

# Матричные разложения в машинном обучении, рекомендательные и тематические системы

Петров Олег, Лысенко Иван, Карлов Владимир  
БПМИ193



## Напоминание

Если  $A \in \mathbb{C}^{m \times n}$ , то эрмитово-сопряженная к  $A$  матрица  $A^* \in \mathbb{C}^{n \times m}$  такова, что

$$(A^*)_{ij} = \overline{A_{ji}}$$

, где  $\bar{z} = a - bi$  – комплексно-сопряженное число к  $z = a + bi$ .

$U \in \mathbb{C}^{n \times n}$  называется унитарной матрицей, если

$$U^*U = UU^* = I$$



## Что такое SVD?

Пусть  $A \in \mathbb{C}^{m \times n}$ ,  $\text{rank } A = r$ . Тогда  $\exists U \in \mathbb{C}^{m \times m}, V \in \mathbb{C}^{n \times n}$  – унитарные и  $\sigma_1 \geq \dots \geq \sigma_r > 0$  – сингулярные числа, такие что

$$A = U \Sigma V^*$$

, где

$$\Sigma = \left( \begin{array}{ccccccc} \sigma_1 & & & & & & \\ & \ddots & & & & & \\ & & \sigma_r & & & & \\ & & & 0 & & & \\ & & & & 0 & & \\ & 0 & & & & \ddots & \\ & & & & & & 0 \\ \hline & & & & & & 0 \end{array} \right)$$



# Сокращенные представления SVD

Пусть  $M \in \mathbb{C}^{m \times n}$ .

- Thin SVD

$$M = U_k \Sigma_k V_k^*$$

, где  $k = \min(m, n)$ . Убирает нижние нулевые строки (или правые нулевые столбцы) из  $\Sigma$ , которые ни на что не влияют.

- Compact SVD

$$M = U_r \Sigma_r V_r^*$$

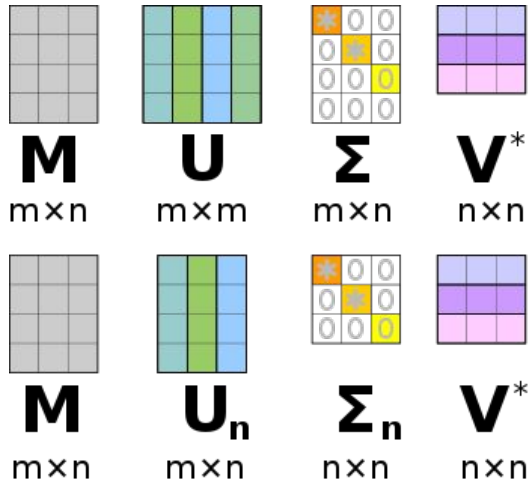
То же самое, что Thin SVD, только мы берем  $k = r$ . Получаем квадратную матрицу, у которой на диагонали стоят только сингулярные числа (никаких нулей).

- Truncated SVD

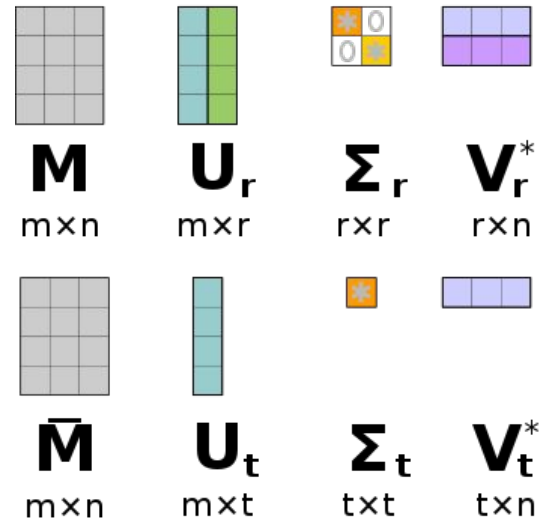
$$\tilde{M} = U_t \Sigma_t V_t^*$$

Аналогично двум предыдущим, только мы оставляем  $t$  самых больших сингулярных значений, а остальные удаляем.

## Сокращенные представления SVD



- 1) Full SVD
- 2) Thin SVD



- 3) Compact SVD
- 4) Truncated SVD



## SVD в задаче наименьших квадратов: полноранговый случай

Пусть  $A \in \mathbb{R}^{m \times n}$ , причем  $m \geq n$  и  $\text{rank } A = n$ . Тогда, в общем случае, вообще говоря, решения системы линейных уравнений  $Ax = b$  не существует. Тогда, мы можем решить задачу


$$J(x) = \|Ax - b\|_2^2 \rightarrow \min_x$$

, чтобы получить такой вектор  $x$ , чтобы  $Ax \approx b$ . Такой метод решения СЛУ называется методом наименьших квадратов.

Как мы помним с курса линейной алгебры, решение задачи наименьших квадратов:

$$x = (A^T A)^{-1} A^T b = A^+ b$$

, где мы определяем псевдообратную Мура-Пенроуза  $A^+ = (A^T A)^{-1} A^T$ .



## SVD в задаче наименьших квадратов: общий случай, определение псевдообратной

Пусть  $A = U\Sigma V^*$ ,  $\text{rank } A = r$ ,

$$\Sigma = \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix}$$

– это полное SVD матрицы  $A$  и

$$\Sigma^+ := \begin{pmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{pmatrix}$$

Тогда, псевдообратная Мура-Пенроуза для матрицы  $A$  определяется как

$$A^+ := V\Sigma^+U^*$$

Можно еще определить через Compact SVD:

$$A^+ := V_r \Sigma_r^{-1} U_r^*$$



## SVD в задаче наименьших квадратов: общий случай, решение

Тогда, все решения задачи  $\|Ax - b\|_2^2 \rightarrow \min_x$  имеют вид

$$x = A^+b + (I - A^+A)y, \forall y$$

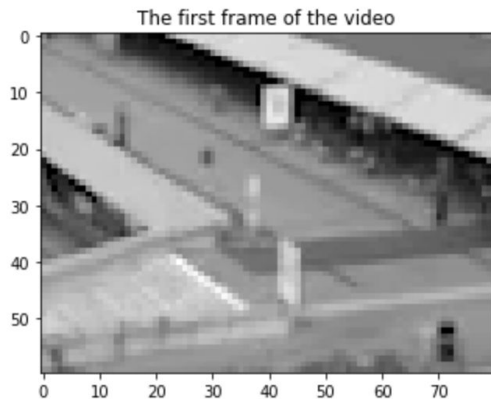
, а  $x_* = A^+b$  имеет среди всех этих решений минимальную 2-ую норму.



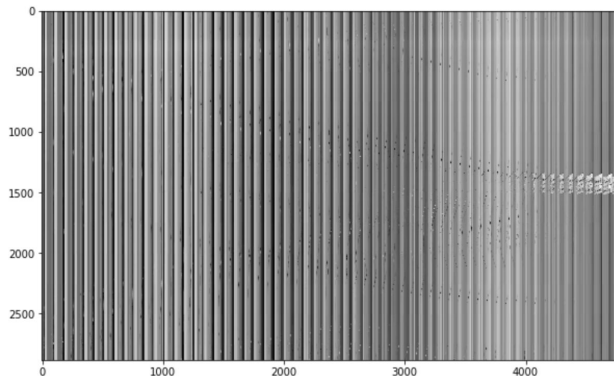
# Пример применения SVD: выделение движущихся частей видео

Видео в виде `np.ndarray`  
размера `(nframes, size_w, size_h)`

image size: 60 x 80,  
number of frames: 2883



`np.reshape`



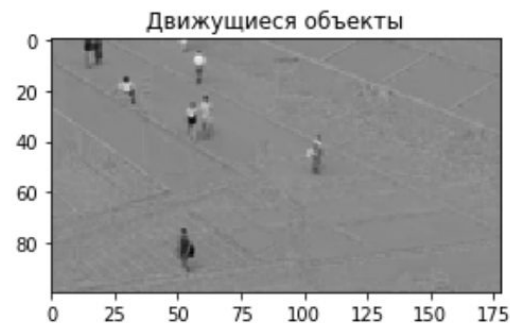
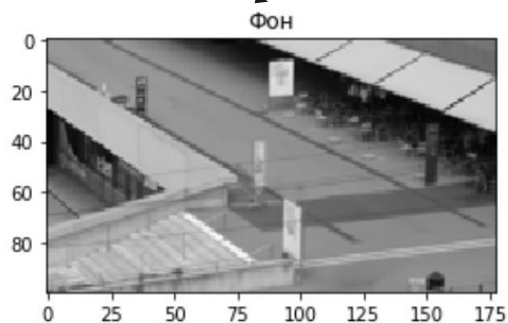
`np.linalg.svd`

# Пример применения SVD: выделение движущихся частей видео

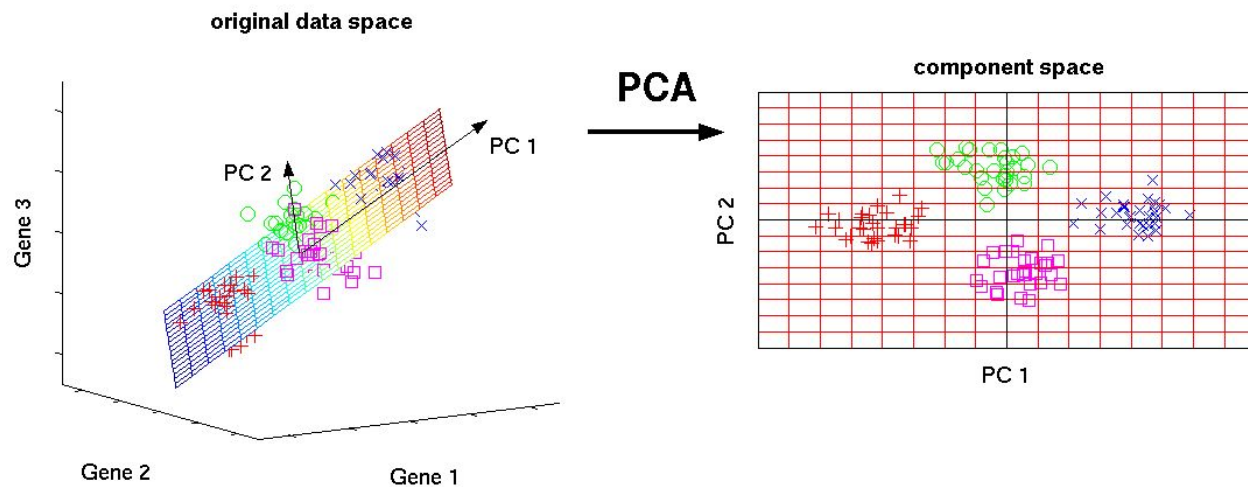
`np.linalg.svd`

→ Truncated SVD матрицы видео ранга 1

Обратный `np.reshape`



# PCA: в чем задача?





# Матрица ковариации

Пусть  $Y = (Y_1, Y_2, \dots, Y_d)^T$  – случайный вектор. Тогда, его ковариационной матрицей называется такая матрица  $K$  размера  $d \times d$ , что

$$K_{ij} = \text{cov}(Y_i, Y_j) = \mathbb{E}[(Y_i - \mathbb{E}Y_i)(Y_j - \mathbb{E}Y_j)]$$

Если  $\mathbb{E}Y = 0$ , то

$$K_{ij} = \mathbb{E}[Y_i Y_j]$$

Пусть  $X$  – матрица объекты-признаки размера  $n \times d$  для случайного вектора  $Y - \bar{Y}$ . Тогда, **с точностью до константы  $C$**  (нам не особенно важно, будет ли там деление на  $n$  или  $n - 1$ ):

$$\hat{K}_{ij} = C \cdot X_i^T X_j \Rightarrow \hat{K} = C \cdot X^T X$$

, где  $\hat{*}$  – оценка случайной величины/матрицы  $*$ .



## Диагонализуемость матрицы ковариации

$$\hat{K}^T = (C \cdot X^T X)^T = C \cdot X^T X = \hat{K}$$

, поэтому матрица  $\hat{K}$  – нормальная. Значит, она также диагонализуема:  $\exists U$  – унитарная, причем ее столбцы – это собственные значения  $\hat{K}$ , и  $\Lambda$  – диагональная матрица с собственными значениями  $\hat{K}$  на диагонали, такие что

$$\hat{K} = U \Lambda U^*$$

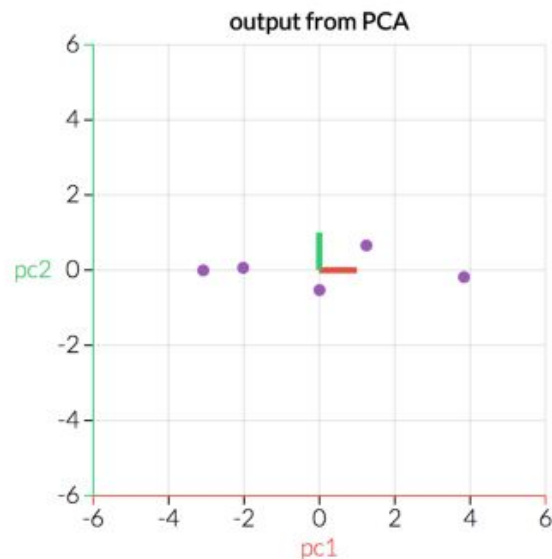
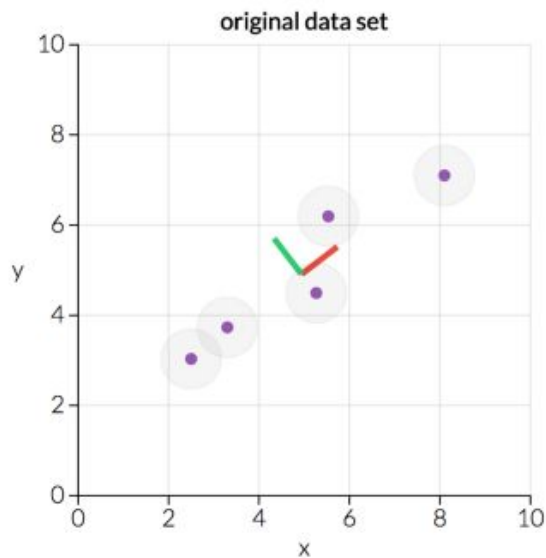


# РСА: формулировка алгоритма, часть 1

Пусть  $X \in \mathbb{R}^{m \times n}$  – матрица объекты-признаки, а  $y \in \mathbb{R}^m$  – вектор ответов. Тогда, алгоритм РСА:

1. Вычитаем из каждого столбца  $X$  его среднее значение.
2. Если важность признаков не зависит от дисперсии, то стандартизуем каждый столбец  $X$ : так как среднее значение каждого столбца уже 0, достаточно разделить каждый столбец на его стандартное отклонение. Назовем полученную матрицу  $Z$ .
3. Оцениваем ковариационную матрицу вектора признаков как случайных величин с точностью до константы:  $K := X^T X$ .
4. Диагонализуем матрицу  $K$ :  $K = U \Lambda U^T$  (причем, пусть собственные значения в матрице  $\Lambda$  отсортированы по убыванию. Если не так, то сортируем их, не забывая про соответствующие столбцы  $U$ ).
5. Вводим новую матрицу  $Z' = ZU$ . Так как столбцы матрицы  $U$  составляют ортонормированный базис,  $i$ -ая строка матрицы  $Z'$  является проекцией  $i$ -ой строки матрицы  $Z$  на подпространство, натянутое на столбцы матрицы  $U$ . Заметим, что матрица  $Z'$  такого же размера, как и  $X$ , то есть признаков меньше не стало.

## РСА: формулировка алгоритма, иллюстрация $Z'$





## РСА: формулировка алгоритма, часть 2

6. Есть два варианта как нам выбрать новые признаки из матрицы  $Z'$ :

- (a) Можно просто взять первые  $p$  признаков. Так можно сделать, например для визуализации, но, вообще говоря, непонятно как оптимально выбрать  $p$  для общего случая.
- (b) Можно посчитать proportion of variance для каждого признака из  $Z'$  и выбрать столько признаков, что в сумме они объясняют  $x\%$  variance.

Как это посчитать? Мы можем рассмотреть собственные значения на диагонали  $\Lambda$  как "веса" и сказать, что  $i$ -ый признак объясняет

$$\frac{\lambda_i}{\lambda_1 + \dots + \lambda_n}$$

долю variance.





## РСА: связь с SVD

Вместо того, чтобы диагонализировать матрицу  $K$ , мы можем воспользоваться SVD. Пусть  $X = U\Sigma V^T$ . Тогда, в пункте 3

$$K := V\Sigma^T U^T U \Sigma V^T = V(\Sigma^T \Sigma) V^T$$

, что уже будет являться диагонализацией.

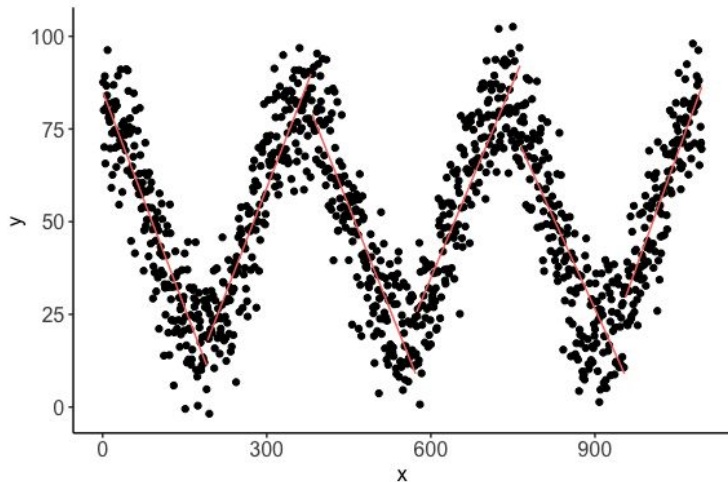
Далее, в пункте 5

$$Z' = U\Sigma V^T \cdot V = U\Sigma$$

Значит, при использовании SVD пункты 3 и 4 можно опустить и в пункте 5 просто ввести  $Z' := U\Sigma$ .

Далее, в 6 пункте мы можем взять соответствующие собственные значения из диагонали матрицы  $\Sigma^T \Sigma$ .

## РСА: проблема с нелинейно зависящими друг от друга признаками



Пример выборки, для которой РСА, скорее всего, сработает не очень хорошо



## Разложение Холецкого

- ❑ Сводит симметричную положительно определенную матрицу к нижнетреугольной матрице, которая при умножении на ее транспонирование создает исходную симметричную матрицу.
- ❑ Первоначально использовалось исключительно для плотных симметричных положительно определенных матриц.
- ❑ Для повышения производительности вычислений часто применяется блочная версия разложения.
- ❑ Для разреженных матриц широко применяется в качестве основного этапа прямого метода решения линейных систем.
- ❑ Варианты разложения Холецкого нашли применения и в итерационных методах для построения переобусловливателей разреженных симметричных положительно определенных матриц.

## Определение

Пусть  $A$  – симметрична и положительно определена. Разложение Холецкого –  $A = LL^*$ , где  $L$  – нижнетреугольная матрица. Допустимо представление в виде  $A = U^*U$ , где  $U$  – верхнетреугольная

$$L = \begin{bmatrix} l_{11} & 0 & 0 & \cdots & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & \cdots & 0 \\ l_{31} & l_{32} & l_{33} & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ l_{n-11} & \cdots & \cdots & l_{n-1\ n-2} & l_{n-1\ n-1} & 0 \\ l_{n1} & \cdots & \cdots & l_{n\ n-2} & l_{n\ n-1} & l_{nn} \end{bmatrix} \quad U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1\ n-1} & u_{1\ n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2\ n-1} & u_{2\ n} \\ 0 & 0 & u_{33} & \cdots & u_{3\ n-1} & u_{3\ n} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & u_{n-1\ n-1} & u_{n-1\ n} \\ 0 & \cdots & \cdots & 0 & 0 & u_{nn} \end{bmatrix}$$



# Алгоритм

Выпишем разложение в общем виде для случая  $n = 3$  и выделим столбцы в матрице  $A$ .

$$\begin{bmatrix} \underline{a_{11}} & a_{12} & a_{13} \\ a_{21} & \underline{a_{22}} & a_{23} \\ a_{31} & a_{32} & \underline{a_{33}} \end{bmatrix} = \begin{bmatrix} l_{11} & & \\ l_{21} & l_{22} & \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ & l_{22} & l_{32} \\ & & l_{33} \end{bmatrix}$$



# Алгоритм

Посчитаем первый столбец матрицы L.

$$\begin{bmatrix} \underline{a_{11}} & a_{12} & a_{13} \\ a_{21} & \underline{a_{22}} & a_{23} \\ a_{31} & a_{32} & \underline{a_{33}} \end{bmatrix} = \begin{bmatrix} l_{11} & & \\ l_{21} & l_{22} & \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ & l_{22} & l_{32} \\ & & l_{33} \end{bmatrix}$$

$$a_{11} = l_{11}^2 \implies l_{11} = \sqrt{a_{11}}$$

$$a_{21} = l_{21}l_{11} \implies l_{21} = \frac{a_{21}}{l_{11}}$$

$$a_{31} = l_{31}l_{11} \implies l_{31} = \frac{a_{31}}{l_{11}}$$



# Алгоритм

Посчитаем второй и третий столбцы матрицы L.

$$\begin{bmatrix} \underline{a_{11}} & a_{12} & a_{13} \\ a_{21} & \underline{a_{22}} & a_{23} \\ a_{31} & \underline{a_{32}} & \underline{a_{33}} \end{bmatrix} = \begin{bmatrix} l_{11} & & \\ l_{21} & l_{22} & \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ & l_{22} & l_{32} \\ & & l_{33} \end{bmatrix}$$

$$a_{22} = l_{21}^2 + l_{22}^2 \implies l_{22} = \sqrt{a_{22} - l_{21}^2}$$

$$a_{32} = l_{31}l_{21} + l_{32}l_{22} \implies l_{32} = \frac{a_{32} - l_{31}l_{21}}{l_{22}}$$

$$a_{33} = l_{31}^2 + l_{32}^2 + l_{33}^2 \implies l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2}$$



## Алгоритм

$$l_{11} = \sqrt{a_{11}},$$

$$l_{j1} = \frac{a_{j1}}{l_{11}}, \quad j \in [2, n],$$

$$l_{ii} = \sqrt{a_{ii} - \sum_{p=1}^{i-1} l_{ip}^2}, \quad i \in [2, n],$$

$$l_{ji} = \frac{1}{l_{ii}} \left( a_{ji} - \sum_{p=1}^{i-1} l_{ip} l_{jp} \right), \quad i \in [2, n-1], j \in [i+1, n].$$





## Преимущества алгоритма

- ❑ Вдвое меньше тратим как объем памяти, так и количество операций в сравнении с, например, разложением по методу Гаусса.
- ❑ Благодаря тому, что разлагаемая матрица не только симметрична, но и положительно определена, её разложение методом Холецкого имеет наименьшее эквивалентное возмущение из всех известных разложений матриц.



## Применение в линейной регрессии

Благодаря разложению Холецкого и его тесной связи с LU-разложениями, мы можем оптимизировать вычисление систем уравнений. Такое свойство особенно актуально в линейной регрессии, где задача формулируется так:

$Y = X * \beta$ , где  $Y$  – ответы (зависимые переменные),  $X$  – признаки (независимые переменные),  $\beta$  – вектор весов признаков.

Как мы можем найти  $\beta$ ? Оказывается, с помощью разложения Холецкого можно достаточно быстро решить уравнения (почти в 2-2.5 раза быстрее, чем с помощью других методов). Это связано с тем, что система линейных уравнений порождает симметричную матрицу ковариаций.



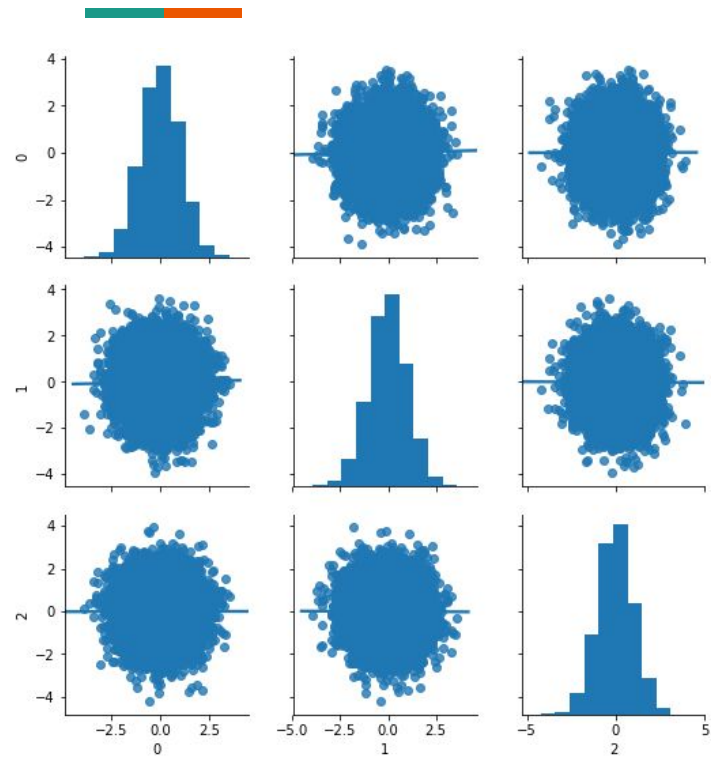
# Генерация случайных величин

Пусть у нас есть вектор  $X$  нормальных некоррелированных случайных величин и их матрица ковариации  $\Sigma$ . Хотим получить нормальные коррелированные случайные величины с той же матрицей  $\Sigma$ .

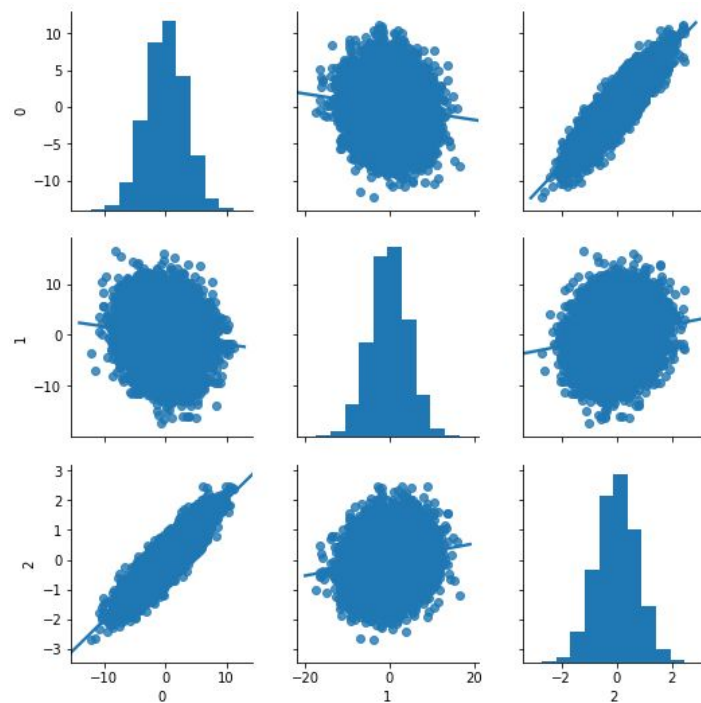
Как это сделать?

*Используем метод моделирования Монте-Карло с помощью разложения Холецкого.*

Посчитаем разложение от ковариационной матрицы и получим новые случайные величины как  $Y = \Sigma X$ .



$\mathcal{N}(0, 1)$



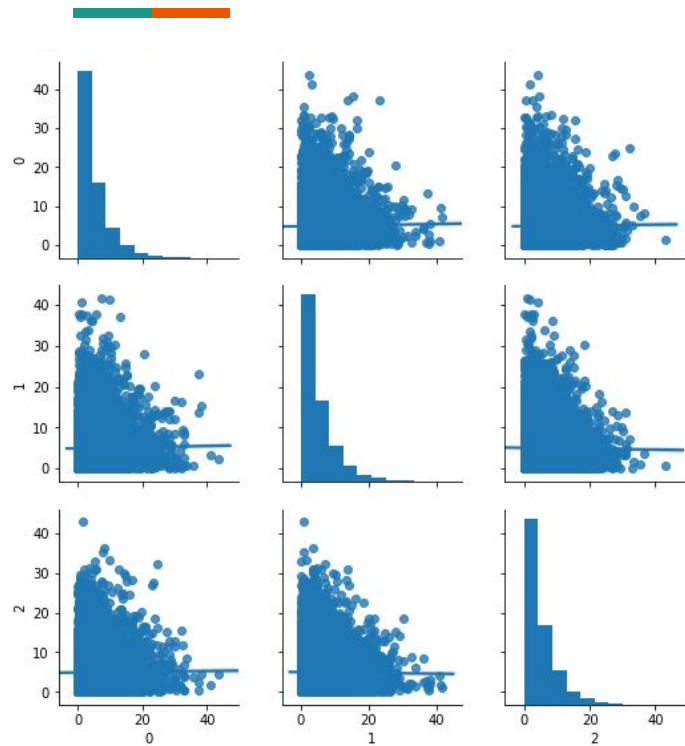


## Что получилось?

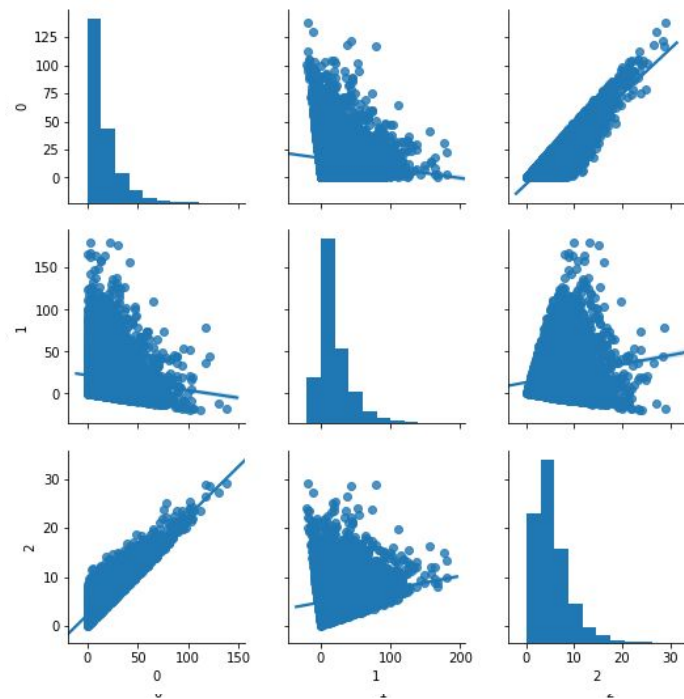
1. Случайные величины на картинке  $[0, 2]$  имеют сильно положительную корреляцию.
2.  $[0, 1]$  слегка отрицательную.
3.  $[1, 2]$  слегка положительную.
4. Стандартное отклонение переменной 2 не особо изменилось, в то время как для 0 и 1 - существенно.



Используем другое распределение?



$\Gamma(1, 5)$





## Что получилось?

На самом деле, подобный метод не работает не с нормальными случайными величинами, именно поэтому у нас ничего не получилось - полученные величины уже не имеют гамма-распределение (переменная 1 принимает отрицательные значения, в то время как гамма-распределение строго положительное).



# Использование

Методы Монте-Карло используются для моделирования *случайных процессов* для решения задач и исследований в области физики (прямое моделирование элементарных частей системы), бизнес-информатики (моделирование бизнес-процессов), химии, математике, экономике и тд.

Примеры случайных процессов (последовательностей):

$\{X_n\}_{n \in \mathbb{N}}$ , где  $X_j \sim N(0, 1)$  - случайная последовательность.

$f : \mathbb{R} \rightarrow \mathbb{R}$ ,  $Y$  - случайная величина, тогда

$X_t(\omega) = f(t)Y(\omega)$  - случайный процесс.

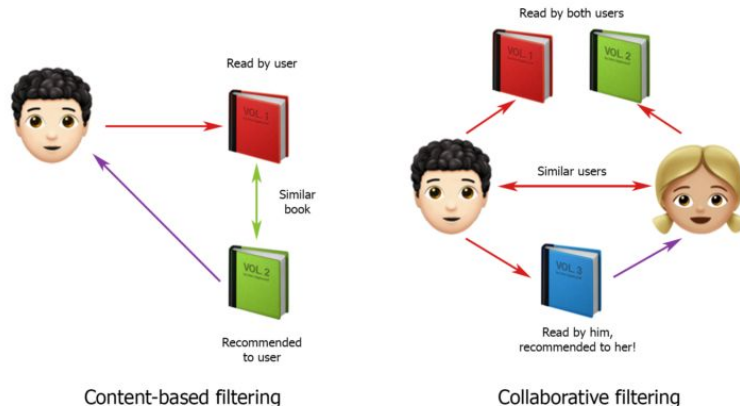


# Рекомендательные системы

Рекомендательные системы - это модели, которые по имеющейся информации о пользователе (сфере его интересов, предпочтений и т.д.) могут предсказать, какие объекты будут интересны этому пользователю.

Типы рекомендательных систем:

- ❑ *Фильтрация на основе содержания* (по характеристикам объекта и пользователя).
- ❑ *Коллаборативная фильтрация* использует известные предпочтения группы пользователей для прогнозирования неизвестных предпочтений другого пользователя.



# Коллаборативные системы

- ❑ Основная идея: те, кто одинаково оценивал какие-либо объекты в прошлом, склонны давать похожие оценки другим объектам и в будущем.
- ❑ Прогнозы составляются индивидуально для каждого пользователя.
- ❑ При КФ используется информация о поведении пользователей в прошлом. Не имеет значения, с какими типами объектов ведется работа, но при этом могут учитываться неявные характеристики, которые сложно было бы учесть при создании профиля.





## Подход “по соседству”

Этот тип появился первым и сегодня используется в большинстве рекомендательных систем. Для пользователя подбирается подгруппа пользователей со схожими интересами, и на основе комбинаций весов и оценок подбирается контент, который с большей долей вероятности заинтересует человека.

Алгоритм прост:

- Присвоить вес каждому пользователю с учётом схожести его оценок и активного пользователя.
- Выбрать несколько пользователей (соседей), которые имеют максимальный вес (максимально похожи на активного пользователя).
- Вычислить предсказание оценок активного пользователя для неоцененных им объектов с учетом весов и оценок соседей.



## Модельный подход

Данный подход предоставляет рекомендации, измеряя параметры статистических моделей для оценок пользователей, построенных с помощью различных математических методов и методов интеллектуального анализа данных и алгоритмов машинного обучения.

Данный метод является более комплексным и обеспечивает лучшую точность, позволяя выявлять скрытые закономерности, лучше масштабирует большие данные, но при этом является более «трудноподъемным». Приходится искать компромиссы между точностью и стоимостью реализации.



## Гибридный подход

Распространен больше остальных, особенно если рекомендательная система разрабатывается для коммерческого сайта: интернет-магазина, маркетплейса и т.п. Он объединяет в себе два первых типа и помогает преодолеть ограничения изначального оригинального подхода (основанного на соседстве) и улучшить точность рекомендаций.

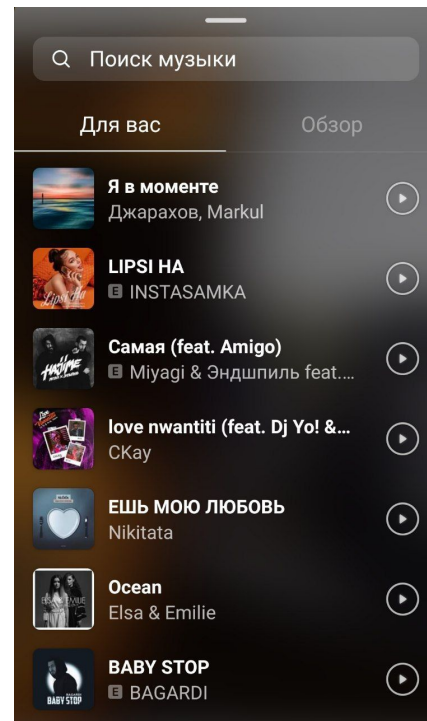
Он решает и другие трудности, например, проблему разреженности данных и потери информации. Из-за этого он сложен и дорог в реализации и применении, но при этом приносит компаниям много пользы.

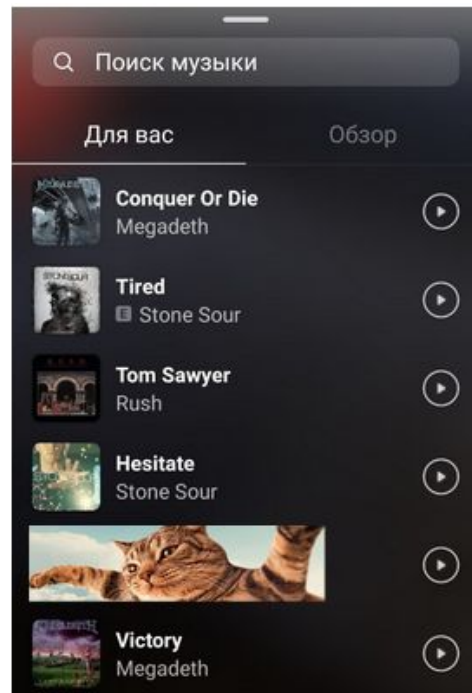
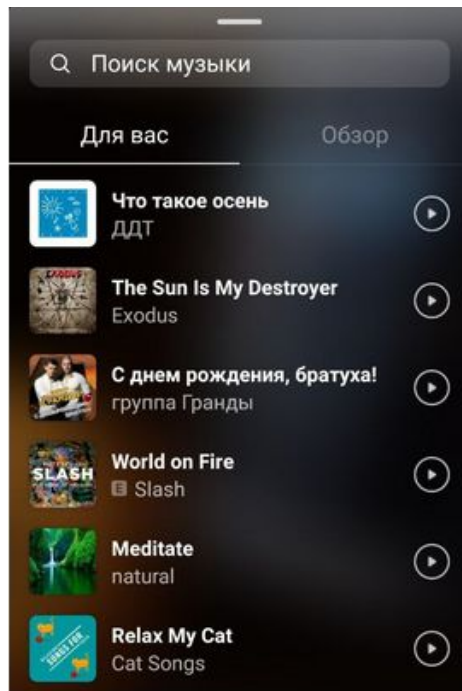
# Проблема “холодного старта”

“Холодный старт” - это главная проблема коллаборативных моделей, которая заключается в том, что неизвестно, что рекомендовать новому пользователю или как продвигать новый товар. Также проблема холодного старта встречается с объектами, с которыми редко осуществляется взаимодействие.

Часто в таких ситуациях рейтинги искусственно корректируют.

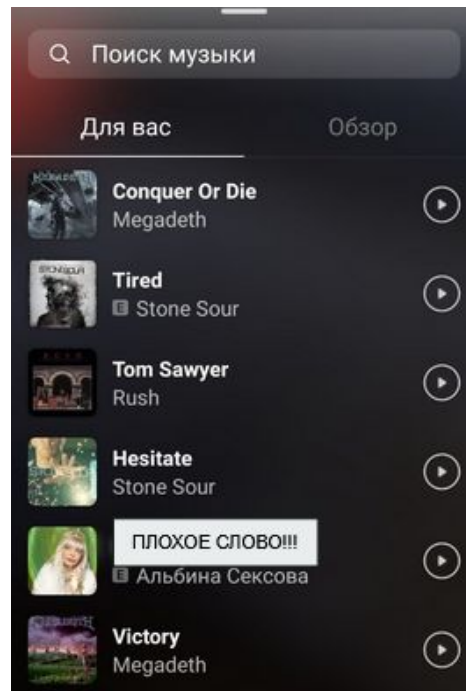
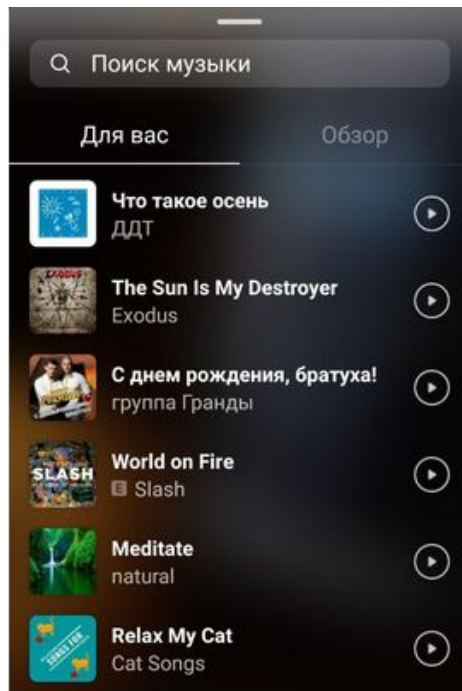
- ❑ Сглаженное среднее.
- ❑ Средний рейтинг и интервал достоверности.





“Не очень холодный старт”

Что скрыто за котиком?



“Не очень холодный старт”

За котиком скрыто нечто  
ужасающее...







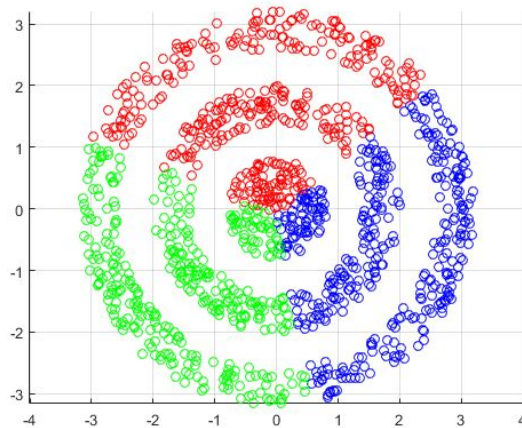
## Другие проблемы КС

- Разреженность данных.
- Масштабируемость.
- Синонимия
- Нечестные конкуренты и накрутка дизлайков.
- Разнообразие
- Белые вороны (поиски решения данной проблемы в настоящее время не ведутся).

# Кластеризация

Кластерный анализ (Data clustering) — задача разбиения заданной выборки объектов (ситуаций) на непересекающиеся подмножества, называемые кластерами, так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались.

В рекомендательных системах кластеризация используется для разбиения числа пользователей на группы по интересам. Нам лишь нужно знать векторы (пользователь), координатами которых являются признаки (интересы).





## K-means

Является наиболее популярным алгоритмом кластеризации благодаря своей простоте и эффективности. Пусть  $k$  – количество кластеров,  $S_i$  – кластер и  $\mu_i$  – центр масс всех векторов (точек) из этого кластера (центроид). Формально, каждый центроид — это вектор, элементы которого представляют собой средние значения соответствующих признаков, вычисленные по всем записям кластера. Тогда задача сводится к минимизации такого функционала:

$$V = \sum_{i=1}^k \sum_{x \in S_i} (x - \mu_i)^2$$

Существует версия алгоритма, где вместо среднего выступает медиана – метод  $k$  медиан.

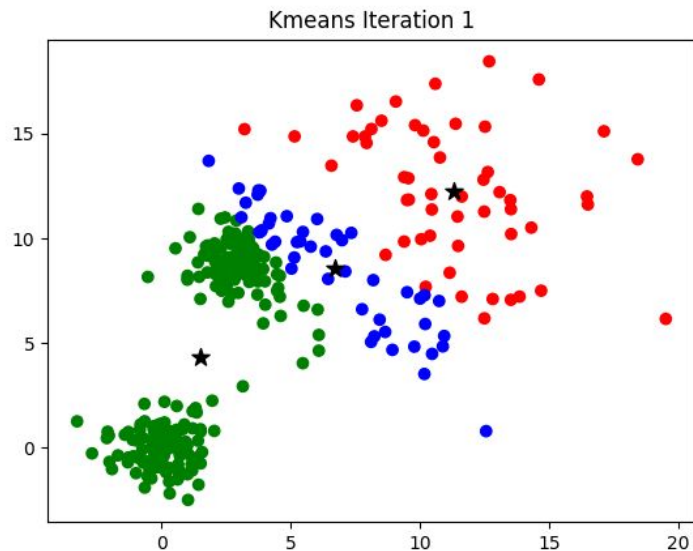


# Алгоритм

1. Выбирается число кластеров  $k$ .
2.  $k$  центроидов случайным образом раскидываются по пространству.
3. Для каждого вектора подсчитывается, к какому кластеру он ближе - формируются “кластерные выборки”.
4. Каждый центроид перемещается в центр такой выборки, которую мы отнесли к этому центроиду.
5. Повторяем фиксированное число раз, либо до сходимости кластеров - пока внутрекластерное расстояние не перестанет уменьшаться.

На каждой итерации происходит изменение границ кластеров и смещение их центров. В результате минимизируется расстояние между элементами внутри кластеров и увеличиваются междукластерные расстояния.

# Наглядный принцип работы

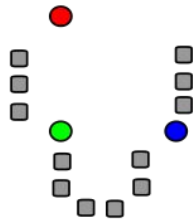


В качестве расстояния обычно используют евклидово расстояние. Формально, распределение векторов выглядит так:

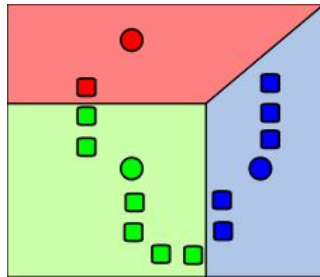
$$S_j^{(t)} = \{x : \|x - \mu_j^{(t)}\|^2 \leq \|x - \mu_i^{(t)}\|^2 \forall i = 1, \dots, k\},$$
где вектор  $x$  относится к одному единственному кластеру,  
 $t$  - номер итерации,  $\mu_i^{(t)}$  -  $i$ -ый центр кластера для итерации  $t$

$$\mu_j^{(t+1)} = \frac{1}{|S_j^{(t)}|} \sum_{x \in S_j^{(t)}} x$$

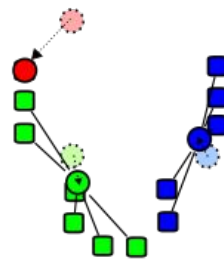
## Еще картиночка



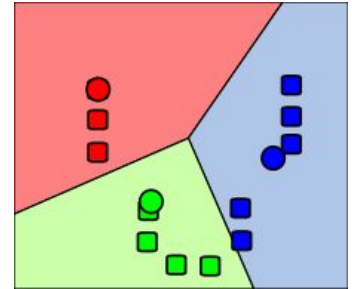
Инициализация



Определение  
кластерных  
выборок



Переход к новым  
центрам  
кластеров



Определение  
кластерных  
выборок

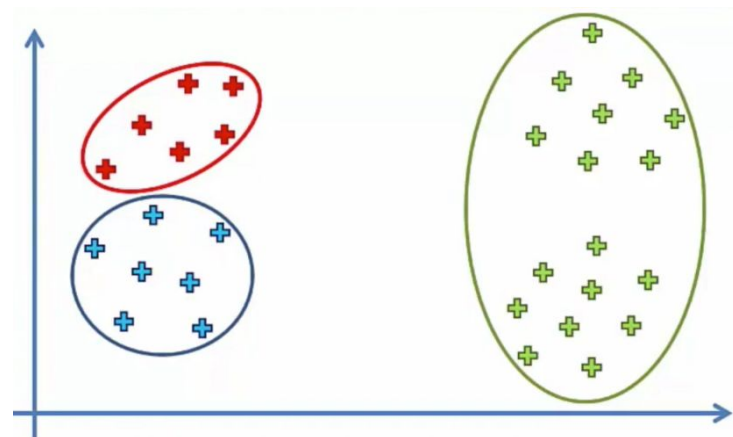
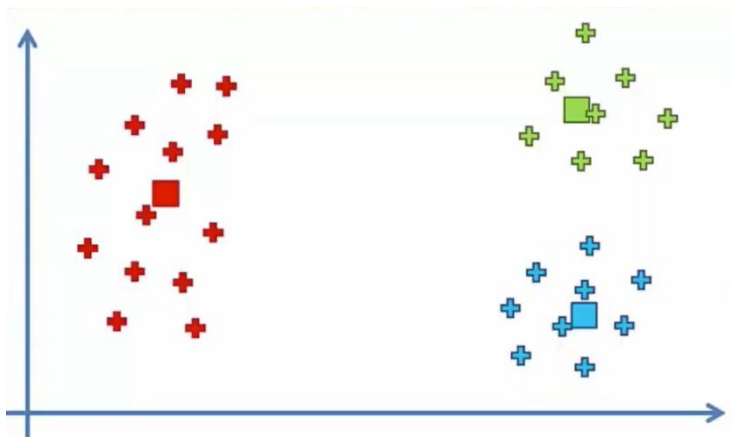


# Проблемы

Алгоритм, несмотря на свою простоту и эффективность имеет несколько проблем:

- Не гарантируется достижение глобального минимума суммарного квадратичного отклонения  $V$ , а только одного из локальных минимумов.
- Результат зависит от выбора исходных центров кластеров, их оптимальный выбор неизвестен.
- Число кластеров надо знать заранее.

## Неоднозначность результата



Решение: алгоритм *k-means++*



# SVD в рекомендательных системах

Дана матрица вида “товар - пользователь” размера. Как предсказать, что будет вместо нулей?

	Миша	Маша	Рома	Дима	Витя	Вова
Овощи	0	1	0	1	2	2
Фрукты	2	3	1	1	2	2
Сладости	1	1	1	0	1	1
Хлеб	0	2	3	4	1	1
Кофе	0	0	0	0	1	0

Подсчитаем SVD-разложение от матрицы и оставим  $f$  сингулярных значений.

$$X = UDV^T = U \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_k \end{bmatrix} V^T \approx U \begin{bmatrix} \sigma_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & \sigma_f & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix} V^T$$

# SVD в рекомендательных системах

SVD-разложение максимизирует правдоподобие имеющихся рейтингов (минимизирует ошибку предсказания рейтингов) и разбивает данные на независимые главные компоненты (скрытые факторы).

Используя PCA-подход, может разложить любую матрицу на две, где первая *содержит скрытый фактор пользователя*, а вторая - *скрытый фактор объекта*.

$$A = U S V^T = \underbrace{\sigma_1}_{\text{более значимый}} \underbrace{u_1}_{m \times 1} \underbrace{v_1^T}_{1 \times n} + \dots + \underbrace{\sigma_r}_{\text{менее значимый}} \underbrace{u_r}_{m \times 1} \underbrace{v_r^T}_{1 \times n}$$

расположите  $\sigma_i$  в порядке убывания

  
$$\begin{matrix} & R & & P & & Q^T \\ \text{пользователь } 1 & \begin{pmatrix} r_{11} & & r_{1n} \\ & \ddots & \\ r_{m1} & & r_{mn} \end{pmatrix} & \approx & \begin{pmatrix} p_{11} & & p_{1k} \\ & \ddots & \\ p_{m1} & & p_{mk} \end{pmatrix} & \begin{pmatrix} q_{11} & & q_{n1} \\ & \ddots & \\ q_{1k} & & q_{nk} \end{pmatrix} \\ & m \times n & & m \times k & & k \times n \\ & \text{предмет } 1 & & \text{скрытый фактор пользователя } m & & \text{скрытый фактор предмета } 1 \end{matrix}$$



## В чем же смысл?

В случае рекомендательных систем получается, что мы представляем каждого пользователя вектором из  $f$  факторов  $v_j$  и каждый продукт вектором из  $f$  факторов  $u_i$ , а потом, чтобы предсказать рейтинг пользователя  $i$  товару  $j$ , берём их скалярное произведение. Также можно использовать и SGD.

Можно сказать, что вектор факторов пользователя показывает, насколько пользователю нравится или не нравится тот или иной фактор, а вектор факторов продукта показывает, насколько тот или иной фактор в продукте выражен. Такое разложение часто возможно имеет содержательный смысл.



## LU-разложение

$A = LU$ , где  $L$  – нижнеунитреугольная,  
 $U$  – верхнетреугольная матрицы

$$L = \begin{pmatrix} 1 & 0 & \dots & 0 \\ * & 1 & \dots & \dots \\ \dots & \dots & 1 & 0 \\ * & \dots & * & 1 \end{pmatrix} \quad U = \begin{pmatrix} * & * & \dots & * \\ 0 & * & \dots & \dots \\ \dots & \dots & * & * \\ 0 & \dots & 0 & * \end{pmatrix}$$



## Решение СЛАУ

LU-разложение – прямой метод решения СЛАУ, который используется на практике

$$Ax = b, A = LU \rightarrow LUx = b$$

Обозначим  $Ux = y \rightarrow \begin{cases} Ly = b - \text{прямая подстановка} \\ Ux = y - \text{обратная подстановка} \end{cases}$



## Связь LU-разложения и метода Гаусса

Рассмотрим квадратную матрицу:

$$\begin{pmatrix} a & c^T \\ b & D \end{pmatrix}$$

Хотим занулить все элементы в первом столбце ниже первого



## Связь LU-разложения и метода Гаусса

Для этого домножаем слева на матрицу:

$$\begin{pmatrix} 1 & 0 \\ -\frac{1}{a}b & I \end{pmatrix} \begin{pmatrix} a & c^T \\ b & D \end{pmatrix} = \begin{pmatrix} a & c^T \\ 0 & D - \frac{1}{a}bc^T \end{pmatrix}$$

$\parallel$   
 $Z_1$



## Связь LU-разложения и метода Гаусса

Проделав аналогичные преобразования для  $(n - 1)$  столбцов получим верхнетреугольную  $U$

$$Z_{n-1} * \dots * Z_2 * Z_1 * A = U$$

$$A = (Z_1^{-1} * Z_2^{-1} * \dots * Z_{n-1}^{-1}) * U$$

$$\begin{matrix} || \\ L \end{matrix}$$





## Определитель матрицы и сложность

- 1) Зная LU-разложение можно очень просто вычислить определитель матрицы:

$$|A| = |L| * |U| = 1 * |U| = \prod_{i=1}^n u_{i,i}$$

- 2) Сложность алгоритма:
- $$\frac{2}{3}n^3 + O(n^2)$$



# Тематическое моделирование

Тематическая модель – модель коллекции текстовых документов, которая определяет, к каким темам относится каждый документ коллекции.

Применение:

- Выявление трендов в новостных потоках
- Анализ текстовых данных социальных сетей
- Классификация и категоризация документов
- Для различных целей в биоинформатике и физике, например, анализе нуклеотидных последовательностей



## Наиболее известные подходы

- Латентно-семантический анализ (LSA)
- Латентное размещение Дирихле (LDA)
- Вероятностный латентно-семантический анализ (PLSA)



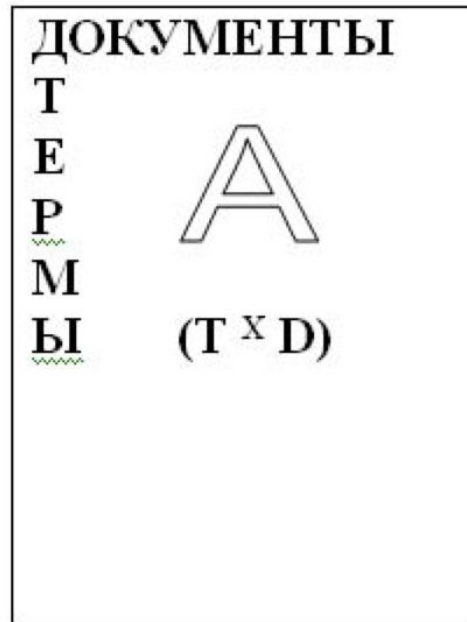
## Латентно-семантический анализ (LSA)

- Используется для выявления скрытых семантических связей между терминами (т.е. словами или n-граммами)
- **Основная идея:**
  - Выполнение некоторого алгебраического преобразования векторного пространства термины-на-документы
  - Поиск зависимостей между векторами в получившемся пространстве

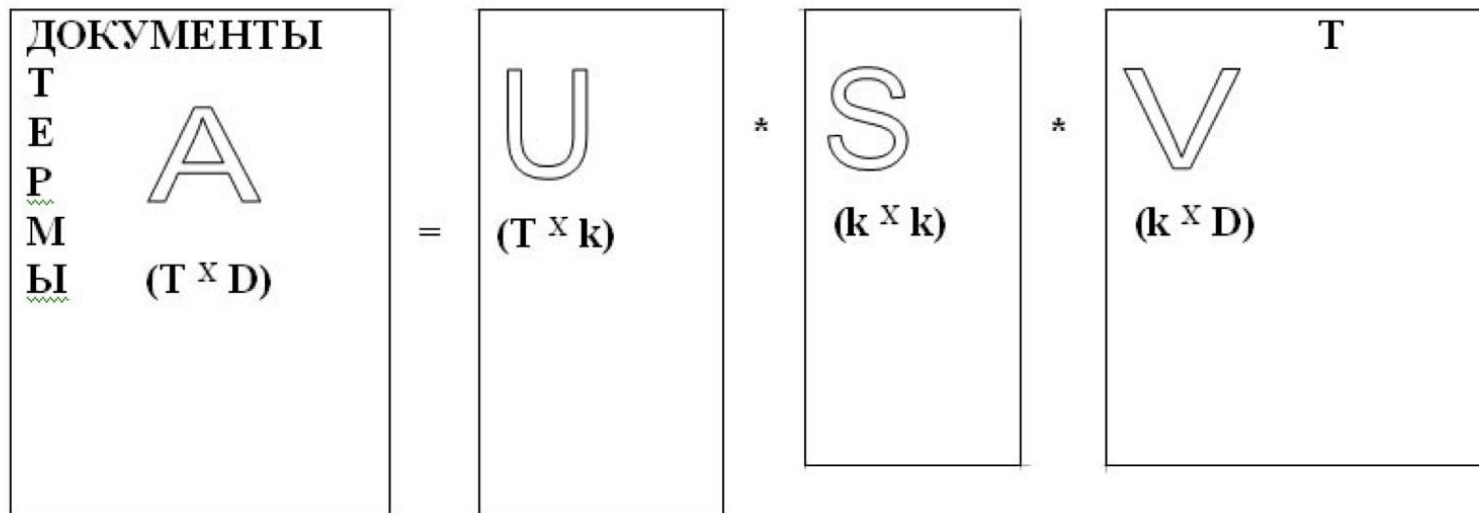
# Латентно-семантический анализ (LSA)

## Некоторые понятия:

- $D$  — множество (коллекция) текстовых документов
- $W$  — множество (словарь) всех употребляемых в них терминов
- Каждый документ  $d \in D$  представляет собой последовательность  $n_d$  терминов  $w_1, \dots, w_n$  из словаря  $W$



## Латентно-семантический анализ (LSA)





## Плюсы и минусы LSA

### Плюсы:

- метод является наилучшим для выявления латентных зависимостей внутри множества документов
- Частично снимается омонимия

### Минусы:

- снижение скорости вычисления при увеличении объёма входных данных (из-за сложности SVD)