Факультет компьютерных наук | Машинное обучение | Москва 2026

Лекция 4
Снижение размерности данных

Преподаватель: Меликян Алиса Валерьевна, amelikyan@hse.ru
кандидат наук, доцент Департамента программной инженерии ФКН НИУ ВШЭ,
академический руководитель магистерской программы
«Искусственный интеллект и продуктовый подход в HR-менеджменте»

Снижение размерности

Снижение размерности (Dimensionality Reduction) помогает упростить данные с большим числом признаков, сохранив при этом как можно больше важной информации. Зачем используют:

- Уменьшение вычислительной нагрузки;
- Устранение мультиколлинеарности признаков;
- Улучшение визуализации;
- Выявление скрытой структуры взаимосвязей между признаками;
- Feature Engineering: создание новых, более компактных и информативных признаков.

Методы снижения размерности

- PCA (Principal Component Analysis): преобразует исходные признаки в новые, ортогональные компоненты, упорядоченные по убыванию дисперсии. Сохраняет максимум вариации данных при меньшем числе измерений. Оценивает линейные взаимосвязи.

- t-SNE (t-Distributed Stochastic Neighbor Embedding): хорош для визуализации сложных данных в 2D/3D, сохраняет локальную структуру. Оценивает нелинейные взаимосвязи.

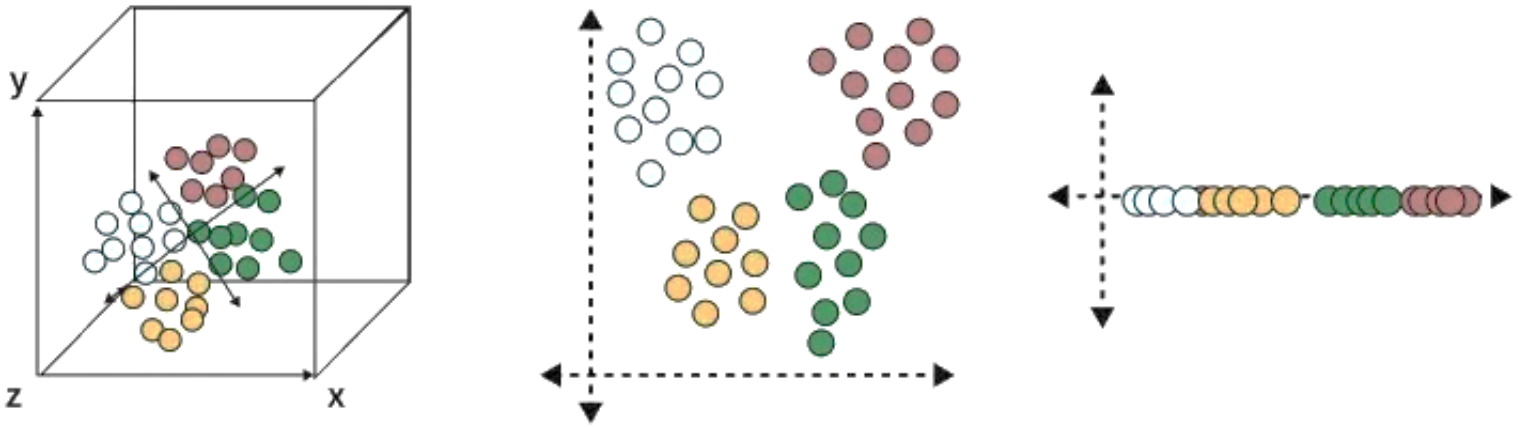
Снижение размерности

Исходные данные представлены в виде матрицы размерности $N \times P$, где:

N – количество объектов (например, образцов, индивидов),

P – количество признаков (размерностей).

Каждый признак соответствует отдельной размерности, и увеличение числа признаков приводит к росту сложности анализа и возникновению проблемы «проклятия размерности».



«Проклятие» размерности

Проклятие размерности (Curse of Dimensionality) – это совокупность проблем, возникающих при работе с данными, где количество признаков (размерность) очень большое. Чем больше признаков, тем сложнее модели обучаться, данные становятся разреженными, и многие алгоритмы начинают плохо работать.

Объем пространства растет экспоненциально с количеством признаков. Например, если есть 10 признаков, и каждый может принимать 10 значений, то всего вариантов – 10^{10}, что уже много.

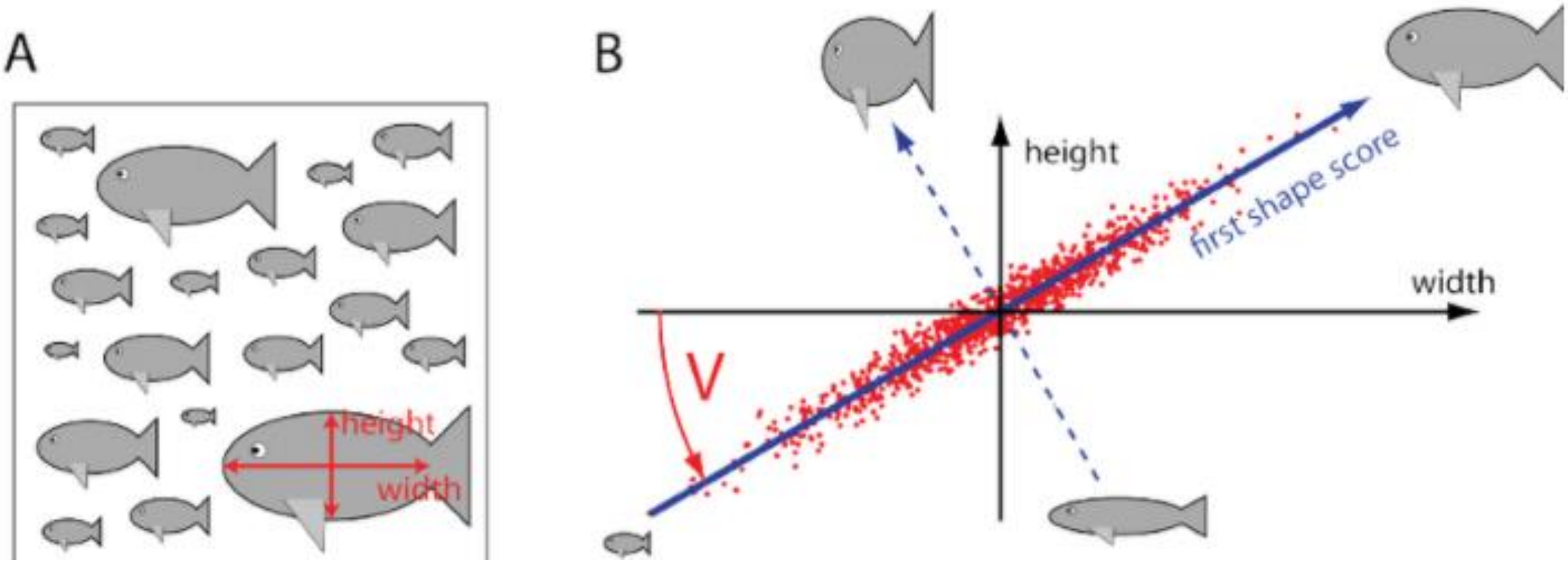
Когда размерность растёт, данные разбиваются по огромному пространству, и «соседей» рядом почти не оказывается. Это мешает алгоритмам, которые опираются на расстояния между точками (например, k-NN, кластеризация). При большом количестве признаков нужно очень много данных, чтобы надежно оценить влияние каждого из них. Если данных мало, модель может выучить шум.

Метод главных компонент (PCA)

Цель метода главных компонент (Principal Component Analysis) – снизить размерность данных, сохранив максимум информации. Исходные признаки преобразуются в меньшее число компонент, рассчитанных на основе их значений, и более информативных, чем отельные признаки. Каждый признак вносит вклад в расчёт каждого компонента, но с разным весом (нагрузка).

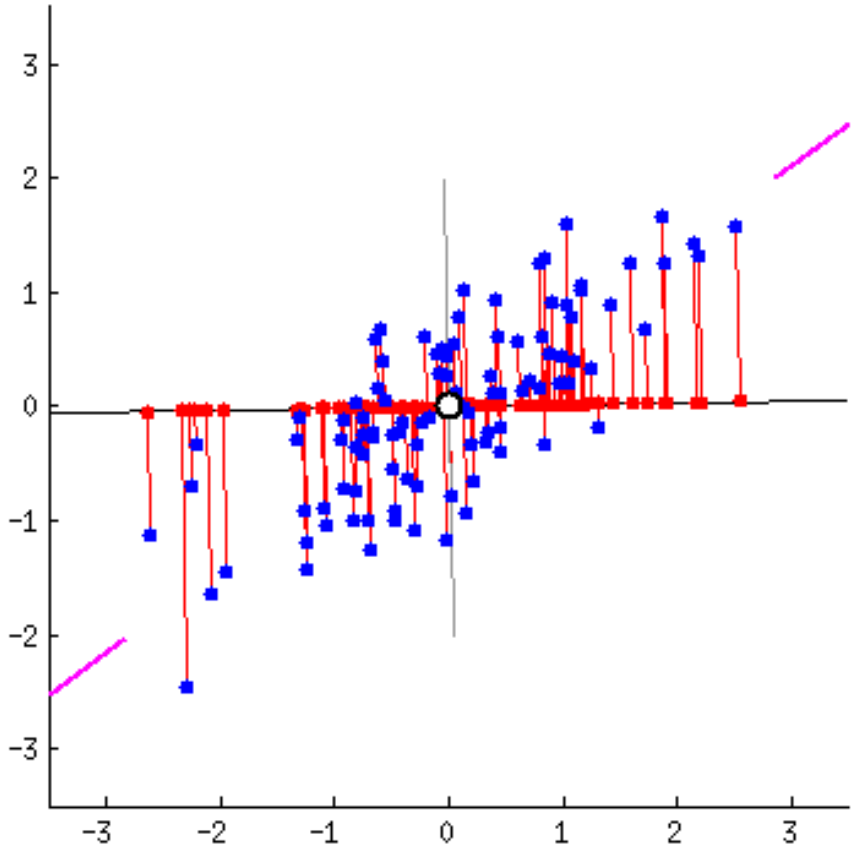
Первая главная компонента аккумулирует в себе максимально возможную долю вариации исходных признаков. Вторая компонента объясняет наибольшую часть оставшейся вариации и при этом не коррелирует с первой.

Метод главных компонент



http://setosa.io/ev/principal-component-analysis/

Метод главных компонент



PCA поворачивает систему координат так, чтобы сумма квадратов расстояний от точек до их проекций (выброшенных перпендикуляров) была минимальна.

Метод главных компонент

	1	2	3	4	5	6	7		200
	Height	Weight	Average blood pressure	Average heart rate	BMI	Cholesterol levels	Average cigarettes/day	...	Sugar levels
Person 1	150	80	140/90	63	36	5.0	0		99
Person 2	174	90	90/60	100	32	4.1	0		95
Person 3	183	109	120/80	95	29	3.6	1		92
Person 4	186	95	123/75	84	28	4.8	5		89
Person 5	170	67	95/60	76	23	2.7	10		100
Person 6	180	82	92/60	78	25	3.7	10		112
Person 7	165	71	124/80	81	26	3.8	0		113
Person 8	172	70	97/70	90	24	3.4	0		100
...									
Person 20	190	75	90/60	78	21	4.2	0		82

200 FACTORS (VARIABLES)

PCA

PC1	PC2	PC3	PC4	PC5
-1	3	-1	4	4
2	4	2	5	5
3	2	4	2	2
4	4	5	-4	-4
5	5	2	2	5
2	5	-4	3	2
-4	-6	5	5	-4
-3	-6	-6	2	5
8	-3	-6	-3	-6

5 PRINCIPAL COMPONENTS

Метод главных компонент: этапы проведения

1. Стандартизация значений исходных признаков (z-стандартизация).
2. Вычисление ковариационной матрицы.
3. Вычисление собственных векторов и собственных значений ковариационной матрицы, чтобы идентифицировать главные компоненты.
4. Определение числа главных компонент (собственные значения превышают 1).
5. Расчёт главных компонент на основе значений исходных признаков.

Метод главных компонент: этап 1

Стандартизация значений исходных признаков (z-стандартизация).

$$z = \frac{value - mean}{standard\ deviation}$$

Метод главных компонент: этап 2

Вычисление ковариационной матрицы.

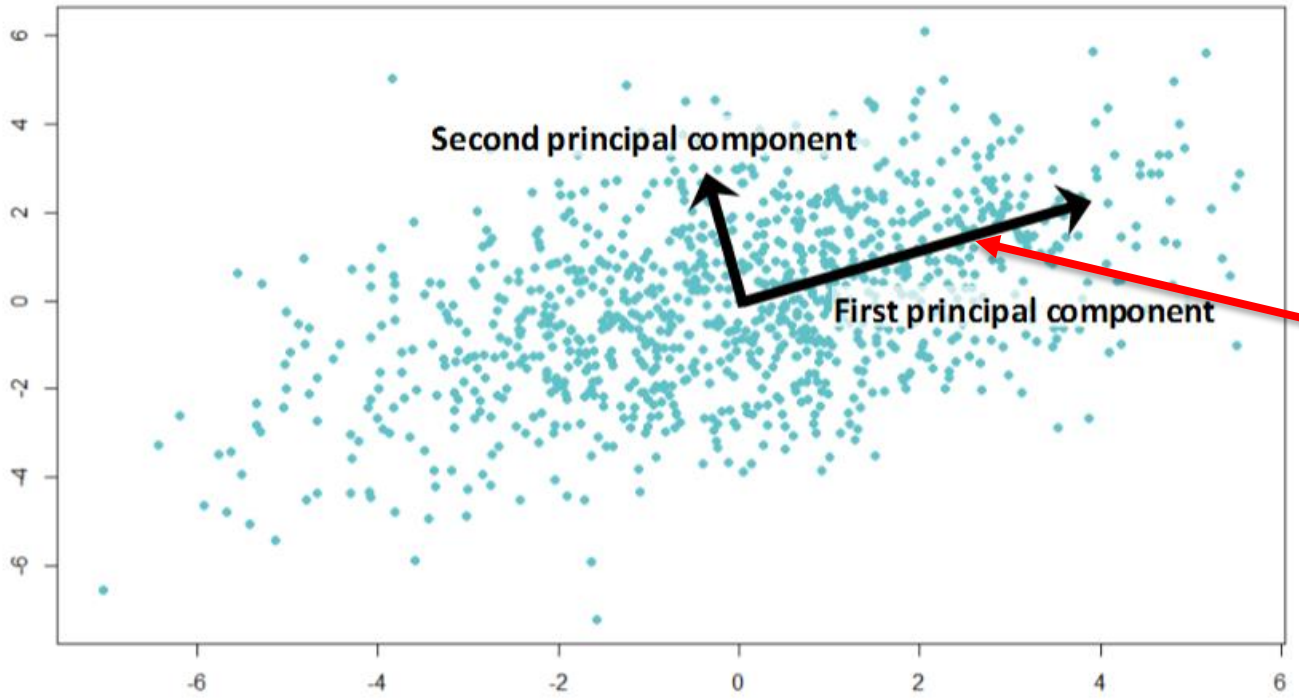
$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

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

$$Correlation = \frac{Cov\,(x,y)}{\sigma x * \sigma y}$$

Метод главных компонент: этап 3

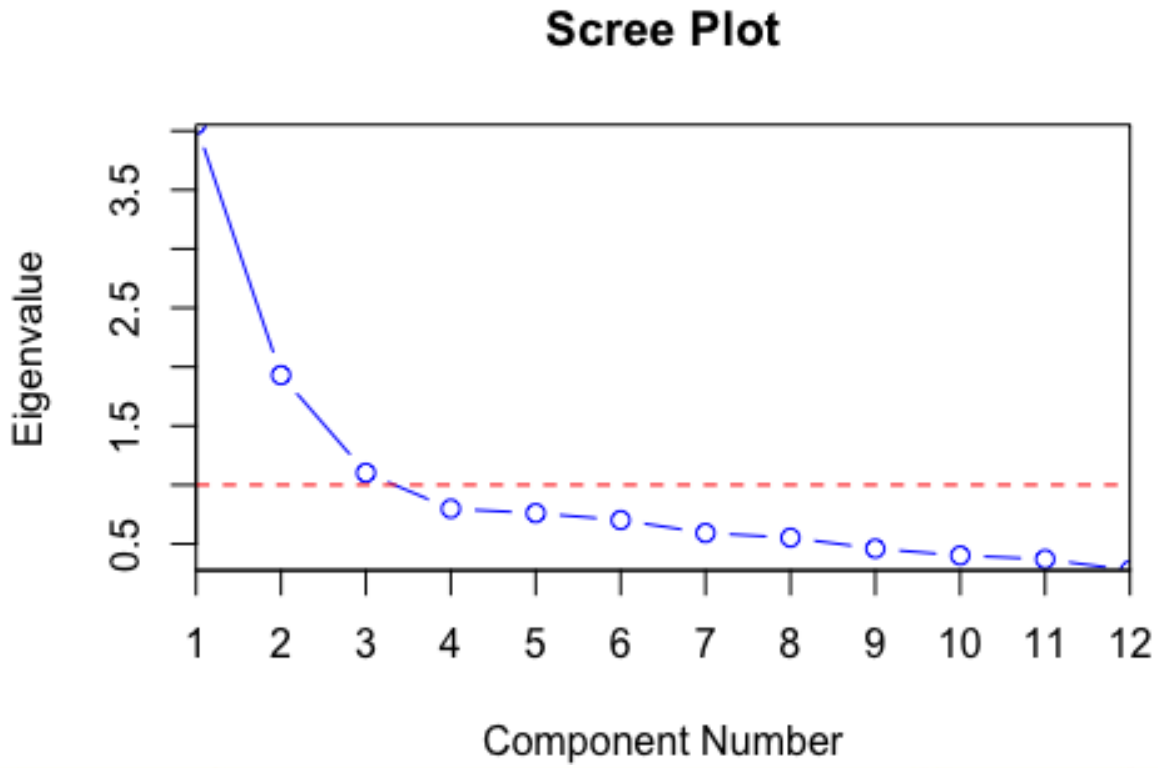
Вычисление собственных векторов и собственных значений ковариационной матрицы, чтобы идентифицировать главные компоненты. Собственные векторы указывают направления, вдоль которых находится максимальная изменчивость данных, а собственные значения показывают, насколько велика эта изменчивость.



Выбираем такую линию, чтобы минимизировать квадраты отклонений точек от линии.

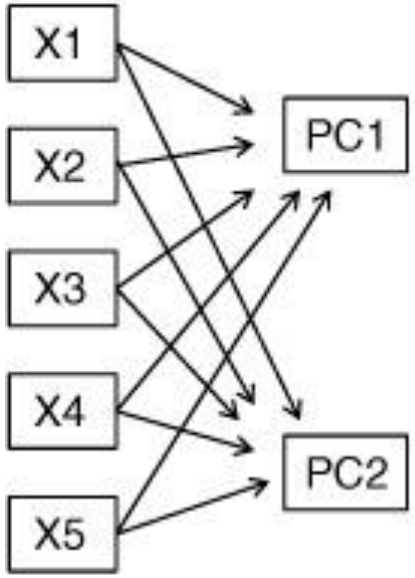
Метод главных компонент: этап 4

Определение числа главных компонент (собственные значения превышают 1). На основе диаграммы каменистой осыпи (scree plot).



Метод главных компонент: этап 5

Расчёт главных компонент на основе значений исходных признаков. Все исходные признаки участвуют в расчёте значений новых главных компонент, но с разными коэффициентами. Фактически главные компоненты представляют собой линейную комбинацию из значений исходных признаков.



$$PC_1 = a_1X_1 + a_2X_2 + \ldots + a_kX_k$$

$$PC_2 = b_1X_1 + b_2X_2 + \ldots + b_kX_k$$

Свойство главных компонент

Главные компоненты подбираются так, чтобы дисперсия каждой главной компоненты на каждом шаге была максимально возможной, поэтому зачастую дисперсия первой главной компоненты вбирает в себя существенную часть суммарной дисперсии всех исходных признаков. Следовательно, несколько первых главных компонент могут вбирать в себя большую долю общей дисперсии.

Применение PCA

Предположим, мы оцениваем сотрудников по 20 показателям: результаты ассессмента, KPI, soft skills, тесты и пр. Многие показатели сильно связаны между собой, например, коммуникация и командная работа, скорость и эффективность.

PCA помогает сократить число метрик, выделив несколько ключевых:

- Профессиональная экспертиза (знания, качество решений, технические навыки)
- Командное взаимодействие (коммуникация, лидерство, обратная связь)
- Общая рабочая эффективность (самоорганизация, обучаемость, стрессоустойчивость)

Вместо 20 разрозненных показателей мы получаем 2–3 понятных измерения, которые отражают основную картину производительности сотрудников.

Пример

Характеристики квартиры:

- Общая площадь
- Число автобусных остановок в шаговой доступности
- Количество жилых комнат
- Число школ в шаговой доступности
- Количество балконов
- Число продуктовых магазинов в шаговой доступности

Какие компоненты можно было бы извлечь?

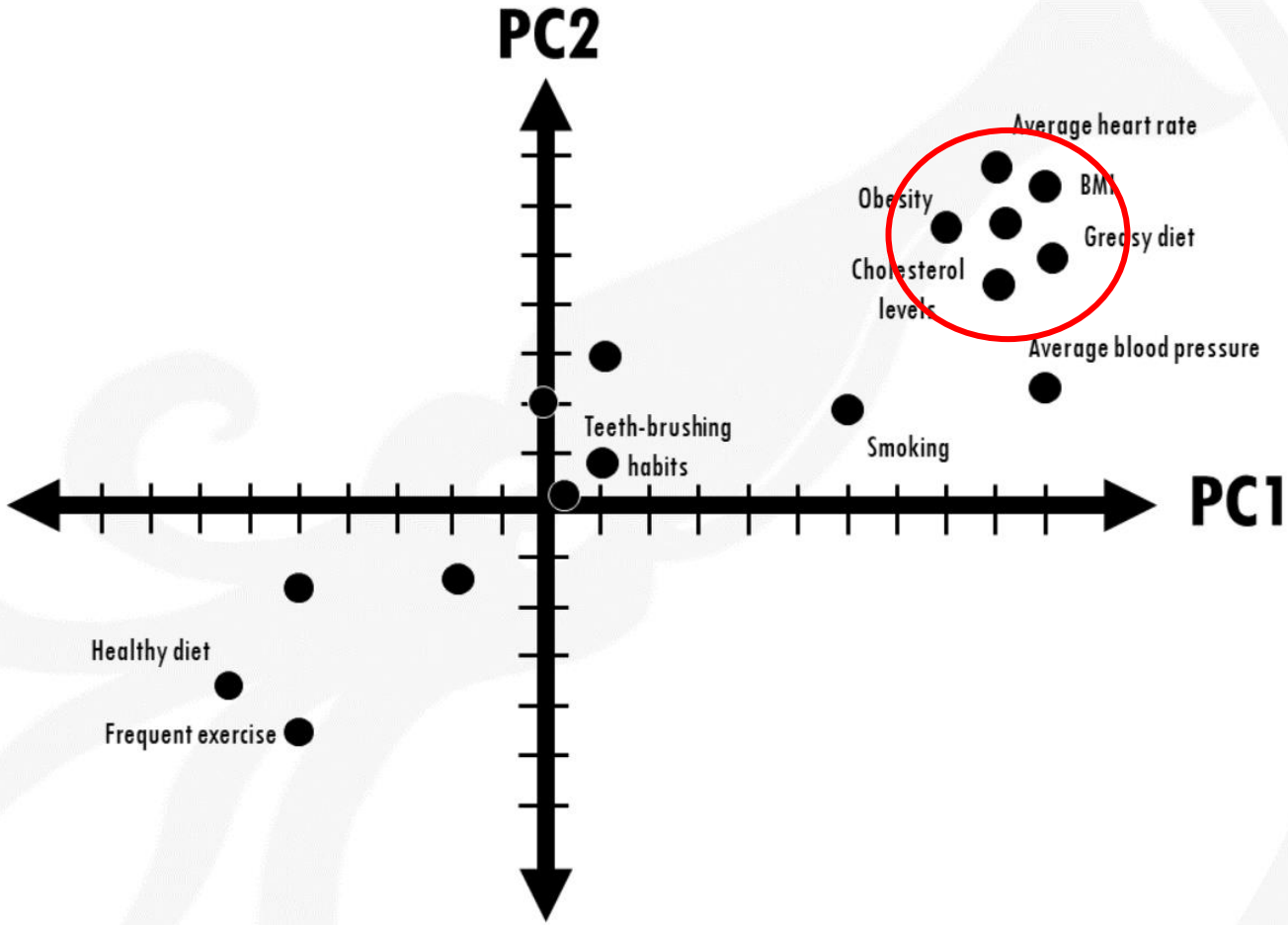
Пример

Характеристики ресторана:

- вкус еды
- температура еды
- время ожидания
- чистота
- свежесть еды
- поведение персонала

Какие компоненты можно было бы извлечь?

Графическое представление



Признаки, которые сильно коррелируют, группируются вместе. У них схожие нагрузки.

Нагрузки компонент

Нагрузки компонент (component loadings) – это коэффициенты, которые показывают, насколько сильно каждый исходный признак связан с каждой главной компонентой. Можно рассматривать как вклад признака в компоненту по знаку и величине.

$|\ loading\ | > 0.7$ – сильная связь

$0.4–0.7$ – умеренная

< 0.3 – слабая

Отрицательная нагрузка – образная взаимосвязь между исходным признаком и компонентой.

Пример

Это названия для компонент, придуманные на основе матрицы нагрузок

	Cost	IT	Org
The product benefits don't outweigh the cost	0.68	0.06	-0.27
Price is prohibitive	0.49	-0.16	0.13
Overall implementation costs	0.41	0.38	0.14
We do not have sufficient technical resources	-0.21	0.58	-0.12
Our IT department cannot support your product	0	0.38	-0.25
Product is not consistent with our business strategy	0.01	-0.03	0.45
We can't reach a consensus in our organization	-0.03	-0.07	0.57
I need to develop an ROI, but cannot or have not	-0.06	0.17	0.62
Your product does not have a feature we require	-0.4	0.05	0.08
We are locked into a contact with another product	-0.21	0.08	-0.27
Other (please specify)	-0.5	-0.06	-0.1
We have no reason to switch	-0.07	-0.77	-0.25

Пример из HR

PC1 «Опыт и компенсация»

PC2 «Удовлетворённость и баланс»

PC3 «Активность и развитие»

PC4 «Перегрузка и производительность»

Матрица нагрузок (Loadings matrix)

Переменная	PC1	PC2	PC3	PC4
Age	0.56	-0.12	-0.30	0.02
Salary	0.63	-0.18	0.45	-0.11
YearsAtCompany	0.60	-0.10	0.22	-0.05
JobSatisfaction	-0.15	0.82	-0.12	0.09
WorkLifeBalance	-0.22	0.77	-0.55	0.04
TrainingTimesLastYear	0.10	0.28	0.69	-0.42
NumberOfProjects	0.05	-0.03	0.70	0.48
Overtime	-0.35	0.07	-0.20	0.85
PerformanceRating	0.42	-0.05	0.07	0.62
DistanceFromHome	0.18	-0.55	0.20	0.28

Размышления про PCA

При проведении PCA важно ответить на три вопроса:

1) Возможно ли вообще формирование компонент на основе выбранных признаков?

2) Сколько информации мы теряем?

3) Как интерпретировать главные компоненты?

Размышления про PCA

1) Возможно ли вообще формирование компонент на основе выбранных признаков?

Если признаки слабо коррелируют или структура их взаимосвязей не выражена, применение PCA может быть нецелесообразным. Если все признаки сильно коррелируют между собой, основная вариация будет описываться одной главной компонентой. Поскольку PCA основан на линейных зависимостях, при наличии нелинейных связей лучше использовать другие методы снижения размерности, например t-SNE.

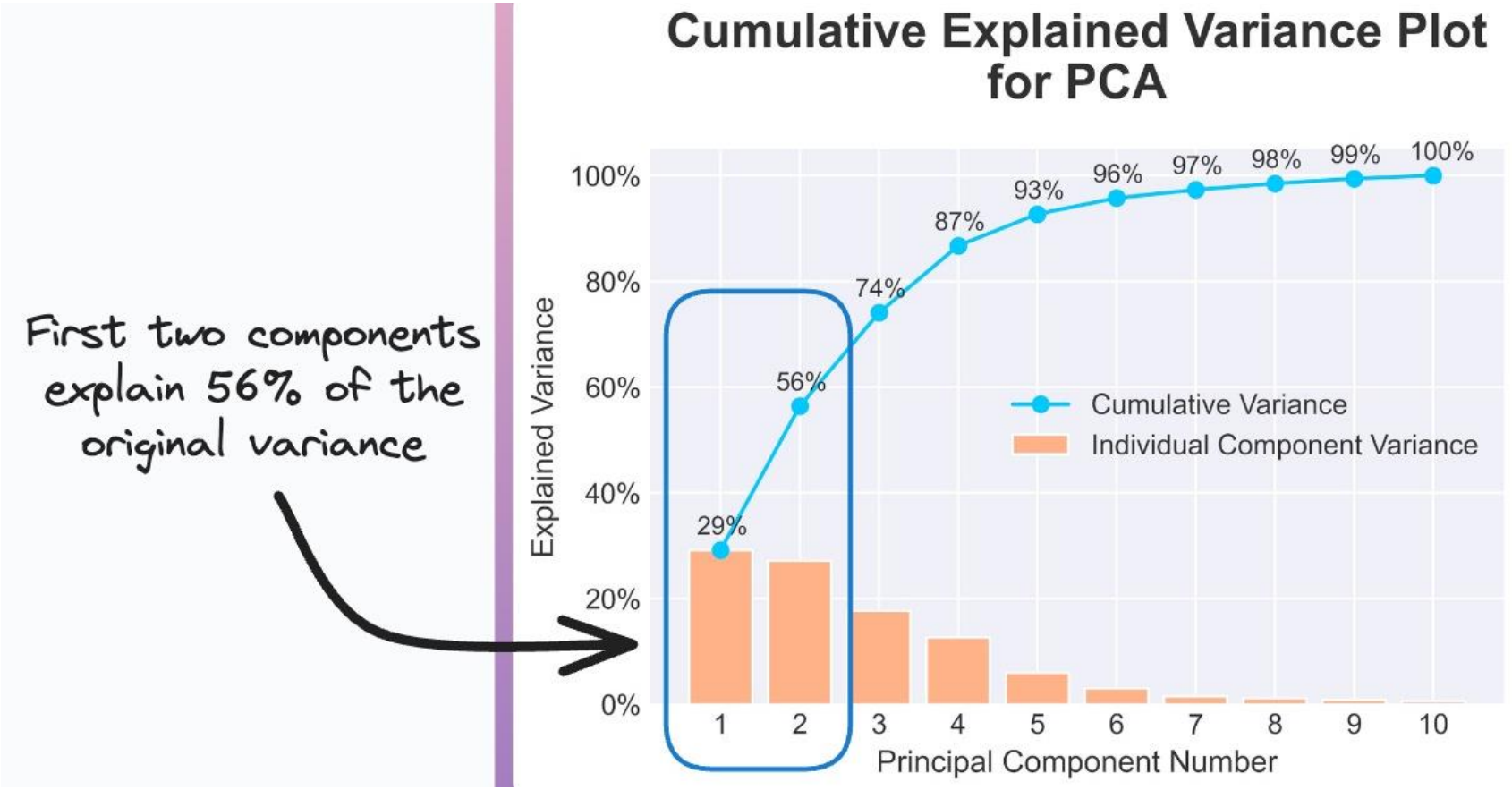
Размышления про PCA

2) Сколько информации мы теряем?

При применении PCA необходимо найти компромисс между снижением размерности и потерей информации. Обычно оценивают долю объяснённой дисперсии: достаточно ли, например, 60% при двух компонентах или стоит добавить третью, чтобы повысить этот процент. Согласно правилу Кайзера, целесообразно сохранять только те компоненты, собственные значения которых больше 1, так как они объясняют вариацию, превышающую вклад одного исходного признака.

Размышления про PCA

2) Сколько информации мы теряем?



Размышления про PCA

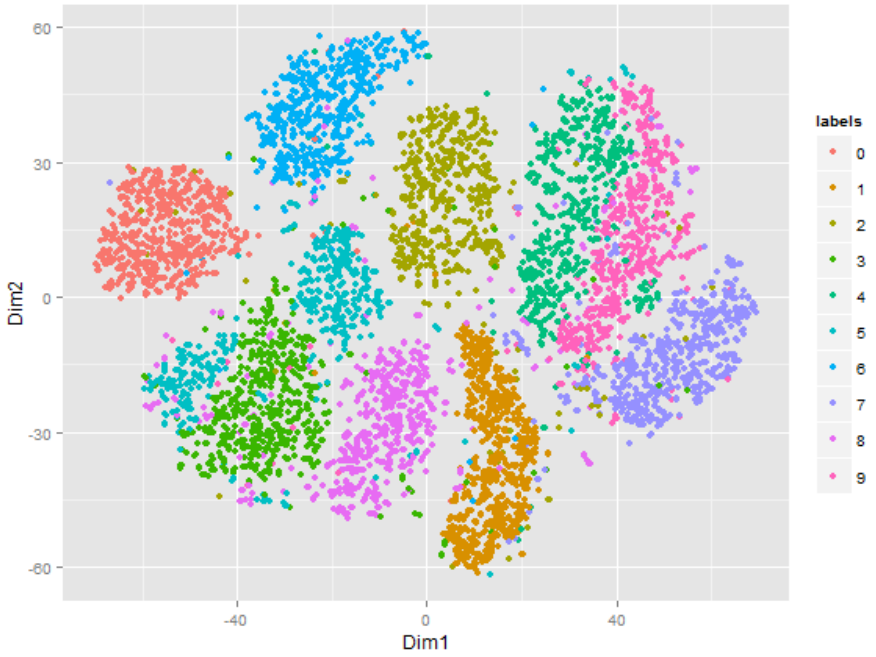
3) Как интерпретировать главные компоненты?

В отличие от исходных признаков с понятными единицами измерения, главные компоненты не имеют интерпретируемой шкалы. Их смысл определяется через нагрузки – корреляции компонент с исходными признаками. Если компонента сильно связана с признаками, характеризующими, например, доход, её можно интерпретировать как обобщённый показатель уровня дохода.

Важно помнить, что первая компонента объясняет наибольшую вариацию данных, но не обязательно является наиболее содержательно значимой. Она выбирается исключительно по математическому критерию – максимизация дисперсии данных. Однако высокая вариативность ≠ высокая смысловая важность для конкретной исследовательской задачи.

t-SNE

t-SNE (t-distributed Stochastic Neighbor Embedding) – метод снижения размерности и визуализации данных, который позволяет сохранить локальные структуры данных и обнаруживать нелинейные зависимости. Основная идея заключается в том, чтобы преобразовать исходные данные таким образом, чтобы схожие объекты в исходном пространстве сохраняли свою схожесть и в новом, сниженном пространстве. PCA сохраняет глобальную дисперсию, но плохо выявляет сложные нелинейные структуры. t-SNE же ориентирован на сохранение локальной структуры (кто с кем сосед).



t-SNE

t-распределение Стьюдента используется на этапе проекции в 2D/3D, чтобы кластеры лучше разделялись.

Stochastic (стохастический) означает, что алгоритм использует случайность, например, при инициализации и в процессе оптимизации. Поэтому результат может отличаться от запуска к запуску, если явно не зафиксировать random seed.

Neighbor (сосед) означает, что алгоритм в первую очередь концентрируется на ближайших соседях, глобальные расстояния вторичны. Это помогает сохранить локальные кластеры.

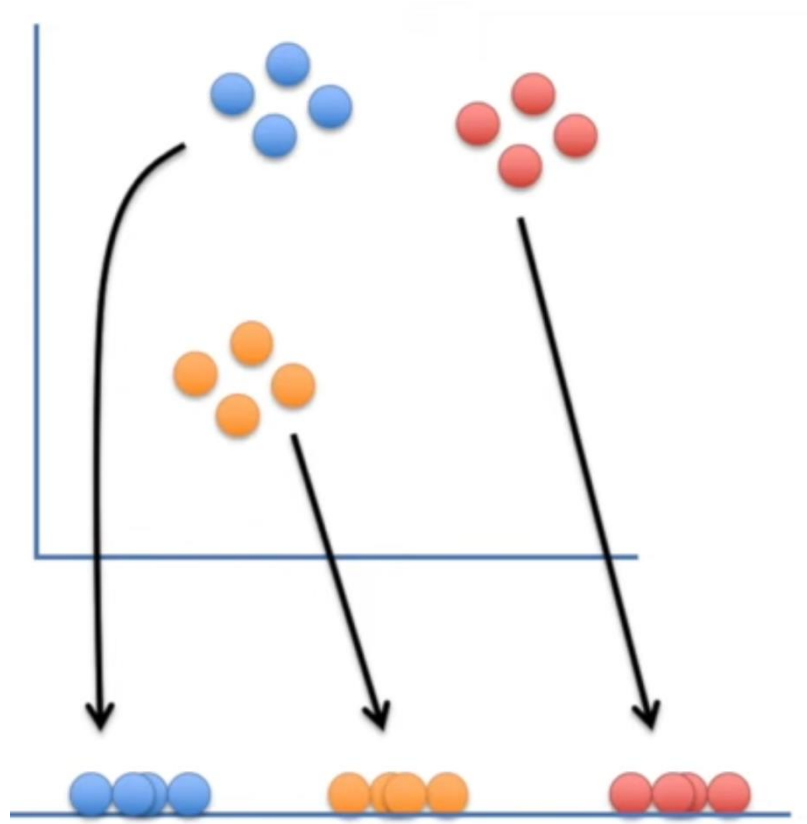
Embedding (встраивание) означает, что алгоритм переносит точки из пространства высокой размерности в пространство низкой размерности.

t-SNE – стохастическое встраивание, которое сохраняет ближайших соседей и использует t-распределение Стьюдента в низкоразмерном пространстве для более качественного разделения кластеров.

t-SNE

Рассмотрим пример снижения размерности с 2 до 1.

t-SNE проецирует данные из пространства высокой размерности в пространство меньшей размерности так, чтобы сохранить структуру кластеров: объекты, которые были похожи и находились близко друг к другу в исходных данных, оставались рядом и после проекции, даже если точные расстояния и взаимное расположение разных кластеров при этом искажаются.



t-SNE: шаги проведения

1. Подсчет похожести точек в исходном пространстве: для каждой точки определяют, какие другие точки для неё близкие (через вероятности).

2. Определение размерности выхода (обычно 2D или 3D) и случайное размещение точек в этом пространстве.

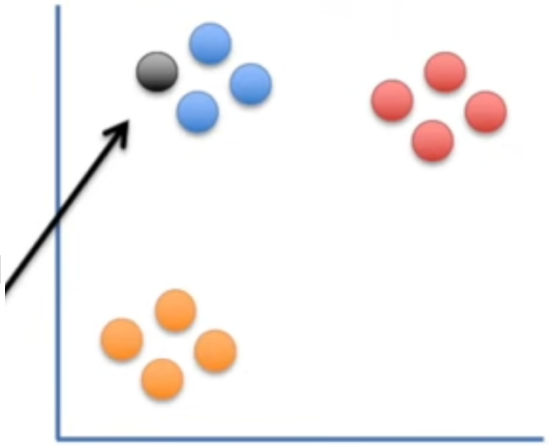
3. Расчет похожести точек в новом пространстве и сравнение с похожестью в исходном.

4. Итеративное движение точек (градиентным спуск), чтобы похожести совпадали как можно лучше.

5. Остановка когда структура локальных кластеров стабилизируется и точки перестают сильно меняться.

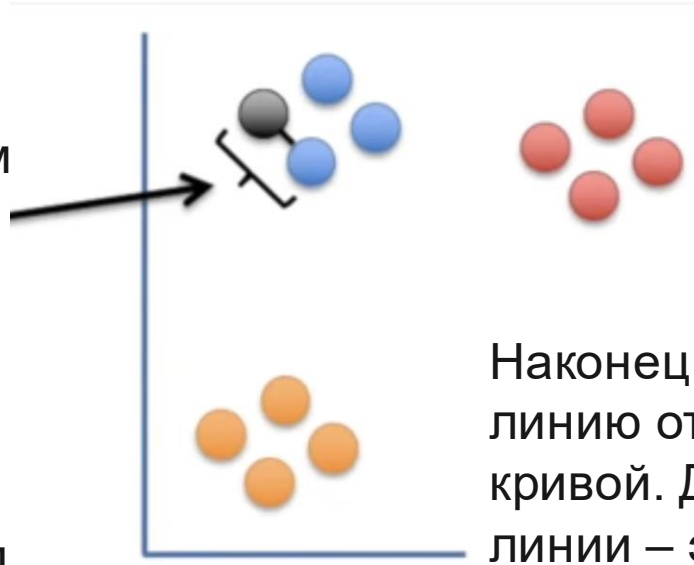
t-SNE: расчёт похожести точек в исходном пространстве

Выбираем точку и рассчитываем похожесть между ней и всеми остальными



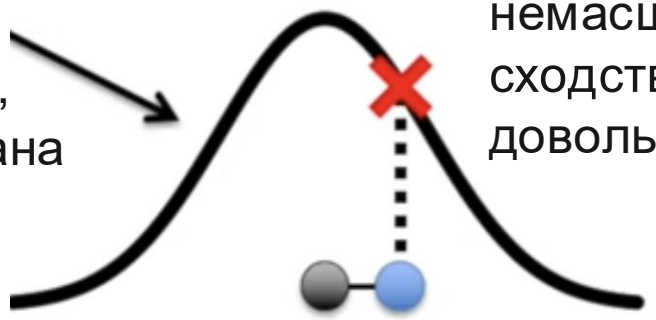
t-SNE: расчёт похожести точек в исходном пространстве

Сначала измеряем расстояние между двумя точками



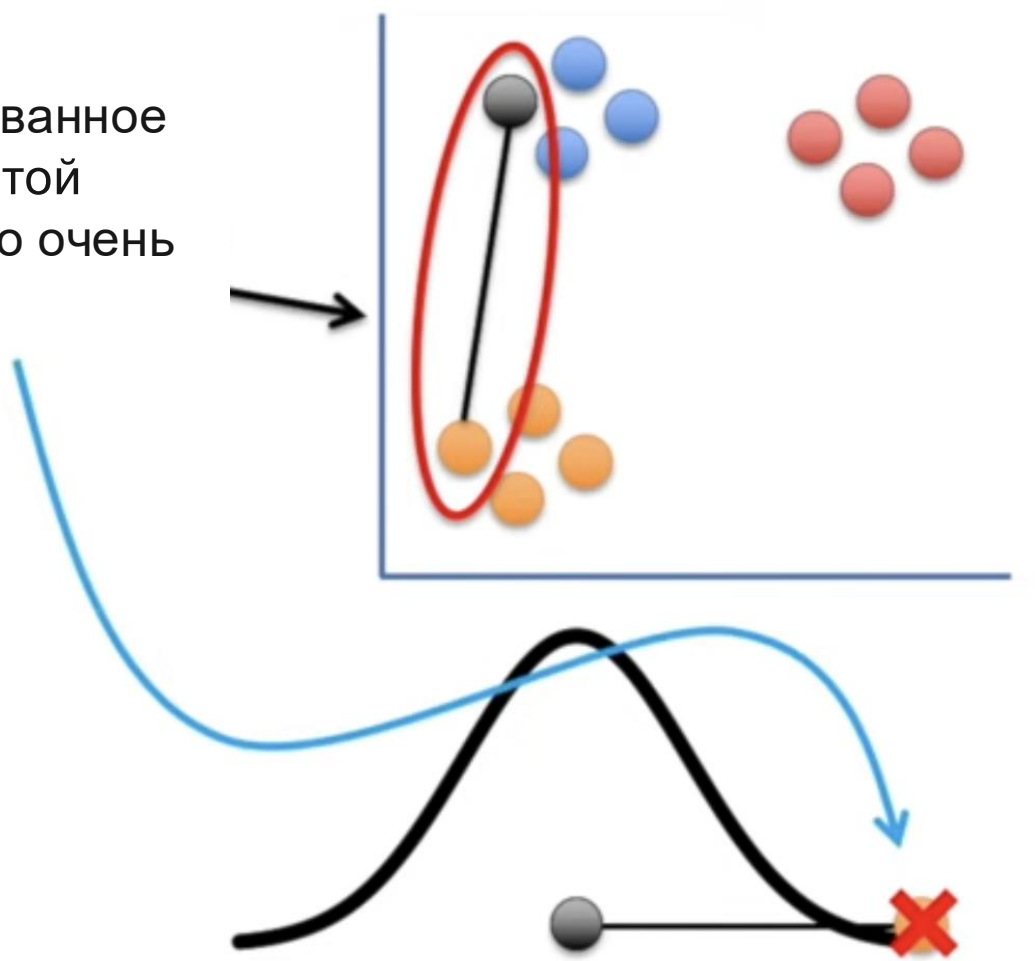
Наконец проводим линию от точки до кривой. Длина этой линии – это немасштабированное сходство, оно довольно высокое

Затем откладываем это расстояние на нормальной кривой, которая центрирована на анализируемой точке



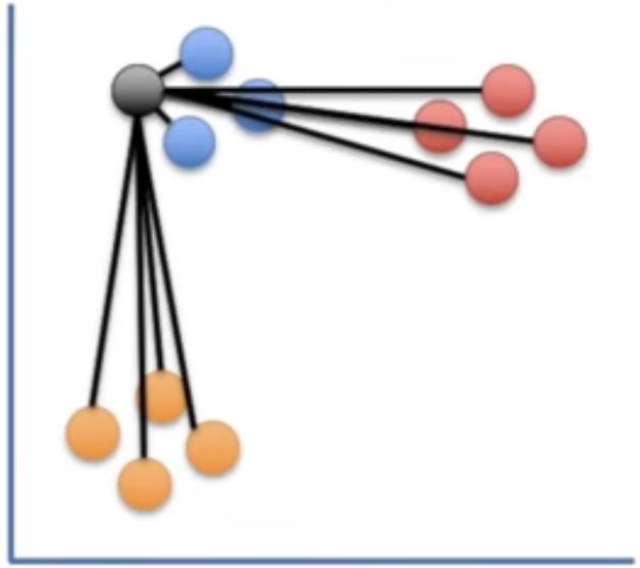
t-SNE: расчёт похожести точек в исходном пространстве

Посчитаем немасштабированное сходство для этой пары точек, оно очень низкое.

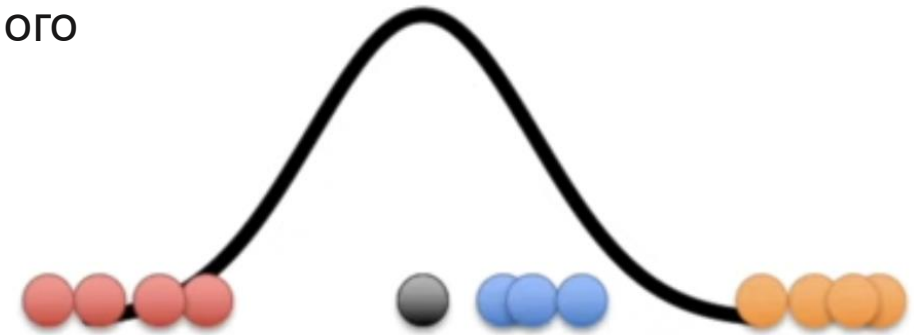


t-SNE: расчёт похожести точек в исходном пространстве

Проводим расчеты для всех пар точек

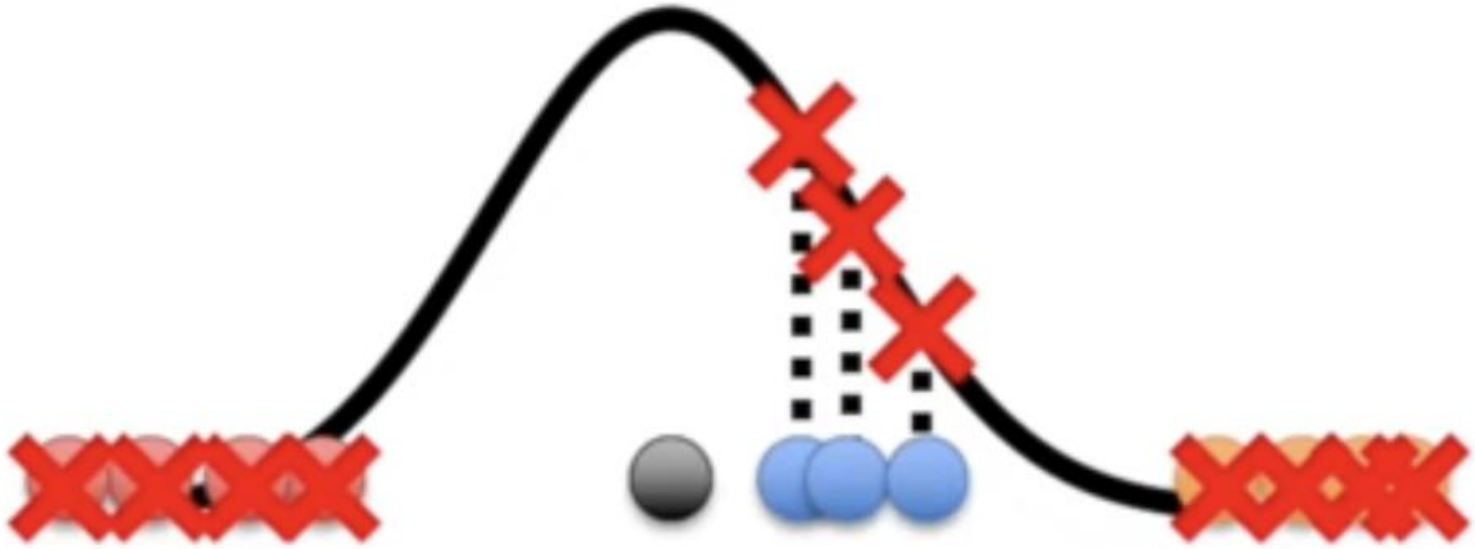


И изображаем их на кривой нормального распределения



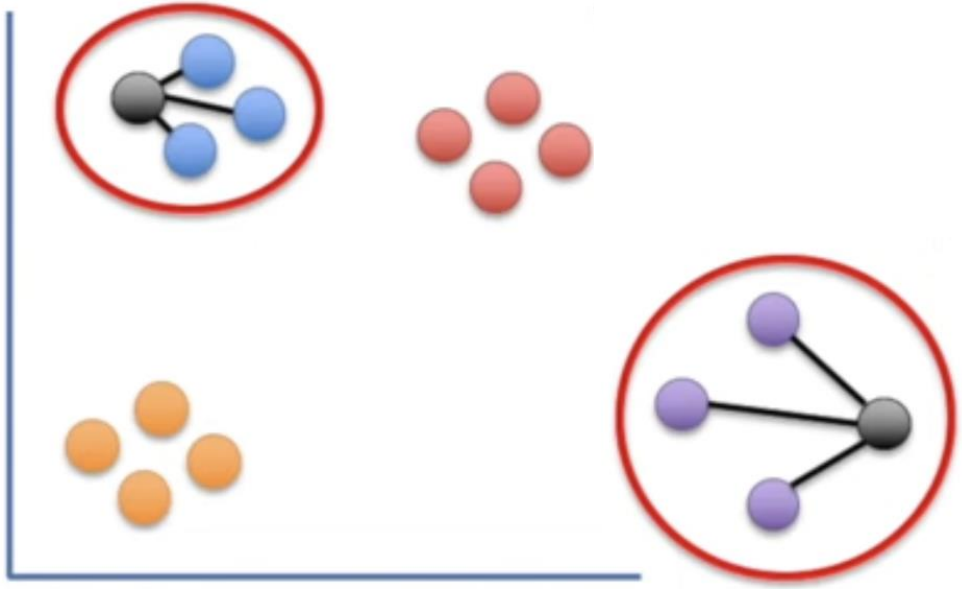
t-SNE: расчёт похожести точек в исходном пространстве

Дальше нужно масштабировать значения схожести, чтобы они в сумме равнялись единице.

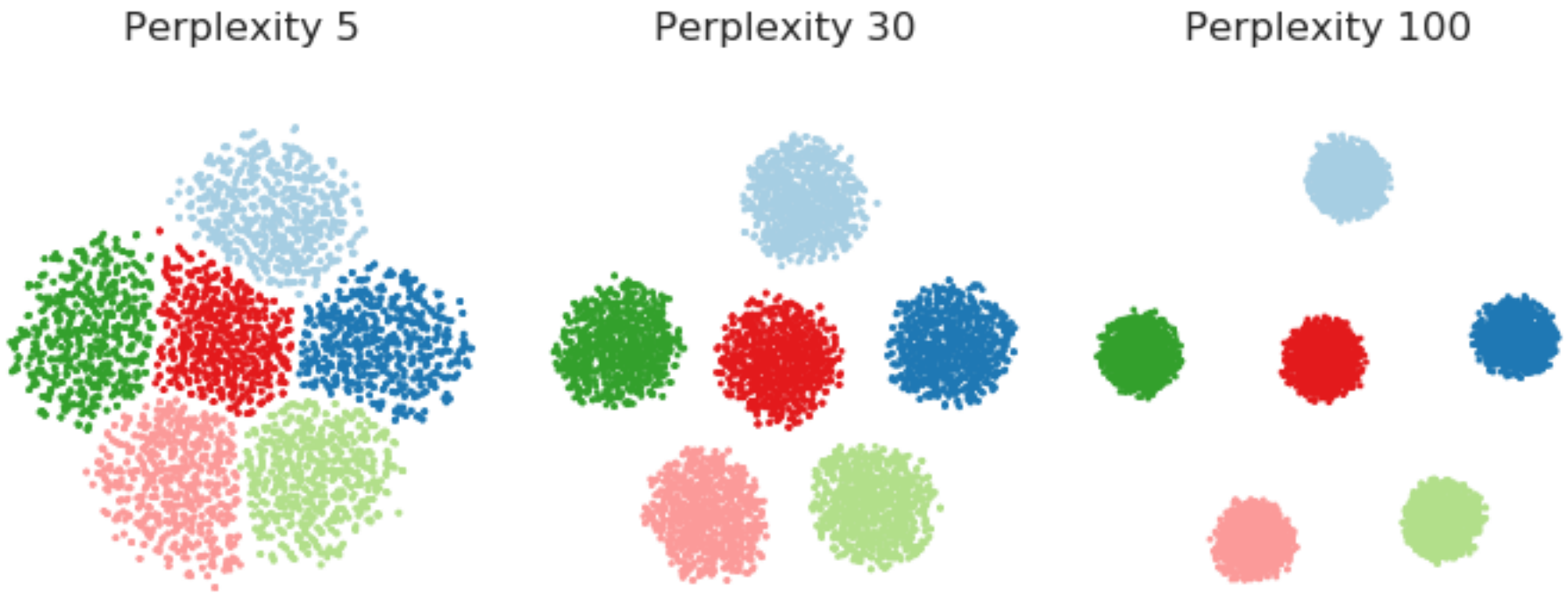


t-SNE: гиперпараметр perplexity

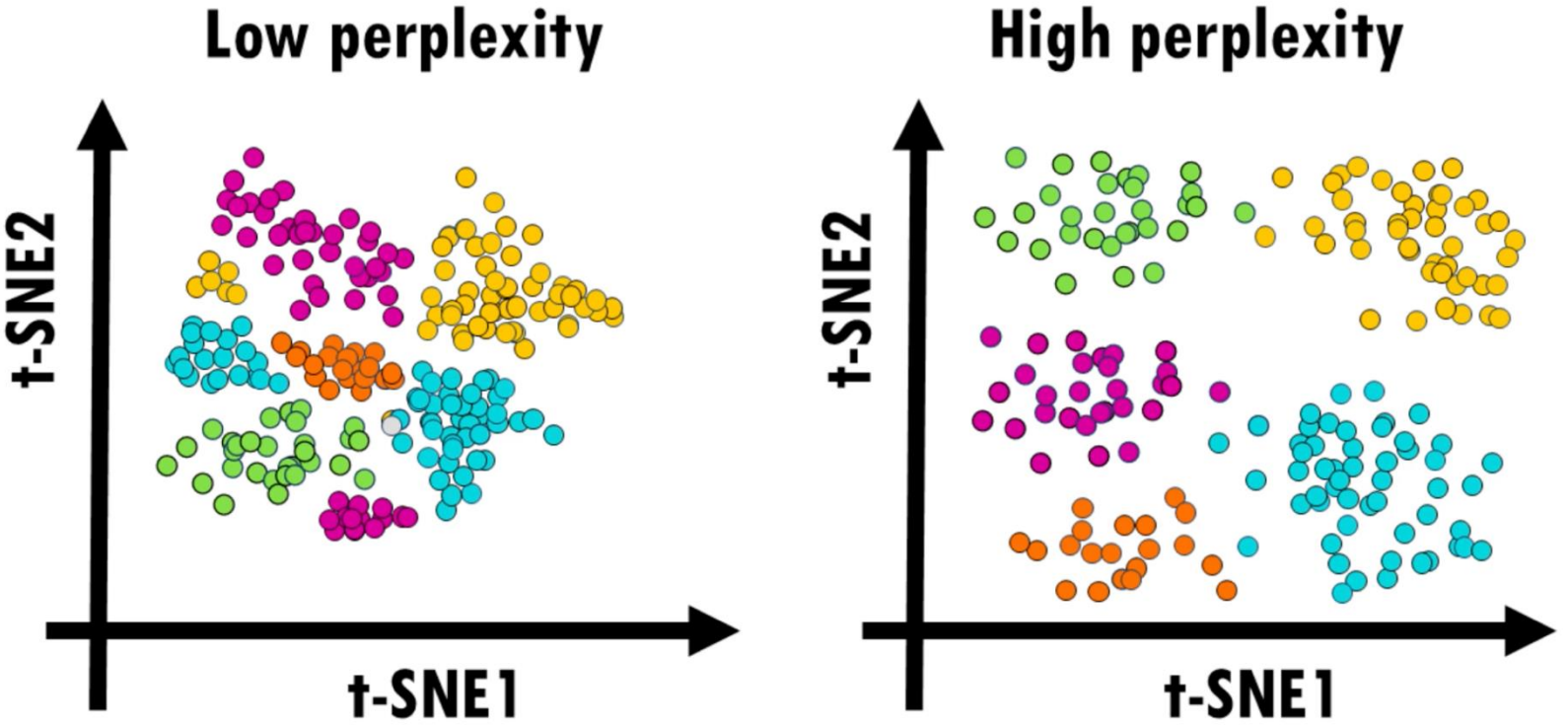
У t-SNE есть параметр perplexity (неопределенность), регулирующий число соседей, учитываемых при построении распределения вероятностей. Связан с тем, насколько плотно точки должны быть сгруппированы. Если perplexity маленькая, то алгоритм видит только очень близких соседей, а следовательно фокусируется на очень локальных структурах; большая – учитывает более дальние точки.



t-SNE: гиперпараметр perplexity



t-SNE: гиперпараметр perplexity



t-SNE: гиперпараметр perplexity

При большой perplexity:

- σ_i увеличивается,
- гауссиана становится более пологой,
- учитывается больше соседей.

При малой perplexity:

- σ_i уменьшается,
- распределение становится более острым,
- учитываются только ближайшие соседи.

σ_i - стандартное отклонение (ширина) гауссовского распределения,

t-SNE: гиперпараметр perplexity

Маленькая perplexity (5-10):

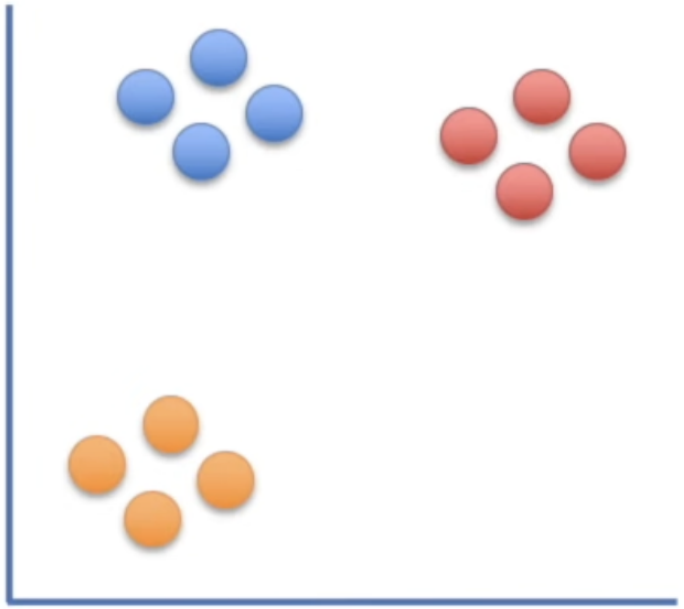
- сохраняется очень локальная структура;
- много маленьких кластеров;
- может разбить один большой кластер на несколько.

Большая perplexity (50–100):

- больше глобальной структуры;
- кластеры крупнее;
- меньше дробления.

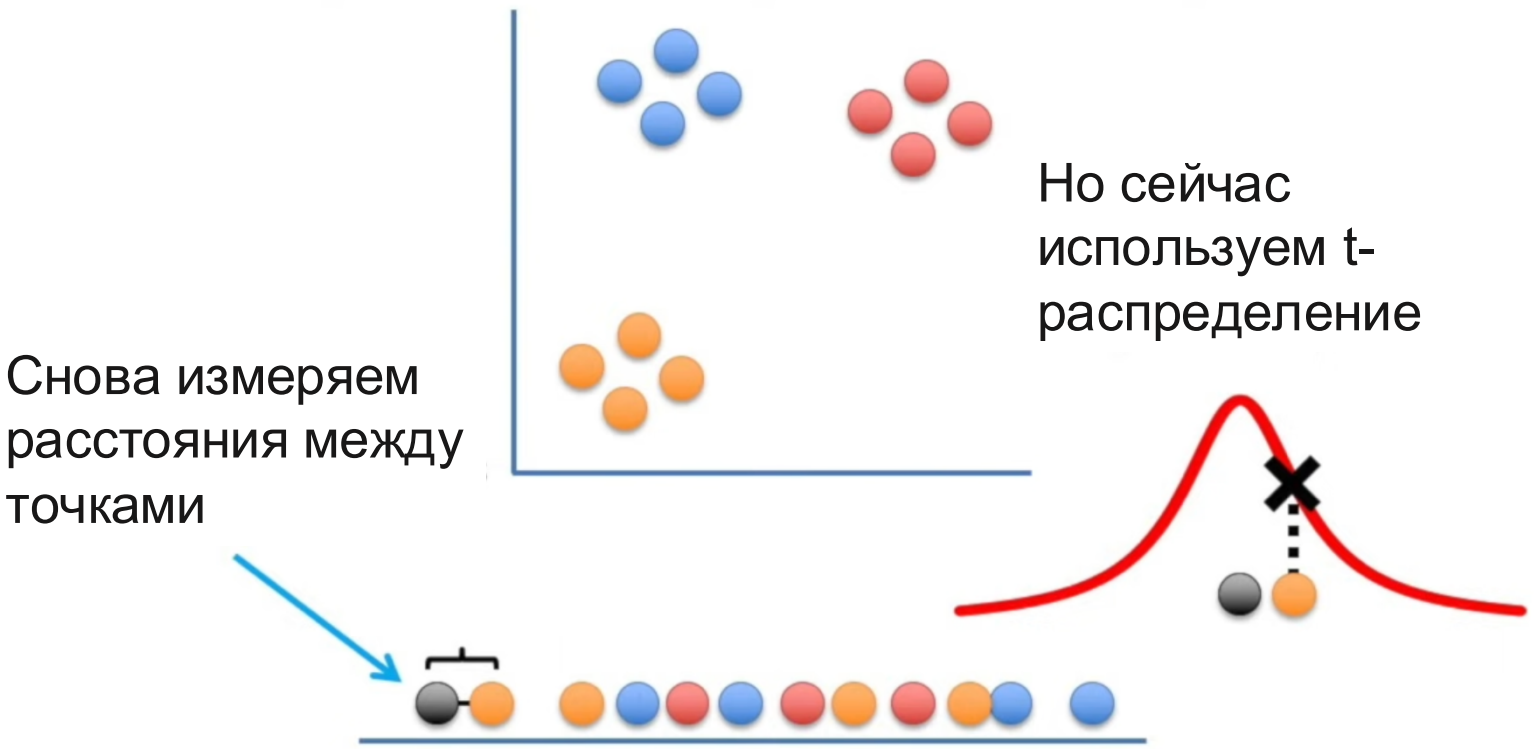
t-SNE: расчёт похожести точек в новом пространстве

Теперь мы случайным образом проецируем данные на числовую прямую.



И вычисляем показатели схожести для точек на числовой прямой.

t-SNE: расчёт похожести точек в новом пространстве



t-SNE: используемые распределения

Расстояния между точками в исходном многомерном пространстве напрямую не сохраняются при проекции в низкую размерность. В низкоразмерном пространстве расстояния между удалёнными точками искусственно увеличиваются, чтобы лучше разделить кластеры и сохранить локальную структуру данных. Поэтому в низкой размерности используется t-распределение Стьюдента, у которого более тяжёлые хвосты и менее выраженный пик, чем у нормального распределения. Это позволяет уменьшить слипание удалённых точек.

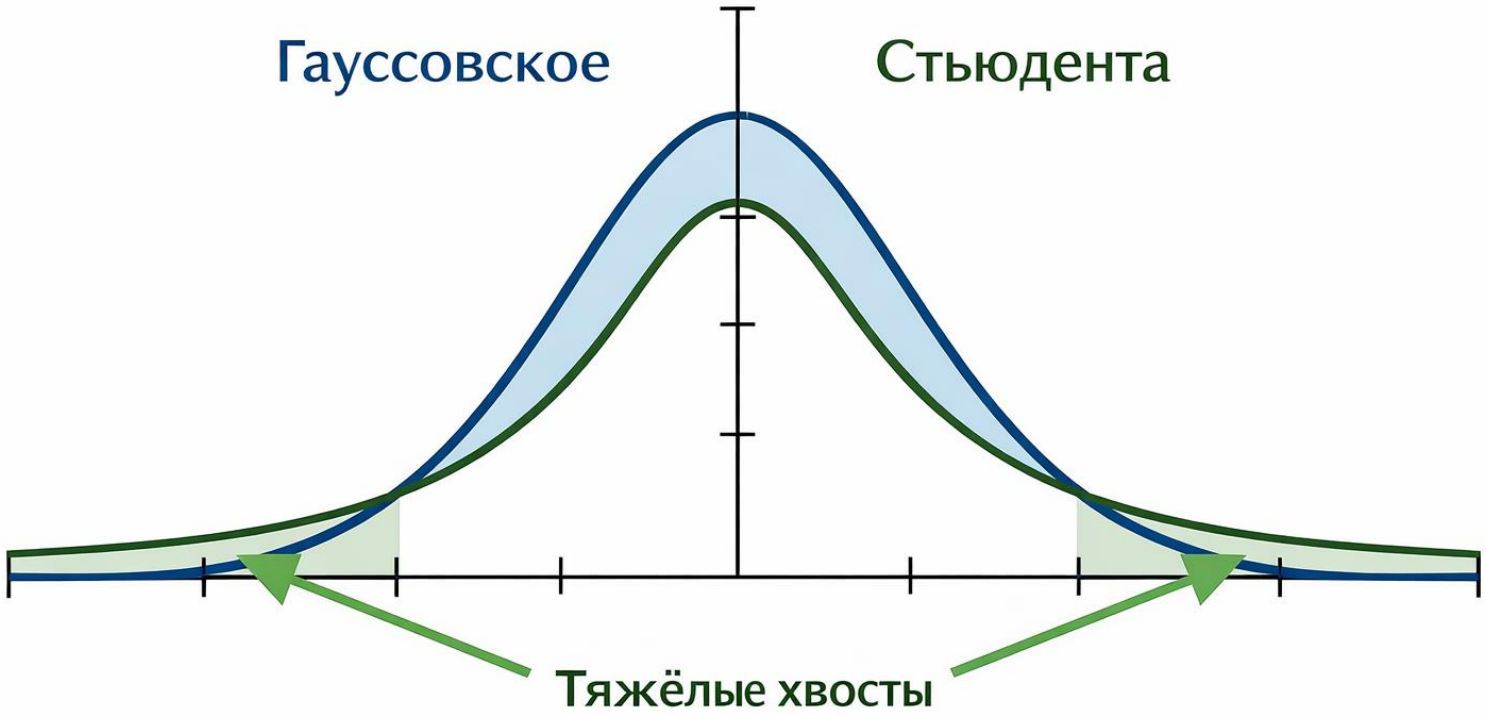
t-SNE: используемые распределения

В 100-мерном пространстве вокруг точки может быть очень много места, на плоскости (2D) места гораздо меньше. Когда мы переходим из 100D в 2D удалённые точки начинают вынужденно сближаться, потому что в 2D просто нет места их всех разнести так же далеко.

В исходном пространстве для оценки близости использует нормальное распределение. Если расстояние увеличивается вероятность довольно быстро стремится к нулю. То есть «далеко» и «очень далеко» почти не различаются. Если использовать его и в 2D, то удалённые точки почти не будут отталкиваться и кластеры будут слипаться.

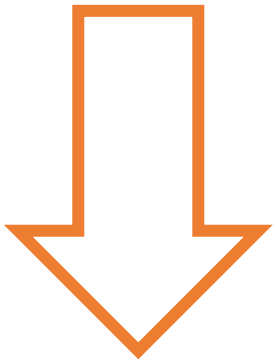
В низкой размерности используется t-распределение. За счет более тяжёлых хвостов происходит менее резкий спад вероятности, за счет чего кластеры раздвигаются.

t-SNE: используемые распределения



t-SNE: оптимизация

Точки в низкоразмерном пространстве итеративно перемещаются так, чтобы распределения вероятностей соседства были максимально похожи на распределения в исходных данных.



t-SNE: функция потерь

t-SNE минимизирует KL-дивергенцию между распределениями P и Q:

$$KL(P \mid\mid Q) = \sum_{i \neq j} P_{ij} \, log \frac{P_{ij}}{Q_{ij}}$$

P_{ij} – соседства в исходном пространстве

Q_{ij} – соседства в 2D

KL-дивергенция измеряет, насколько одно распределение вероятностей отличается от другого. Если точки, которые близки в высокоразмерном пространстве, далеко в низкоразмерном, KL-дивергенция растёт.

t-SNE перемещает точки в 2D/3D, чтобы минимизировать эту функцию потерь.

Особенности t-SNE

Алгоритм сохраняет:

- локальную структуру;
- ближайших соседей;
- кластеризацию.

Алгоритм НЕ сохраняет:

- глобальные расстояния;
- реальные размеры кластеров;
- межкластерные расстояния.

Если два кластера далеко друг от друга на картинке – это не обязательно значит, что они были далеко в исходном пространстве.

Когда использовать t-SNE?

Хорошо подходит для визуализации, исследования структуры данных, поиска кластеров. Но не подходит для количественной интерпретации расстояний и последующего обучения моделей.

Алгоритм не выделяет кластеры, он лишь визуализирует. Если нужны конкретные кластеры, то нужно использовать алгоритмы кластеризации.

В HR используется не для предсказаний, а для анализа и визуализации данных о сотрудниках, выявления скрытых закономерностей.

Спасибо за внимание!