

Машинное обучение

Методы снижения размерности

(28.09.2023)

нет у

либо не
использует

Многомерные данные



Объекты могут описываться большим числом признаков



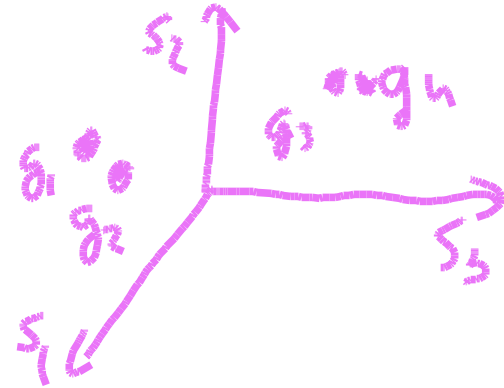
Зачем нужны такие данные?



Q и R анализы

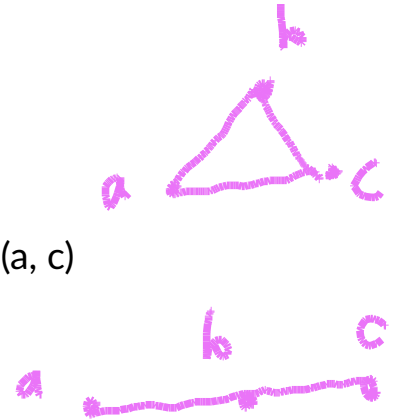
R и Q

1. R-анализ: Выясняем взаимоотношения между признаками
2. Q-анализ: Выясняем взаимоотношения между объектами



Меры сходства и различия

1. Сходство (S) достигает максимума, когда объекты обладают идентичными признаками, различия (D), наоборот - достигает минимума.
2. Обычно (но не всегда) коэффициенты сходства распределены от 0 до 1.
3. Для большинства метрик будут верны следующие свойства:
 - a. Если $a == b$, то $D(a, b) == 0$
 - b. Симметричность $D(a, b) == D(b, a)$
 - c. Справедливо неравенство треугольника $D(a, b) + D(b, c) \geq D(a, c)$



Меры сходства и различия. Примеры



Меры сходства и различия. Значимость нулей

$$a \quad (10, 5) \\ b \quad (7, 8)$$

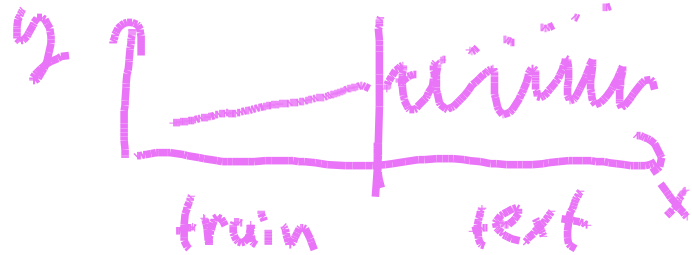
$$D(a, b) = \sqrt{(10-7)^2 + (5-8)^2}$$

$$a \quad (10, 5, 0) \\ b \quad (7, 8, 0)$$

$$D(a, b) = \sqrt{(10-7)^2 + (5-8)^2 + (0-0)^2}$$

Минусы большого количества признаков

1. Случайные признаки ✓
2. Корреляция признаков ✓
3. Разреженные данные ✓
4. Долго считать ✓



$$n=2 \quad \bar{D}(a,b)=0.5$$

$$n=3 \quad \bar{D}(a,b)=0.54$$

$$n=10^6 \quad \bar{D}(a,b)=4.00$$

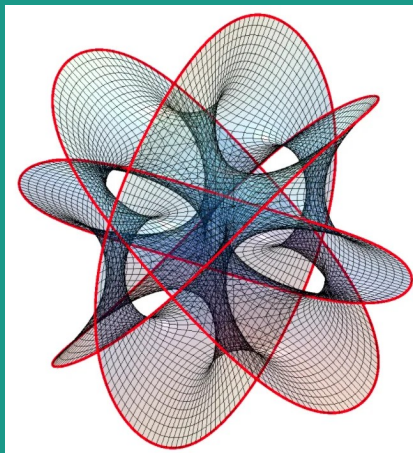
$$1) p=x$$

$$2) p=2x$$

$$y = (w_1) p + (w_2) q$$

$$y = 3x \cdot (w^*) = 10$$

Методы снижения размерности



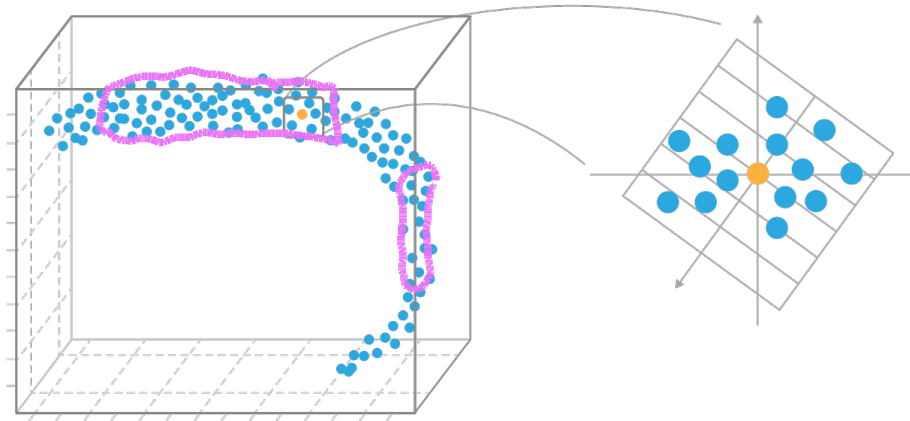
Зачем снижать размерность?



1. Многие алгоритмы показывают себя плохо на пространствах большой размерности в принципе (проклятие размерности) ✓
2. Некоторые - просто будут значительно дольше работать, при этом качество их работы не изменится от уменьшения размерности ✓
3. Помогает понижение размерности и избавиться от шума ✓
4. Задача визуализации - хочется взглянуть на наши объекты, а делать это в 100-мерном или 100000-мерном пространстве неудобно ✓
5. Удаление выбросов - в пространствах меньшей размерности можем их увидеть глазами ✓
6. Можем увидеть закономерности в данных ✓

Manifold assumption

Мы будем терять часть информации об объектах. Но мы считаем, что при правильных настройках алгоритма понижения размерности, потери будут незначительны. Что нам позволяет это делать - мы предполагаем, что наши данные на самом деле лежат в пространстве меньшем, чем пространство исходных признаков.



Анализ главных компонент

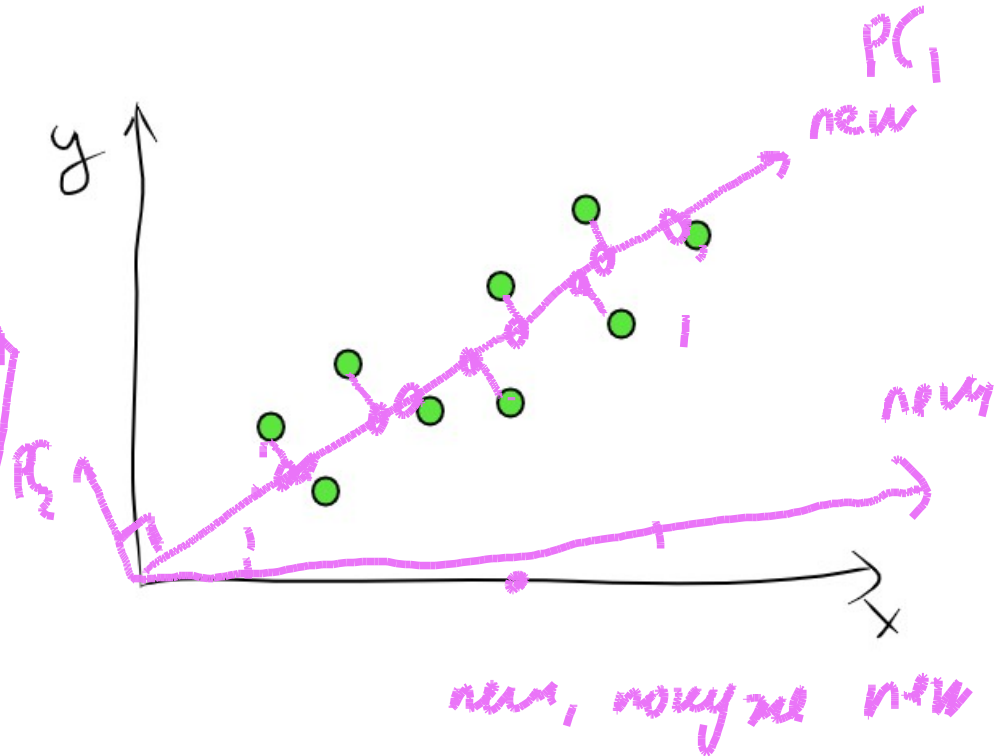
Анализ главных компонент (PCA)

$$new = \underline{a} \cdot x + \underline{b} \cdot y$$

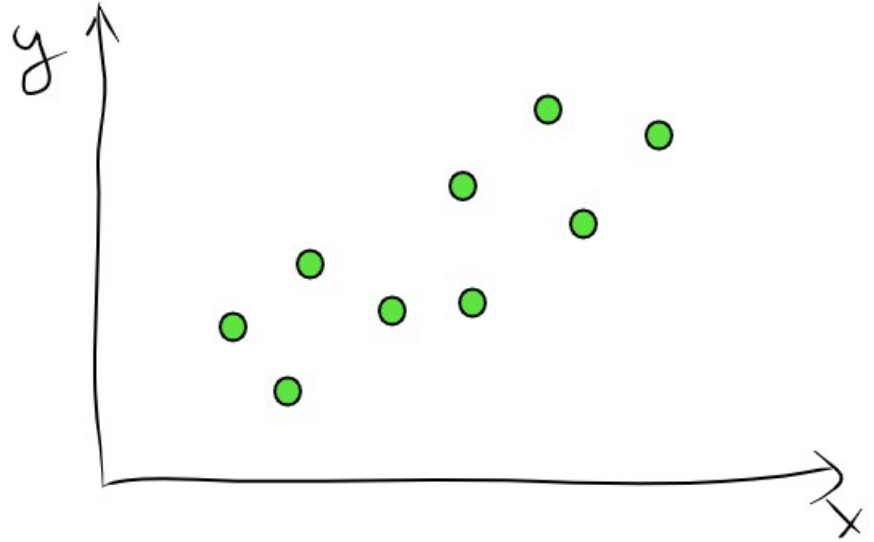
$$D(\underline{a} \cdot x + \underline{b} \cdot y) \rightarrow \max$$

$$PC_1 = \begin{pmatrix} a \\ b \end{pmatrix}$$

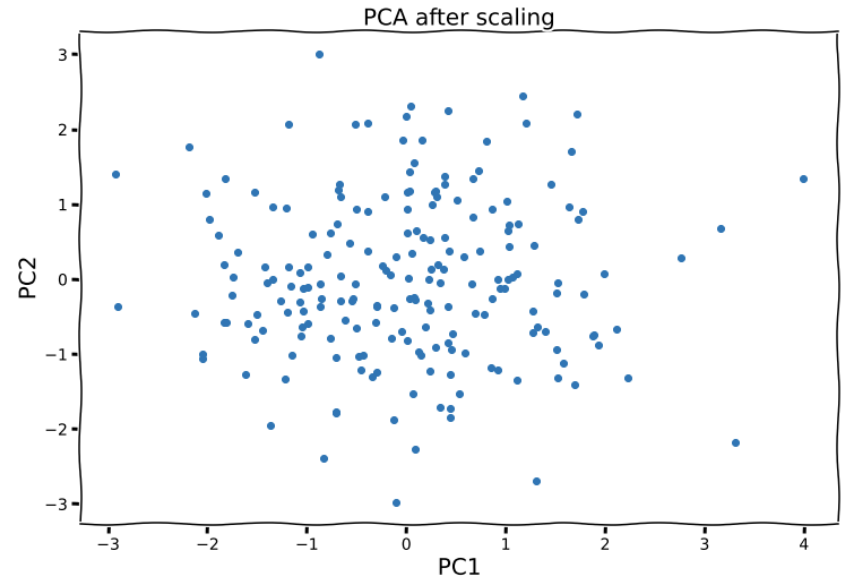
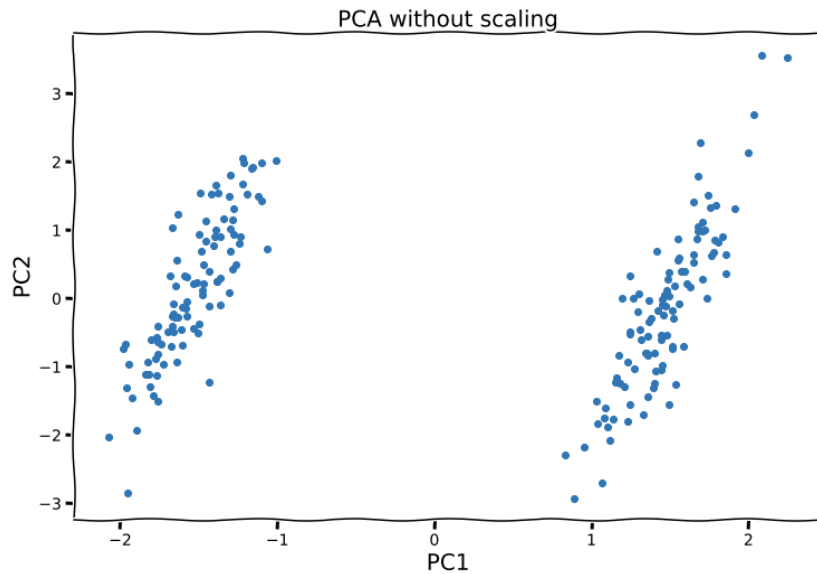
$$\|PC_1\|_2 = 1$$
$$\sqrt{a^2 + b^2} = 1$$



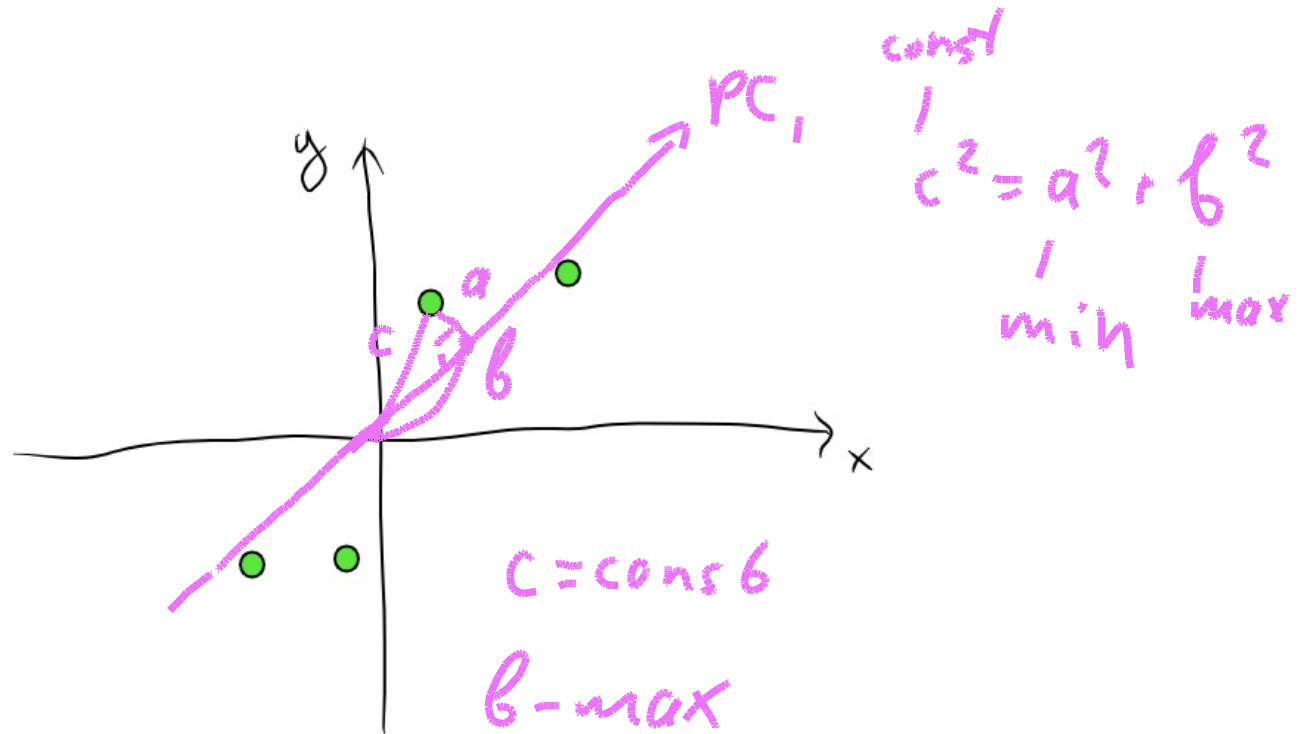
Максимизируем дисперсию



Стандартизация



Что мы вообще делаем?



А как получить главные компоненты?

$$\text{Cov}(X_i, X_j) = E[(X_i - E(X_i))(X_j - E(X_j))] = \\ = E(X_i, X_j) - E(X_i)E(X_j) = \dots$$

$$\text{Cov}(X_i, X_i) = \dots$$

$$\begin{matrix}
 & X_1 & X_2 & \dots & X_n \\
 \begin{matrix} X \\ p \times n \end{matrix} & X_1 & \begin{bmatrix} D(X_1) & \text{Cov}(X_1, X_2) \\ \text{Cov}(X_2, X_1) & D(X_2) \\ \vdots & \vdots & \ddots & \vdots \\ X_n & \text{Cov}(X_n, X_1) & \text{Cov}(X_n, X_2) & \dots & D(X_n) \end{bmatrix}
 \end{matrix}$$

$$D(x) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

$$D(y) = \frac{1}{n-1} \sum_{i=1}^n (y_i - \bar{y})^2$$

$$\text{Cov}(x, y) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

$$\text{Cov}(X) = X^T \cdot X$$

d - kon-Bo nryrur.

np. l: noly eig($X^T X$)

$$A \odot v_i = \lambda_i v_i$$

codeword beumup
womb zuam

$$X^T X$$

2x2

$$\left[\begin{array}{cc} \lambda_1 = 2 & v_1 \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} \\ \lambda_2 = 1 & v_2 \begin{pmatrix} a_2 \\ b_2 \end{pmatrix} \end{array} \right]$$

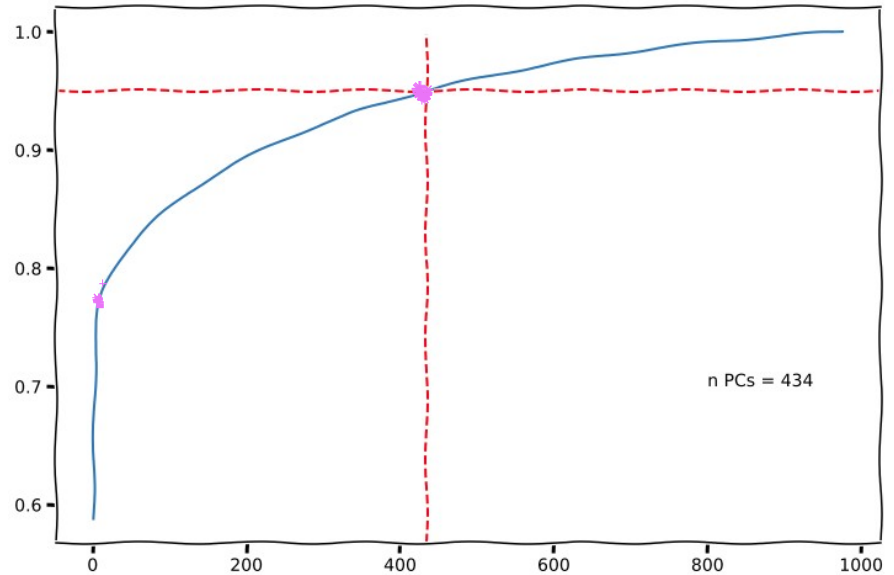
$$new_1 = a_1 \cdot x + b_1 \cdot y$$

$$new_2 = a_2 \cdot x + b_2 \cdot y$$

$$P_1 = \frac{2}{2+1} = 0.67$$

$$P_2 = \frac{1}{2+1} = 0.33$$

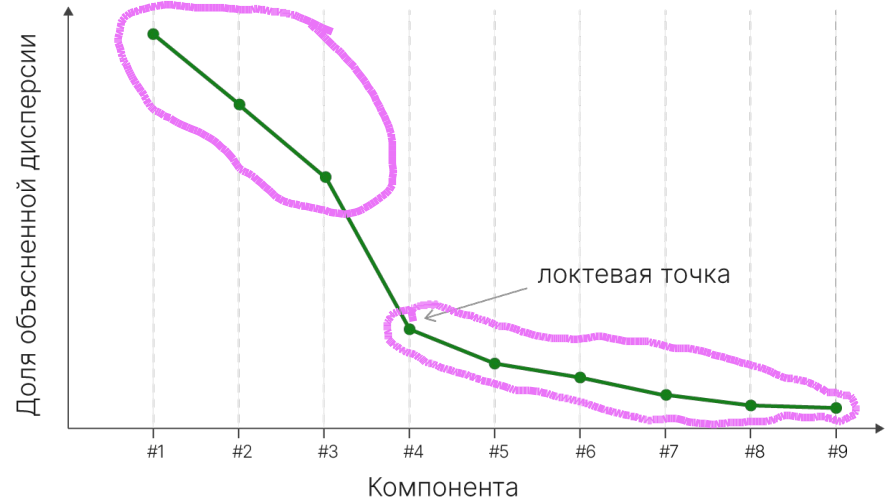
А сколько их всего?



н.с.р.
н.с.р.
н.с.р.

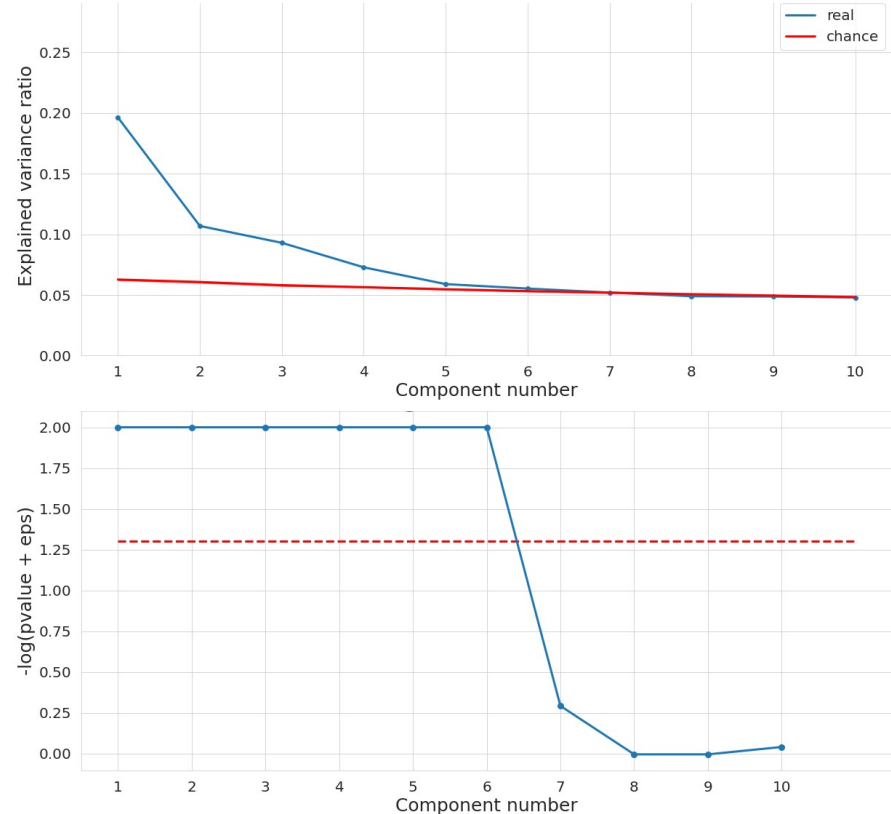
н.с.р.

А сколько их всего?



А сколько их всего?

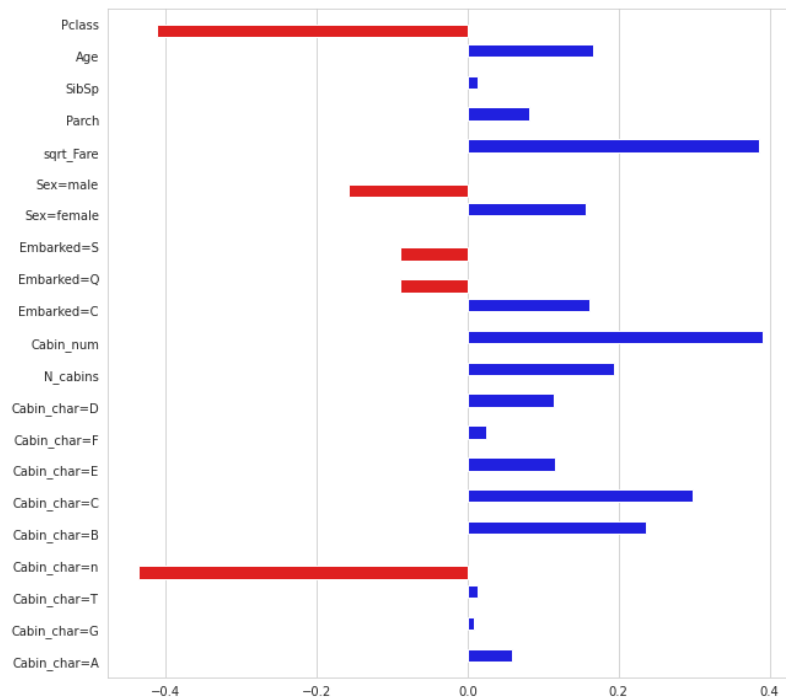
1. Перемешиваем значения каждого признака.
2. Получаем матрицу признаков, которая не содержит никакой информации о данных
3. Делаем PCA
4. Любая explained variance - просто из-за природы данных
5. Делаем так много раз
6. Пусть на реальных данных k -я компонента объясняет $n\%$ дисперсии.
7. Смотрим на распределение доли дисперсии, объясняемой k -компонентой для случайных данных (полученных перемешиванием).
8. Можем сравнить и принять решение, объясняет ли k -я компонента что-то реальное, или просто шум



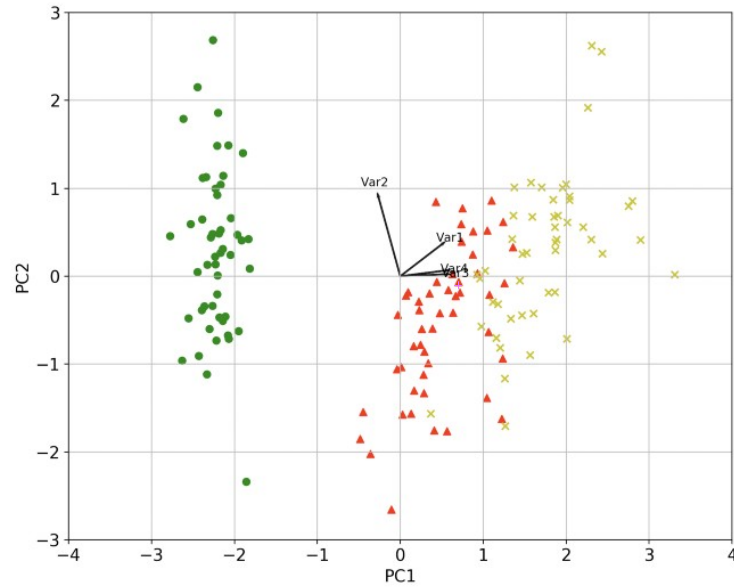
$$new = a_1 \cdot x + b_1 \cdot y$$

$$PC_1 \begin{pmatrix} a_1 \\ b_1 \end{pmatrix}$$

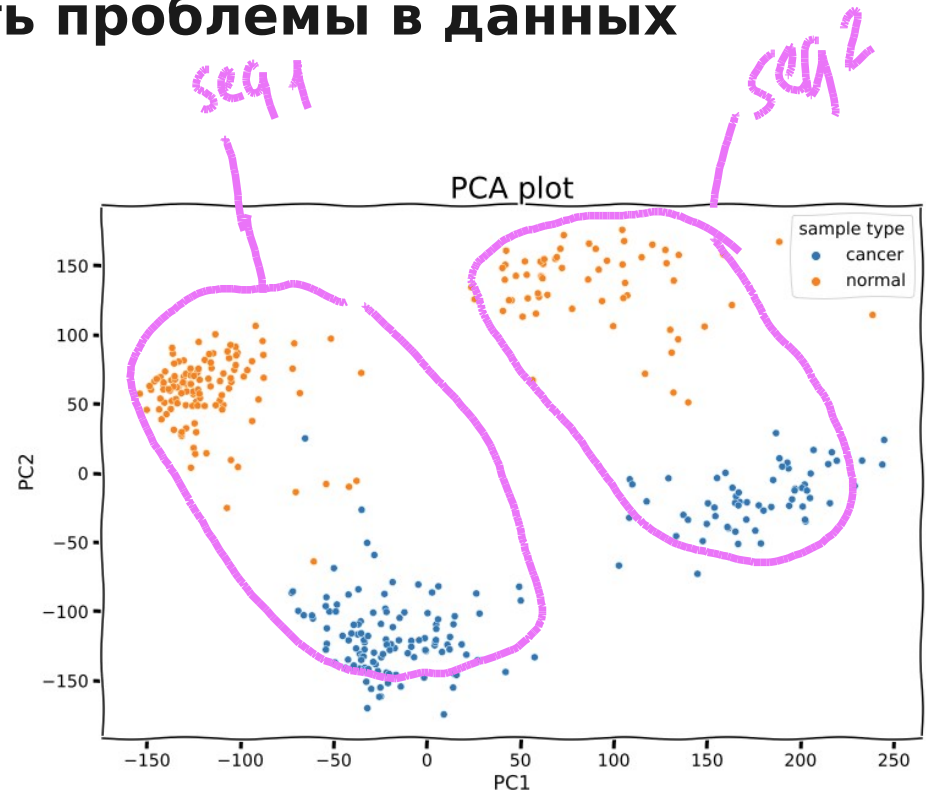
Вклад исходных признаков в компоненты



Biplot

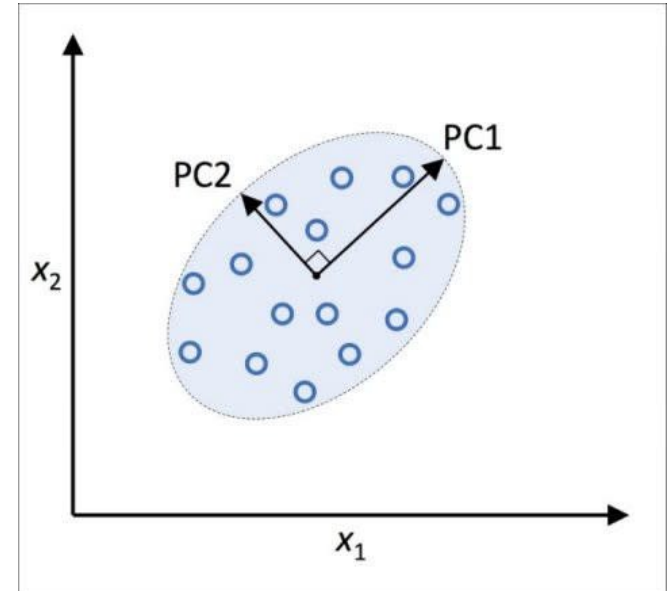


Также можно заметить проблемы в данных




Еще раз

1. Каждая последующая компонента объясняет меньше изменчивости, чем предыдущая
2. Компоненты перпендикулярны
3. Компоненты это линейные комбинации исходных признаков
4. Количество признаков - количество компонент

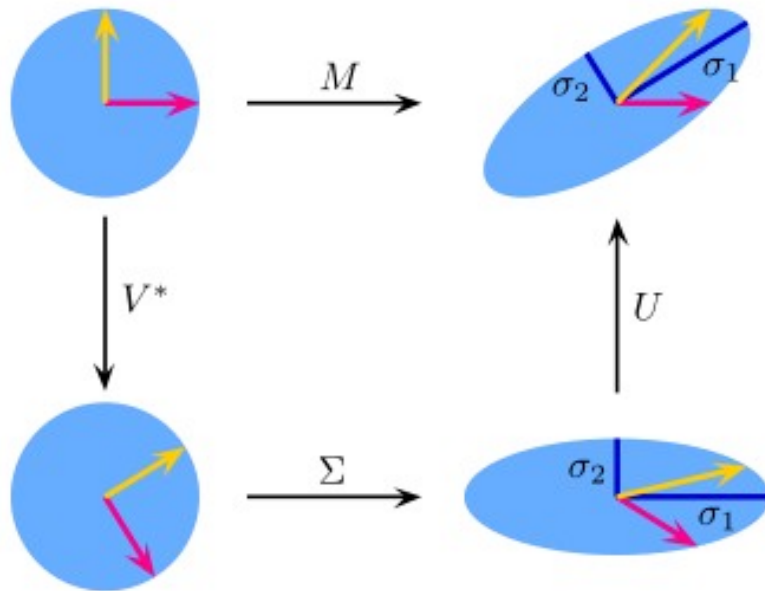


В реальности PCA делается чуть сложнее...


$$X_{[n \times m]} = U_{[n \times r]} \mathcal{E}_{[r \times r]} (V_{[m \times r]})^T$$

SVD

Физический смысл SVD



$$M = U \cdot \Sigma \cdot V^*$$

А с картинками тоже можно?..

Brenda Wilson



Jose Maria Aznar



Tom Hanks



Chen Shui-bian



Lindsay Davenport



“Среднее лицо”

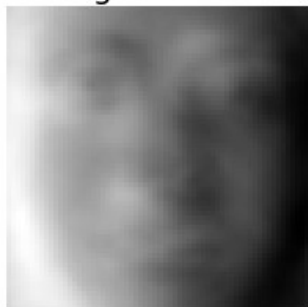


“Собственные” лица

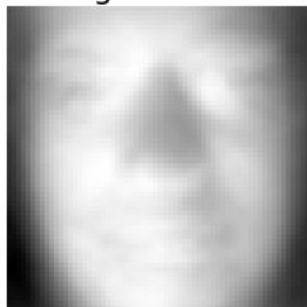
eigenface 0



eigenface 1



eigenface 2



eigenface 3



eigenface 4



Почти то же самое

Components 10



Components 25



Components 100



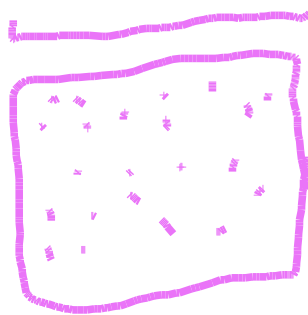
Components 500



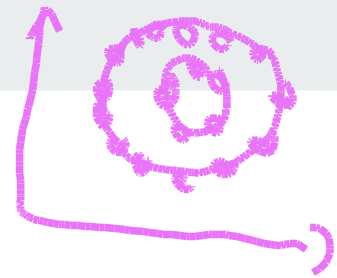
Original



64 x 64
~~128 x 128~~



t-SNE

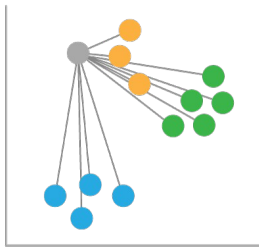


t-SNE (t-distributed stochastic neighbor embedding)

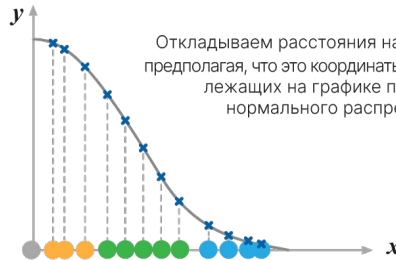
Идея состоит не в том, чтобы напрямую максимизировать дисперсию, а найти такое пространство в котором расстояние между объектами будет сохраняться или по крайней мере не сильно меняться. При этом будем больше беспокоиться о расстоянии между близкими объектами, нежели о расстоянии между далекими

Описываем расстояния в исходном пространстве

1.



Считаем все расстояния от заданной точки до остальных



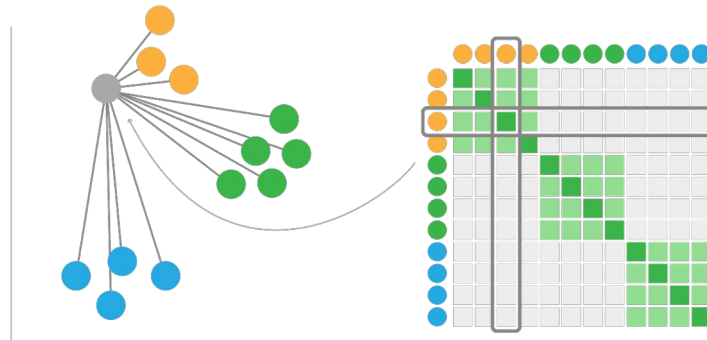
Откладываем расстояния на прямой, предполагая, что это координаты \mathbf{x} точек, лежащих на графике плотности нормального распределения

2.

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|_2 / 2\sigma_i^2)}$$

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

3.

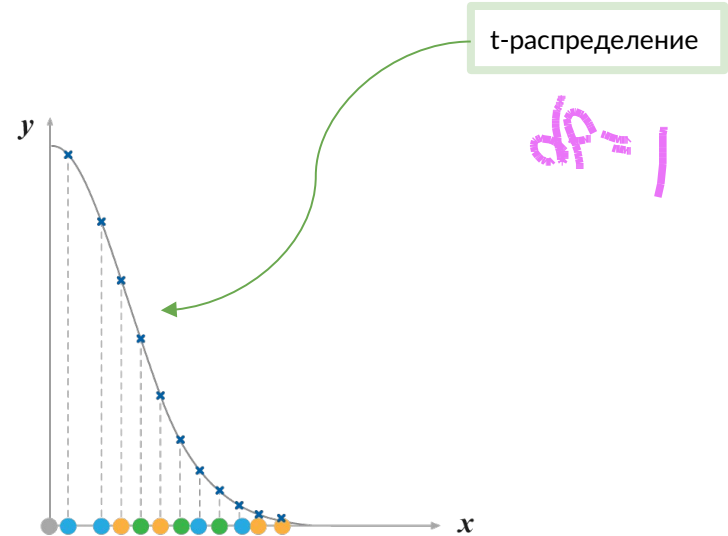


■ Высокая similarity
■ Низкая similarity

Описываем расстояния в пространстве низкой размерности

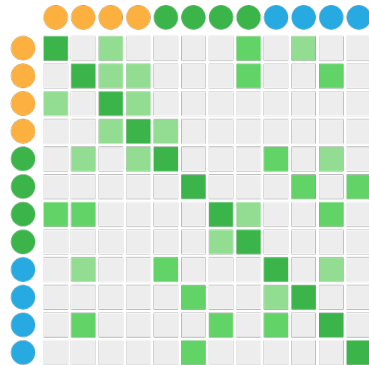


Снова считаем “похожести”... на этот раз назовем их не p , а q

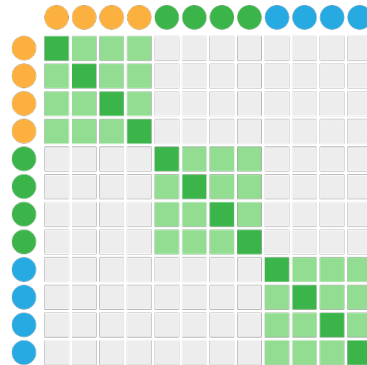


Но они же не похожи...

KL - дивергенция
любых - любых



Матрица расстояний
в пространстве низкой размерности



Матрица расстояний
в пространстве высокой размерности

→ GD
grad. desc.

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

Важные параметры tSNE

perplexity

Определяет то, как подбирается стандартное отклонение для распределения расстояний для каждой точки. Чем больше perplexity - тем более на глобальную структуру мы смотрим

metric

Как считаются расстояния между точками - metric. По умолчанию используется евклидово расстояние, но часто помогают и другие (например, косинусное)

learning rate

Шаг градиентного спуска, тоже влияет на полученное представление

Минусы tSNE

1. Стохастичность ✓
2. Добавление новых точек ✓
3. Расстояния между кластерами точек могут ничего не значить (плохо сохраняются далекие расстояния) ✓
4. Размеры кластеров ничего не значат ✓
5. Можно увидеть артефактные кластеры ✓
6. Можно увидеть не ту структуру, которая по идее должна быть ✓

Tricks

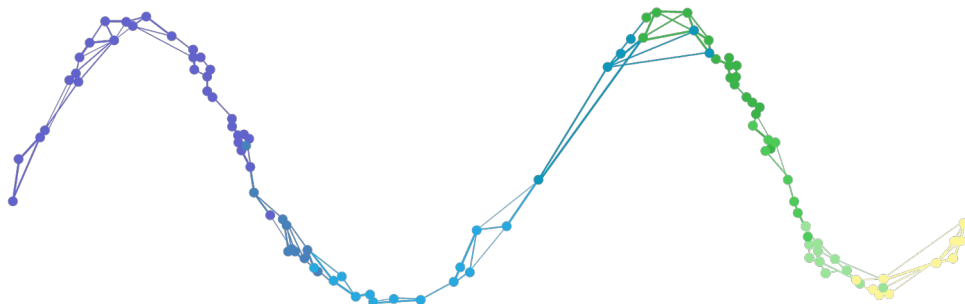
1. Инициализация при помощи PCA
2. Kernel PCA



UMAP

UMAP (uniform manifold approximation and projection)

Внутри себя метод строит граф, в котором ребрами соединены между собой k ближайших соседей. При этом эти ребра неравноправны - если для данной пары точек расстояние между ними сильно больше, чем расстояния между ними и другими точками - то и ребро будет иметь маленький вес.



Далее задача состоит в том, чтобы в пространстве более низкой размерности получился граф похожий на тот, который был в высокой размерности. Для этого опять же, оптимизируем низкоразмерное представление градиентным спуском

Плюсы

1. Быстрее чем tSNE
2. Можно добавлять новые данные