо работает с вытянутыми, вогнутыми кластерами или кластерами разной плотности.
- **Чувствительность к выбросам:** Выбросы могут сильно смещать центроиды.

3 Кластеризация DBSCAN (Density-Based)

Идея: Кластеризация на основе плотности

DBSCAN находит **плотные регионы** точек, разделенные областями с низкой плотностью. Позволяет находить кластеры **произвольной формы** и автоматически определяет **выбросы (шум)**. Не требует заранее задавать число кластеров K.

Аналогия "Поиск островов"

Представь карту с точками-домами. DBSCAN ищет "острова" (кластеры), где дома стоят близко друг к другу (*плотные регионы*), отделенные "водой" (*разреженные области*). Дома, стоящие совсем одиноко в "воде", считаются шумом.

Ключевые Параметры и Понятия DBSCAN

Требует два параметра:

- `eps` (ϵ): Радиус окрестности. Максимальное расстояние между двумя точками, чтобы считать их соседями.
- `min_samples`: Минимальное число соседей (включая саму точку) в ϵ-окрестности, чтобы точка считалась "основной".

Основные типы точек:

- **Основная точка (Core Point):** Точка, у которой в ϵ-окрестности $\geq$ `min_samples` точек (включая себя).
- **Граничная точка (Border Point):** Не основная точка, но находится в ϵ-окрестности некоторой *основной* точки.
- **Шум (Noise Point):** Точка, не являющаяся ни основной, ни граничной. Выброс.

Принцип Работы Алгоритма DBSCAN

1. Выбирается произвольная непосещенная точка.
2. Если точка *основная*, начинается формирование нового кластера. Все точки, достижимые по плотности от неё (т.е. её ϵ-соседи, и их ϵ-соседи, если они основные, и так далее), добавляются в этот кластер. Граничные точки тоже добавляются, но от них кластер не "растет".
3. Если точка *не основная* (граничная или шум), она помечается как посещенная (возможно, как шум; если позже окажется в окрестности основной точки, станет граничной и войдет в кластер).
4. Шаги 1-3 повторяются, пока все точки не будут посещены.

Точки, оставшиеся помеченными как шум, не принадлежат ни одному кластеру.

Плюсы и Минусы DBSCAN

Плюсы:

- Не нужно заранее задавать количество кластеров K.
- Способен находить кластеры сложной, невыпуклой формы.
- Устойчив к выбросам и явно их идентифицирует как шум.

Минусы:

- Результат чувствителен к выбору параметров `eps` и `min_samples`. Их подбор может быть нетривиальным (часто используют k-distance graph для `eps`).
- Плохо справляется с кластерами, имеющими сильно различающуюся плотность, так как параметры `eps` и `min_samples` глобальны.
- Может быть вычислительно затратным на очень больших датасетах (сложность около $O(N \log N)$ или $O(N^2)$ без пространственных индексов).