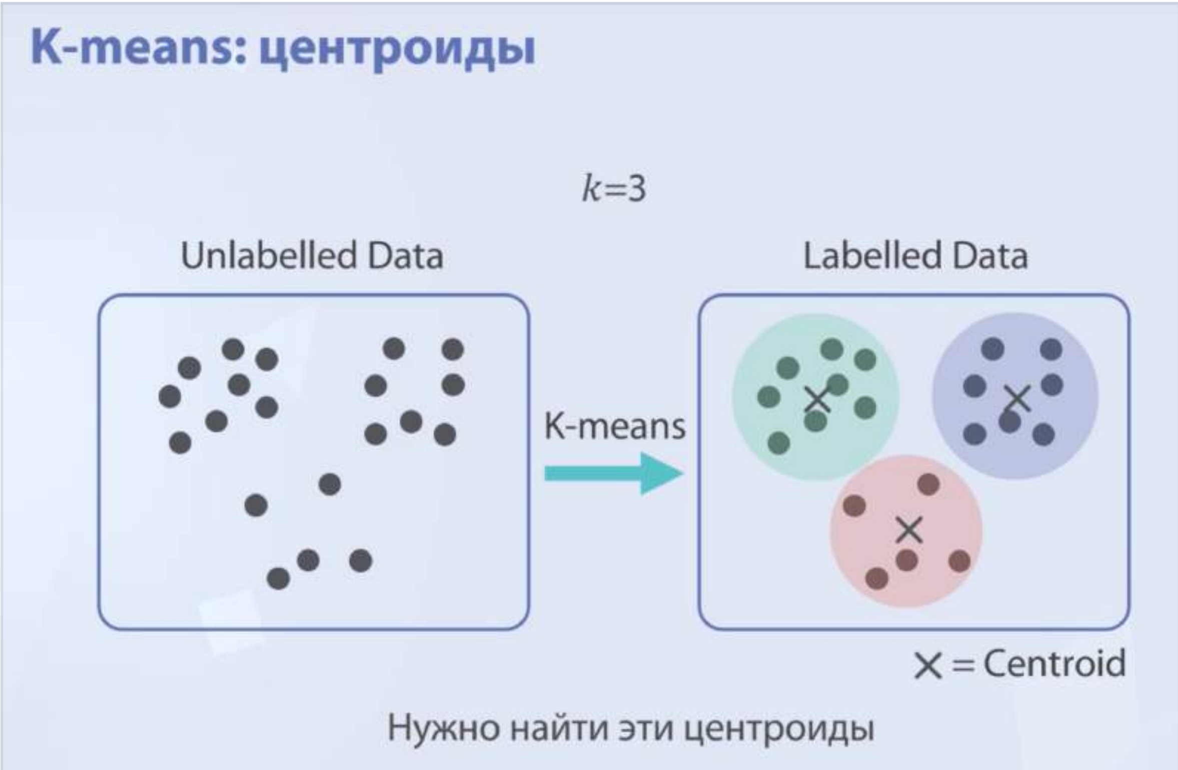
ML

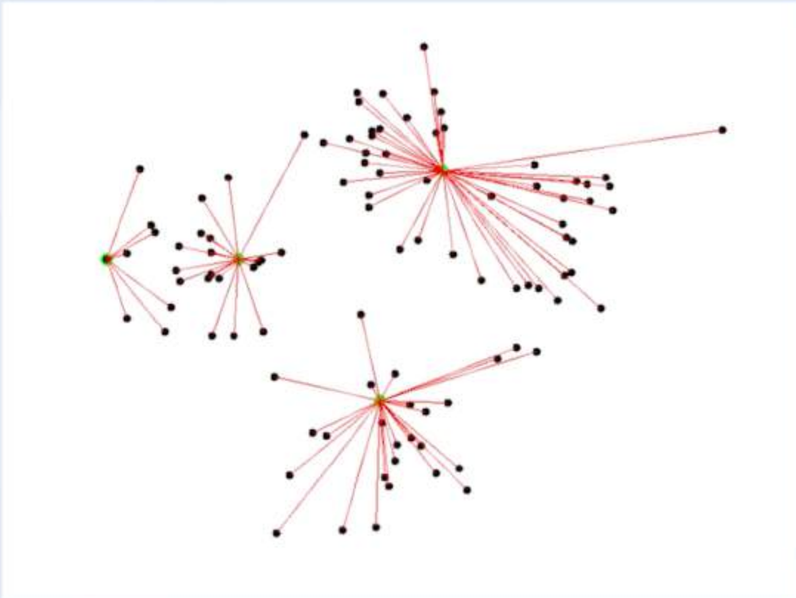
K-MEANS
https://colab.research.google.com/drive/1fg_NyOIgcM2IdJuQSBIKLFu1vKpb4qP1?usp=sharing




Для каждой точки находим ближайший центр


Пересчитываем центры

Опять ищем для каждой точки ближайший центр



+

- Очень простой алгоритм
- Работает даже на больших данных

−

- Надо задавать число k руками
- Не всегда находит кластеры правильно

DBSCAN

https://colab.research.google.com/drive/1fg_NyOIgcM2IdJuQSBIKLFu1vKpb4qP1?usp=sharing

DBSCAN: идея

- Хотим найти скопления точек, не зная заранее количество кластеров.
- **Параметры:**

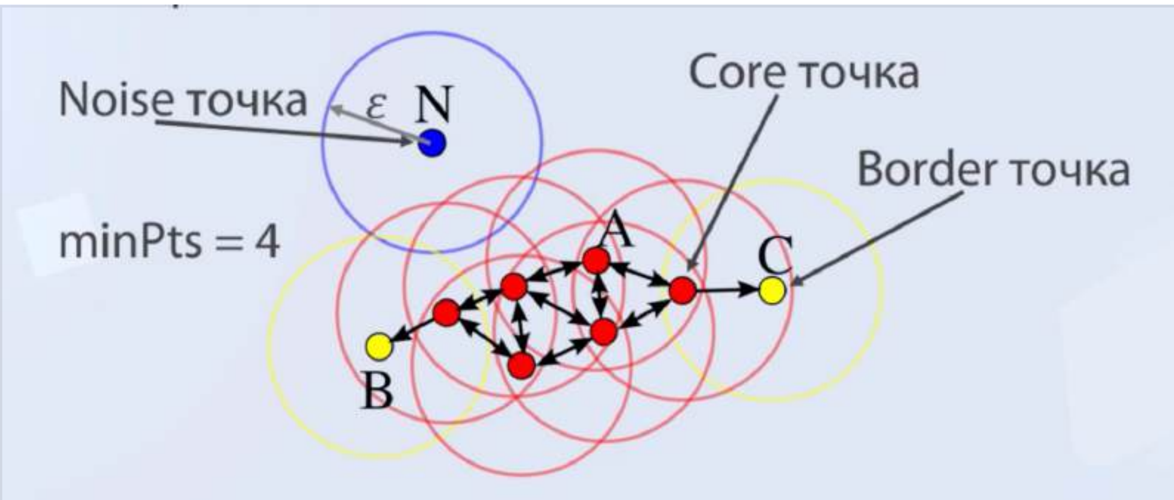
  ε – размер окрестности точки

  minPts – минимальное количество точек в ε-окрестности
- **Все точки делятся на 3 типа:**

  Core точки – от minPts соседей в ε-окрестности

  Border точки – не Core точки, но достижимы из Core точек

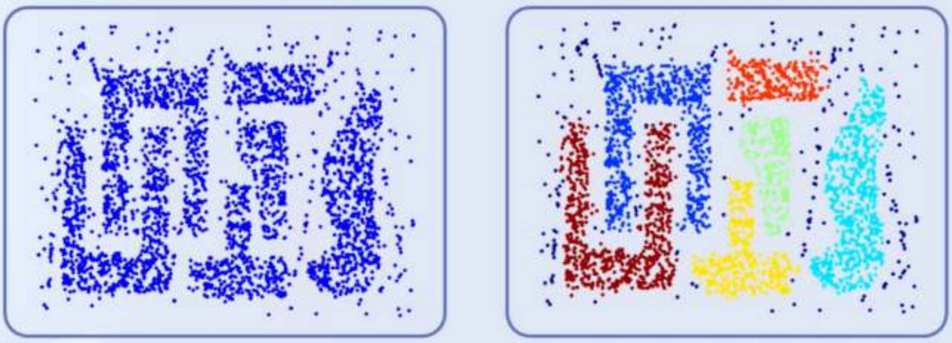
  Noise точки – все остальные, меньше minPts соседей в ε-окрестности



DBSCAN: алгоритм

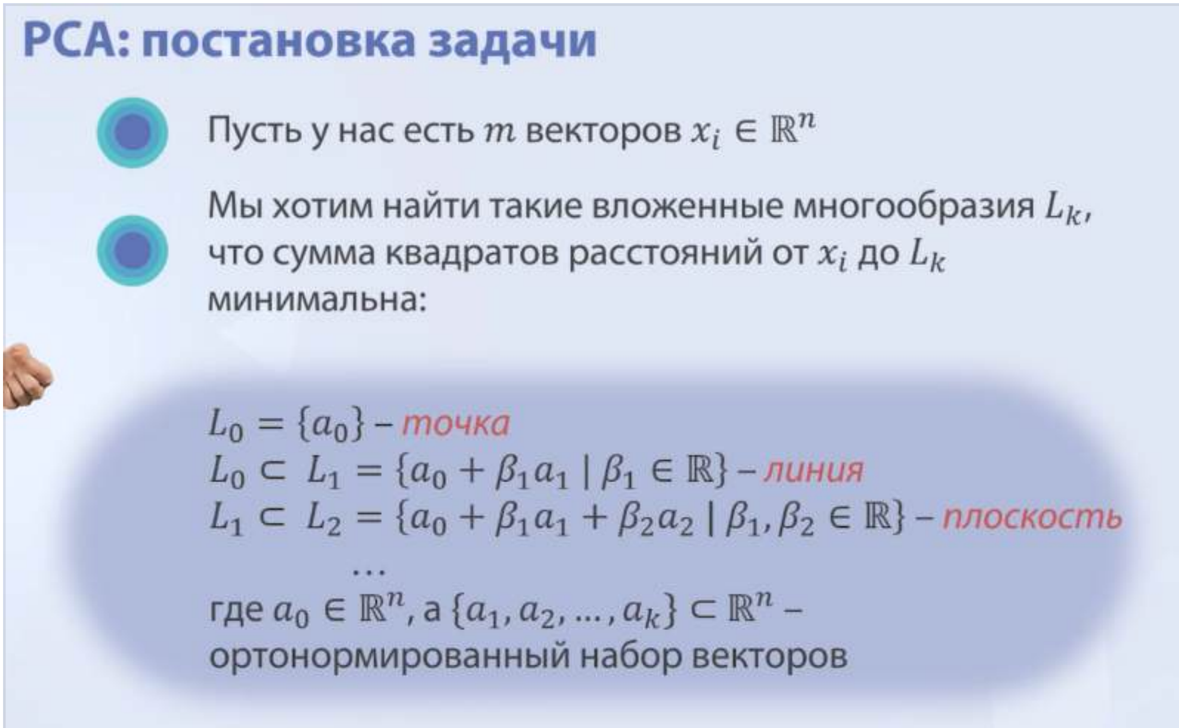
- Для следующей произвольной точки ищем соседей в ε-окрестности
- Если их как минимум minPts, начинаем поиск связной компоненты из этой Core-точки
- Иначе помечаем точку как Noise, она может быть подключена позднее к какой-то Core-точке как Border точка

DBSCAN: пример




Резюме: DBSCAN
+
– Не нужно задавать кол-во кластеров
– Кластеры могут быть любой формы
– Может работать с шумными данными
–
– Долго работает на больших данных
– Чувствителен к выбору гипер-параметров (лучше использовать HDBSCAN)

PCA


PCA: постановка задачи
Пусть у нас есть m векторов $x_i \in \mathbb{R}^n$
Мы хотим найти такие вложенные многообразия L_k, что сумма квадратов расстояний от x_i до L_k минимальна:
$L_0 = \{a_0\}$ – точка
$L_0 \subset L_1 = \{a_0 + \beta_1 a_1 \mid \beta_1 \in \mathbb{R}\}$ – линия
$L_1 \subset L_2 = \{a_0 + \beta_1 a_1 + \beta_2 a_2 \mid \beta_1, \beta_2 \in \mathbb{R}\}$ – плоскость
…
где $a_0 \in \mathbb{R}^n$, а $\{a_1, a_2, \dots, a_k\} \subset \mathbb{R}^n$ – ортонормированный набор векторов

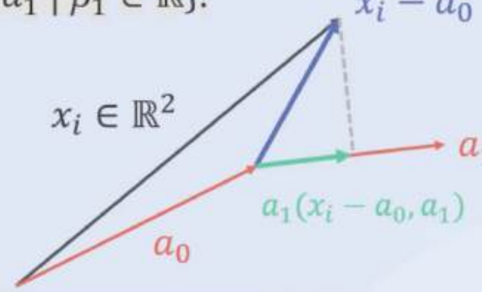
PCA: расстояние до точки

- Пусть у нас есть m векторов $x_i \in \mathbb{R}^n$
- Мы хотим найти такие вложенные многообразия L_k, что сумма квадратов расстояний от x_i до L_k минимальна
- Для точки $L_0 = \{a_0\}$:

$$a_0 = \underset{a_0 \in \mathbb{R}^n}{\operatorname{argmin}} \left(\sum_{i=1}^{m} \|x_i - a_0\|^2 \right) = \frac{1}{m} \sum_{i=1}^{m} x_i$$

PCA: расстояние до линии

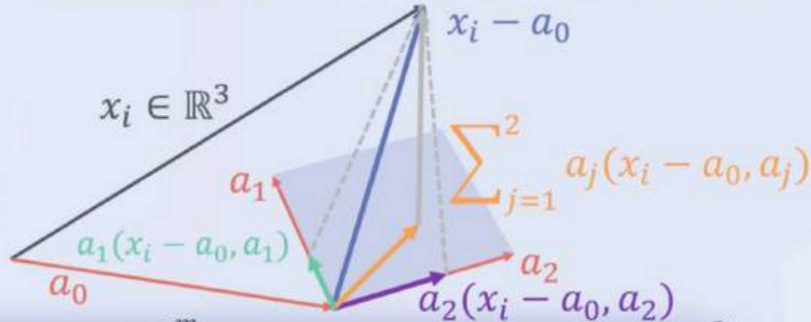
- Пусть у нас есть m векторов $x_i \in \mathbb{R}^n$
- Мы хотим найти такие вложенные многообразия L_k, что сумма квадратов расстояний от x_i до L_k минимальна
- Для линии $L_1 = \{a_0 + \beta_1 a_1 \mid \beta_1 \in \mathbb{R}\}$:



$$a_1 = \underset{\|a_1\|=1}{\operatorname{argmin}} \left(\sum_{i=1}^{m} \|x_i - a_0 - a_1(x_i - a_0, a_1)\|^2 \right)$$

PCA: расстояние до плоскости

- Пусть у нас есть m векторов $x_i \in \mathbb{R}^n$
- Мы хотим найти такие вложенные многообразия L_k, что сумма квадратов расстояний от x_i до L_k минимальна
- Для плоскости $L_2 = \{a_0 + \beta_1 a_1 + \beta_2 a_2 \mid \beta_1, \beta_2 \in \mathbb{R}\}$:



$$a_2 = \underset{\|a_2\|=1}{\operatorname{argmin}} \left(\sum_{i=1}^{m} \left\| x_i - a_0 - \sum_{j=1}^{2} a_j(x_i - a_0, a_j) \right\|^2 \right)$$

PCA: расстояние до многообразия

$$a_1 = \underset{\|a_1\|=1}{\operatorname{argmin}} \left(\sum_{i=1}^{m} \| x_i - a_0 - a_1(x_i - a_0, a_1) \|^2 \right)$$

$$a_2 = \underset{\|a_2\|=1}{\operatorname{argmin}} \left(\sum_{i=1}^{m} \left\| x_i - a_0 - \sum_{j=1}^{2} a_j(x_i - a_0, a_j) \right\|^2 \right)$$

- В общем виде:

$$a_k = \underset{\|a_k\|=1}{\operatorname{argmin}} \left(\sum_{i=1}^{m} \left\| x_i - a_0 - \sum_{j=1}^{k} a_j(x_i - a_0, a_j) \right\|^2 \right)$$

PCA: алгоритм

1. $a_0 = \frac{1}{m}\sum_{i=1}^{m} x_i$
2. $x_i := x_i - a_0$
3. $a_1 = \underset{\|a_1\|=1}{\operatorname{argmin}}\left(\sum_{i=1}^{m}\|x_i - a_1(x_i, a_1)\|^2\right)$
4. $x_i := x_i - a_1(x_i, a_1)$
5. $a_2 = \underset{\|a_2\|=1}{\operatorname{argmin}}\left(\sum_{i=1}^{m}\|x_i - a_2(x_i, a_2)\|^2\right)$ *так как* $a_2 \perp a_1$
6. $x_i := x_i - a_2(x_i, a_2)$
7. Повторяем пока не найдем k компонент

PCA: компоненты с максимальной дисперсией

Распишем одно слагаемое в задаче оптимизации:

$$
\begin{aligned}
&\|x_i - a_k(x_i, a_k)\|^2 = \\
&= \|x_i\|^2 + \|a_k(x_i, a_k)\|^2 - 2\big(x_i, a_k(x_i, a_k)\big) = \\
&= \|x_i\|^2 + (x_i, a_k)^2 - 2(x_i, a_k)^2 = \\
&= \|x_i\|^2 - (x_i, a_k)^2
\end{aligned}
$$

Перепишем нашу задачу оптимизации:

$$a_k = \underset{\|a_k\|=1}{\operatorname{argmin}}\left(\sum_{i=1}^{m}\|x_i - a_k(x_i, a_k)\|^2\right) = \underset{\|a_k\|=1}{\operatorname{argmax}}\left(\sum_{i=1}^{m}(x_i, a_k)^2\right)$$

Так как $\mathbb{D}\xi = \mathbb{E}\xi^2 - (\mathbb{E}\xi)^2$ и x_i центрированы в нуле,
то мы максимизируем дисперсию длин проекций на a_k

PCA: компоненты с максимальной дисперсией

Максимизация дисперсии в матричной записи (x_i – строки матрицы X):

$$a_1 = \underset{\|a_1\|=1}{\operatorname{argmax}} \left(\sum_{i=1}^{m} (x_i, a_1)^2 \right) = \underset{\|a_1\|=1}{\operatorname{argmax}} \|Xa_1\|^2 =$$
$$= \underset{\|a_1\|=1}{\operatorname{argmax}} (Xa_1)^T X a_1 = \underset{\|a_1\|=1}{\operatorname{argmax}} a_1^T X^T X a_1$$

Применим метод множителей Лагранжа:

$$L = a_1^T X^T X a_1 - \lambda(a_1^T a_1 - 1)$$
$$\frac{\partial L}{\partial a_1} = X^T X a_1 - \lambda a_1 = 0$$

То есть a_1 – собственный вектор $X^T X$, соответствующий максимальному собственному значению, так как $a_1^T X^T X a_1 = a_1^T \lambda a_1 = \lambda$

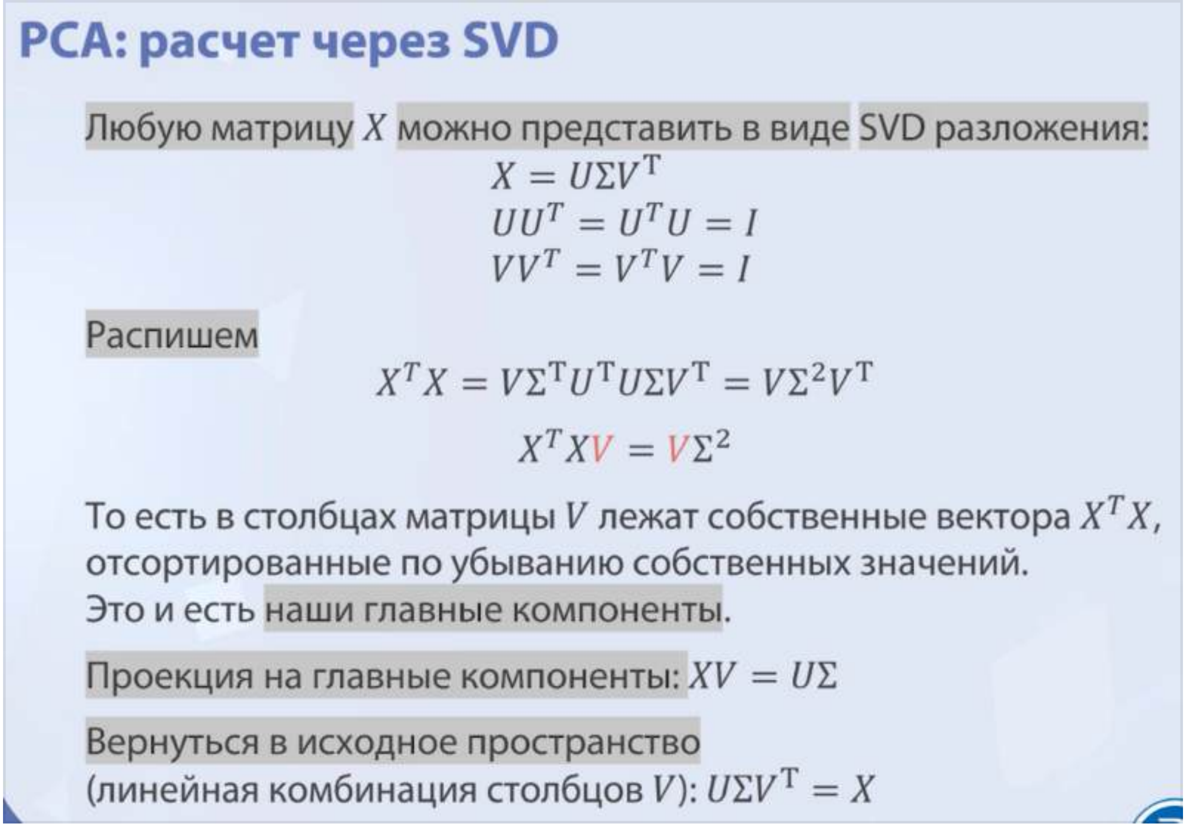
PCA: компоненты с максимальной дисперсией

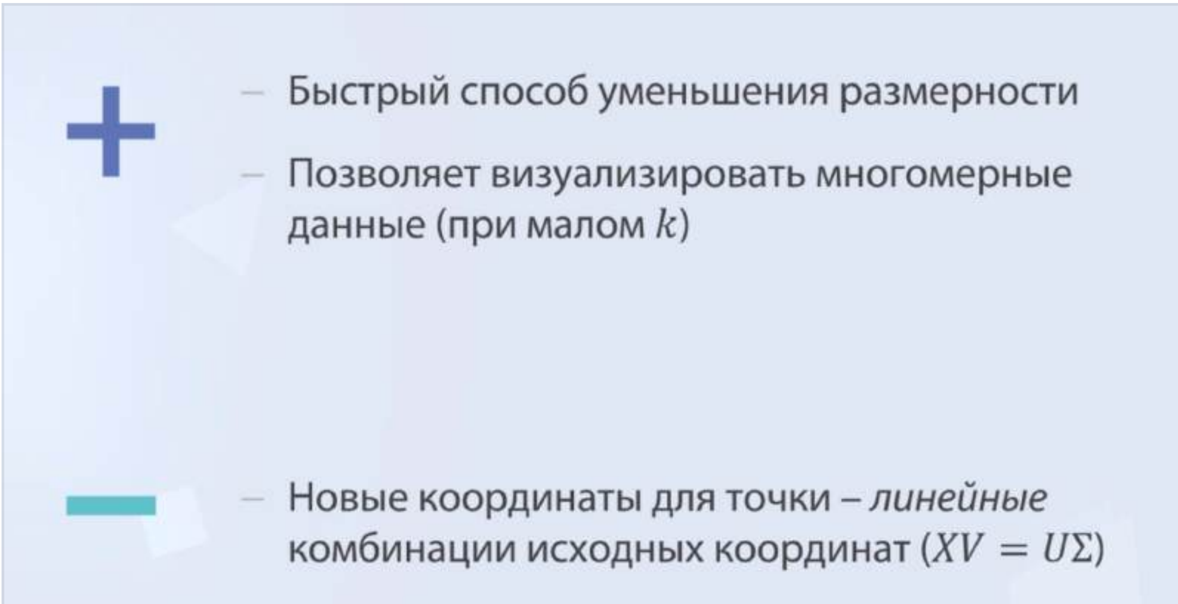
Продолжим алгоритм: $y_i \coloneqq x_i - a_1(x_i, a_1)$ в матричном виде $Y \coloneqq X - X a_1 a_1^T$, ищем $a_2 = \underset{\|a_2\|=1}{\operatorname{argmax}} a_2^T Y^T Y a_2$:

$$Y^T Y = (X^T - a_1 a_1^T X^T)(X - X a_1 a_1^T) =$$
$$= X^T X - X^T X a_1 a_1^T - a_1 a_1^T X^T X + a_1 a_1^T X^T X a_1 a_1^T =$$
$$= X^T X - \lambda_1 a_1 a_1^T - a_1 a_1^T X^T X + \lambda_1 a_1 \underbrace{a_1^T a_1}_{1} a_1^T =$$
$$= X^T X - a_1 a_1^T X^T X$$

$$a_2 = \underset{\|a_2\|=1}{\operatorname{argmax}} a_2^T Y^T Y a_2 = \underset{\|a_2\|=1}{\operatorname{argmax}} (a_2^T X^T X a_2 - \underbrace{a_2^T a_1}_{0} a_1^T X^T X a_2) =$$

$= \underset{\|a_2\|=1,\, a_2 \perp a_1}{\operatorname{argmax}} a_2^T X^T X a_2$, то есть a_2 – следующий с.в. $X^T X$


PCA: расчет через SVD
Любую матрицу X можно представить в виде SVD разложения:
$$X = U\Sigma V^{\mathrm{T}}$$
$$UU^T = U^T U = I$$
$$VV^T = V^T V = I$$
Распишем
$$X^T X = V\Sigma^{\mathrm{T}} U^{\mathrm{T}} U \Sigma V^{\mathrm{T}} = V\Sigma^2 V^{\mathrm{T}}$$
$$X^T X V = V\Sigma^2$$
То есть в столбцах матрицы V лежат собственные вектора $X^T X$,
отсортированные по убыванию собственных значений.
Это и есть наши главные компоненты.
Проекция на главные компоненты: $XV = U\Sigma$
Вернуться в исходное пространство
(линейная комбинация столбцов V): $U\Sigma V^{\mathrm{T}} = X$


+
– Быстрый способ уменьшения размерности
– Позволяет визуализировать многомерные данные (при малом k)
–
– Новые координаты для точки – *линейные* комбинации исходных координат ($XV = U\Sigma$)

t-SNE

t-SNE: постановка задачи

- Пусть у нас есть m векторов $x_i \in \mathbb{R}^n$
- Хотим снизить размерность каждого вектора x_i, сохранив информацию о расстоянии до остальных векторов:

1. Зададим распределение пропорциональное расстоянию до остальных точек:

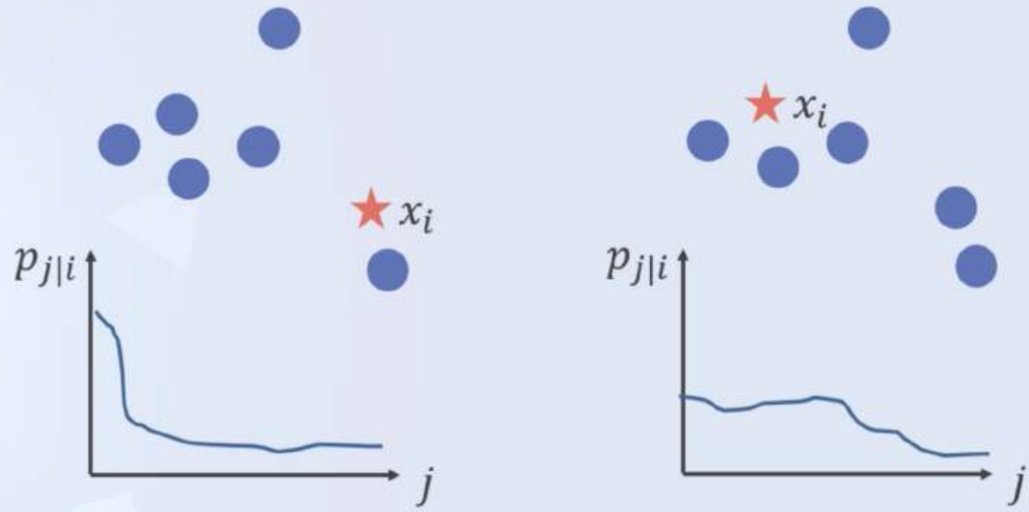
$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2/2\sigma_i^2)}$$

2. Сделаем его симметричным:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2}$$

3. Пока что положим $\sigma_i \equiv \sigma$ (*какая-то константа*)

t-SNE: как выглядит $p_{j|i}$ с $\sigma_i \equiv \sigma$



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

t-SNE: уменьшение размерности

- Нужно найти такие $y_i \in \mathbb{R}^2$ (или $\mathbb{R}^3$), соответствующие x_i, чтобы распределение q_{ij} расстояний между y_i было похоже на распределение p_{ij}:

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$

- Похожесть распределений будем считать при помощи KL-дивергенции:

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

- Будем минимизировать $KL(P||Q)$ по y_i градиентным спуском:

$$\frac{\partial KL}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$$

t-SNE: выбор σ_i

Для визуализации лучше сохранить информацию о k ближайших соседях каждой точки x_i

Для этого нужно сделать энтропию распределений $p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$ примерно одинаковой для всех i

Подберем такие σ_i, чтобы $\mathrm{Perp}(P_i) = 2^{-\sum_j p_{j|i} \log p_{j|i}}$ была примерно одинаковой для всех i

Этого можно добиться бинарным поиском каждой σ_i

$\mathrm{Perp}(P_i)$ становится гиперпараметром метода (от 5 до 50)

t-SNE: визуализация MNIST

На входе пиксели вытянутые в строку

lvdmaaten.github.io

t-SNE: визуализация ImageNet

На входе нейросетевые имбединги картинок

для практики https://colab.research.google.com/drive/18j8sNLcqTscjn6W--LSCHb0DrUofYNQj?usp=sharing