

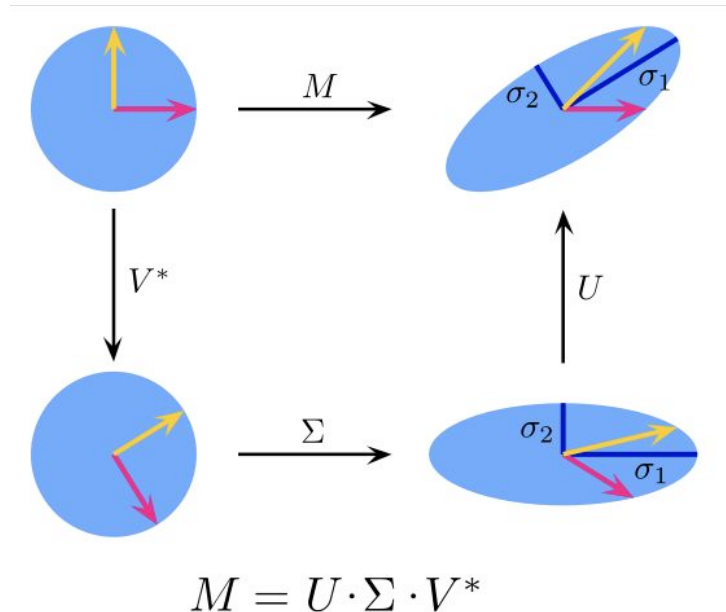
# Матричные разложения и их применения

Павлов Вадим, Молодык Петр

# Сингулярное разложение

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

- где  $\Sigma$  — матрица размера  $m \times n$  с неотрицательными элементами с нулями вне главной диагонали, элементы на главной диагонали — это сингулярные числа
- $U$  и  $V$  — унитарные матрицы, соответствующие ортонормированным базисам



*Геометрический смысл сингулярного разложения*

# Базовые применения

---

- Вычисление псевдообратной матрицы:
  - Решения линейных систем
  - Метод наименьших квадратов

$$\mathbf{M}^\dagger = \mathbf{V} \mathbf{\Sigma}^\dagger \mathbf{U}^*$$

- Приближение матрицей меньшего ранга:
  - Для сжатия данных

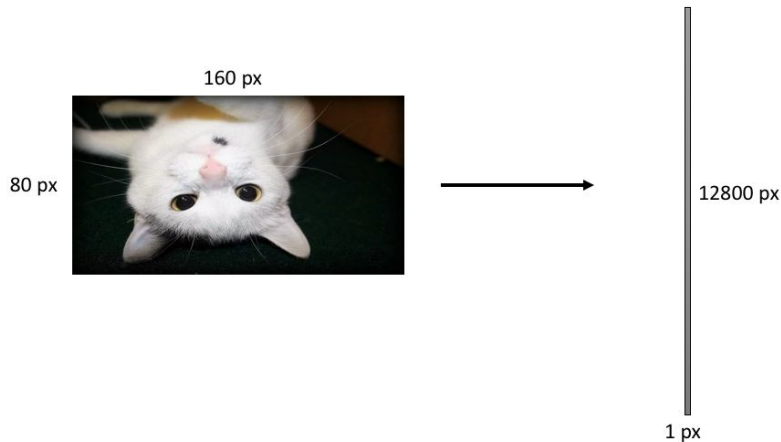
$$M_k = U \Sigma_k V^*$$

# Выделение фона на видео

**Шаг 1.** Разделить видео по кадрам

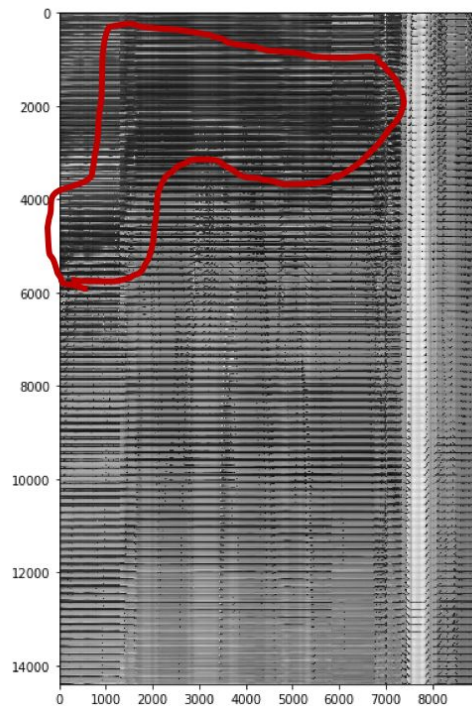


**Шаг 2.** Каждый кадр преобразовать в одномерный вектор

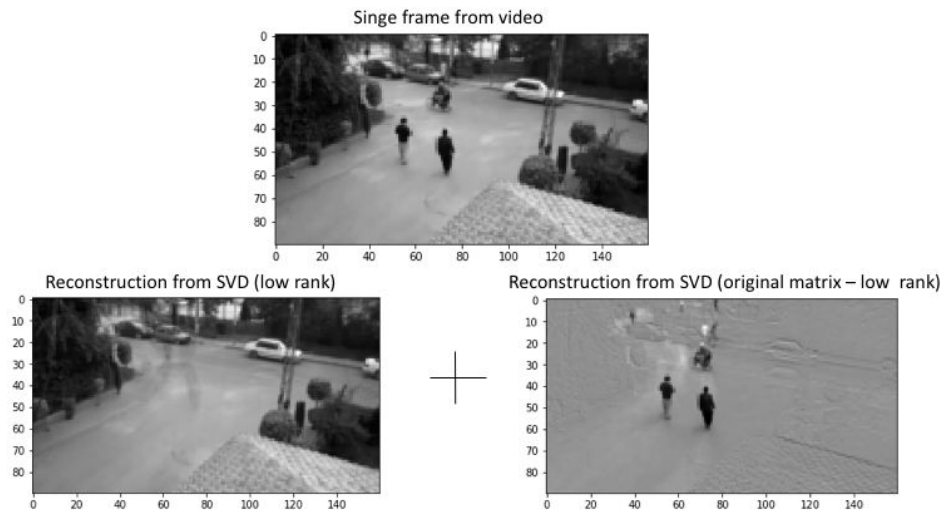


# Выделение фона на видео

**Шаг 3.** Совместить эти вектора для каждого кадра



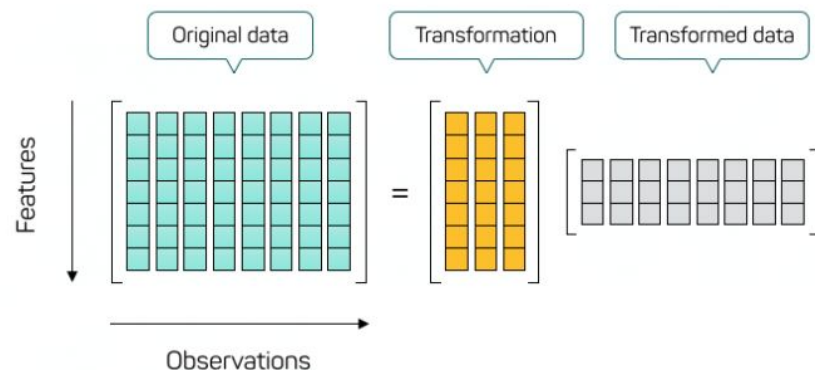
**Шаг 4.** Применить сингулярное разложение к получившейся матрице и низкоранговое приближение окажется фоном



# Выделение общих черт у лиц

**Шаг 1.** Берем набор изображений человеческих лиц

**Шаг 2.** Каждый кадр преобразовать в одномерный вектор и кладем один за другим



**Шаг 3.** К полученной матрице применить сингулярное разложение и взять низкоранговое приближение  $M = HW$

**Шаг 4.** Столбцы матрицы  $H$  образуют набор векторов, через которые хорошо выражаются все изображения из набора

# Выделение общих черт у лиц

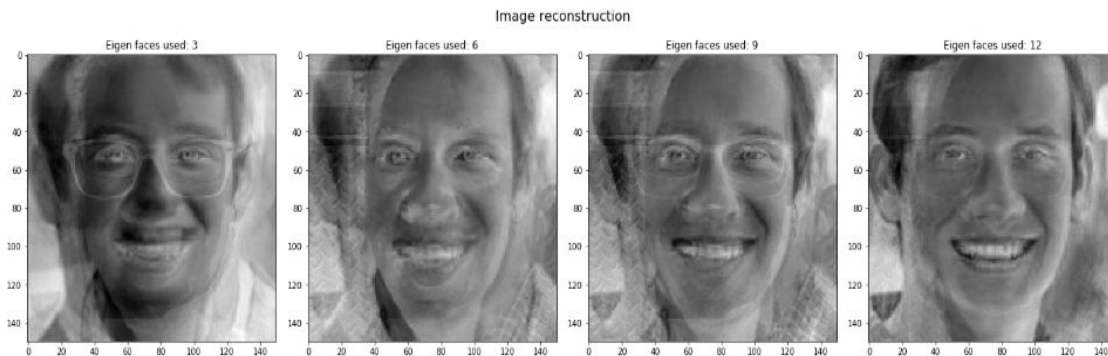
В результате предыдущего шага получаются так называемые “собственные лица” (по аналогии с собственными векторами)



Каждое лицо можно представить в виде их суммы с некоторыми коэффициентами



Использовать эти коэффициенты для сравнения и распознавания лиц

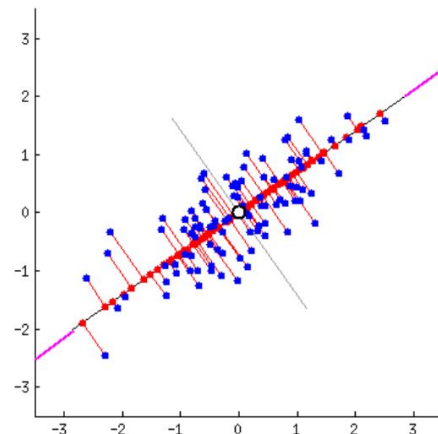
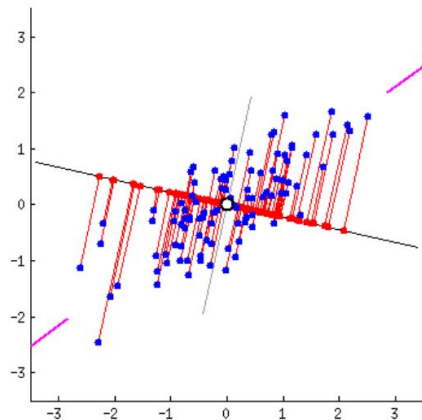


*Реконструкция лица по первым  $k$  собственным лицам  
(как же выбрать итоговое количество?)*

# Метод главных компонент

- В машинном обучении часто возникает задача уменьшения размерности признакового пространства.
- Естественно искать новые признаки среди линейных комбинаций исходных признаков.
- Что хотим от новых признаков?
  - Во-первых, чтобы они сильно различались между объектами.
  - Во-вторых, чтобы эти признаки как можно лучше описывали исходные данные

Из картинок несложно заметить, что хотим мы одно и то же:





# Метод главных компонент

---

Формальная постановка задачи:

- $X \in \mathbb{R}^{\ell \times D}$  - матрица “объект-признак”
- Ищем  $u_1, \dots, u_D \in \mathbb{R}^D$  - главные компоненты, которые удовлетворяют свойствам:
  1. Они **ортогональны**
  2. Они **нормированы**
  3. При проекции выборки на первые  $k$  главных компонент получается **максимальная дисперсия** по всем возможным способам выбрать эти  $k$  компонент

Решая эту задачу, оказывается, что оптимальный выбор  $u_1$  - собственный вектор соответствующий максимальному собственному значению  $X^T X$  и так далее.

Ровно такой же ответ мы бы получили, если бы применили сингулярное разложение к исходной матрице!

# Метод главных компонент

Получается, нам нужно уметь вычислять собственные вектора, соответствующие наибольшему собственному значению матрицы  $\mathbf{X}^T\mathbf{X}$ .

Вычисление самой матрицы затратно, но можно ее не вычислять.

Идея: свести задачу к вычислению  $\mathbf{X}^T(\mathbf{X} \mathbf{r})$

Если наибольшее собственное значение значительно больше последующих, то можно использовать приближенный алгоритм, итеративно вычисляя  $\mathbf{X}^T(\mathbf{X} \mathbf{r})$ , пока не получим удовлетворительный результат.

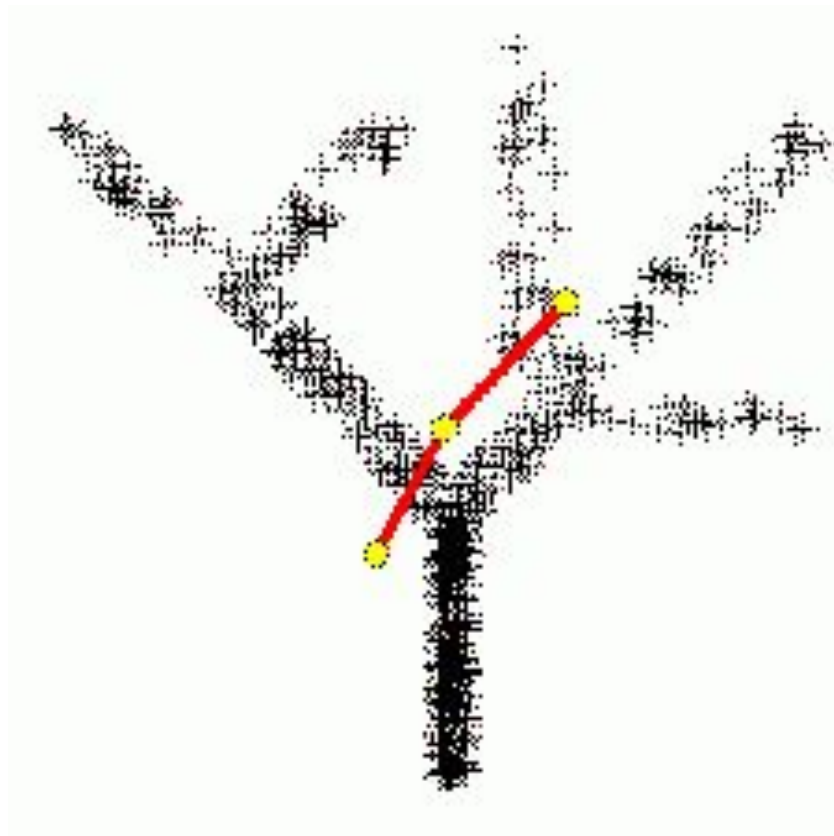
```
 $\mathbf{r}$  = a random vector of length  $p$   
 $\mathbf{r} = \frac{\mathbf{r}}{|\mathbf{r}|}$   
do  $c$  times:  
     $\mathbf{s} = 0$  (a vector of length  $p$ )  
    for each row  $\mathbf{x} \in \mathbf{X}$   
         $\mathbf{s} = \mathbf{s} + (\mathbf{x} \cdot \mathbf{r})\mathbf{x}$   
     $eigenvalue = \mathbf{r}^T \mathbf{s}$   
     $error = |eigenvalue \cdot \mathbf{r} - \mathbf{s}|$   
     $\mathbf{r} = \frac{\mathbf{s}}{|\mathbf{s}|}$   
    exit if  $error < tolerance$   
return  $eigenvalue, \mathbf{r}$ 
```

# Метод главных компонент

---

Понятно, что для данных со сложной структурой достичь хорошего результата используя линейное приближение можно не всегда, поэтому используются более продвинутые методы.

Например, **метод топологических грамматик**.

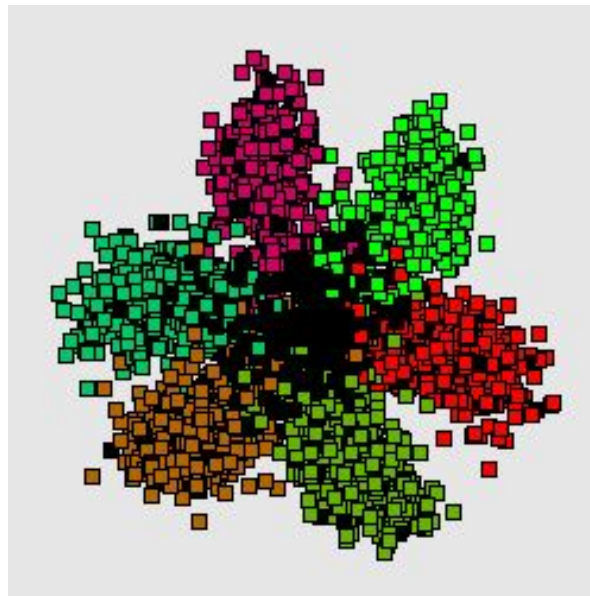


# Метод главных компонент

---

Применения:

- Визуализация данных
- Компрессия изображений и видео
- Индексация видео
- Психодиагностика
- Сенсорная оценка пищевых продуктов



*Проекция ДНК-блуждания на  
первые 3 главные компоненты  
для генома бактерии  
Streptomyces coelicolor.*

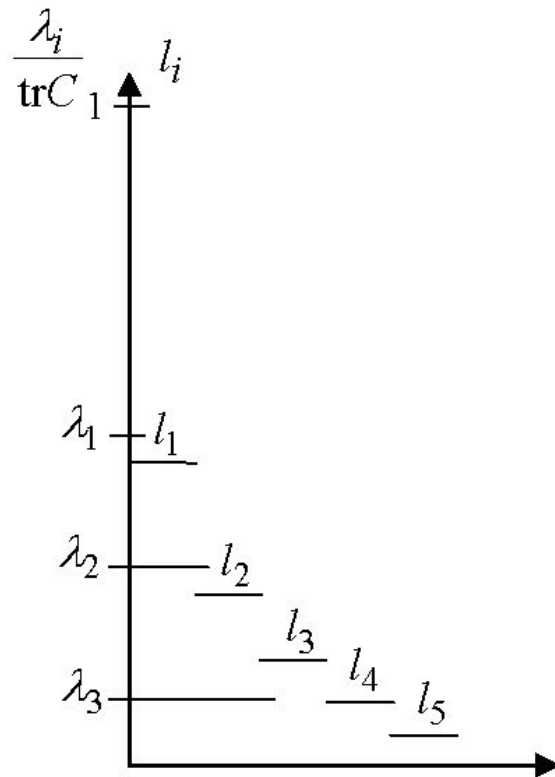
# Правило сломанной трости

Набор нормированных на единичную сумму собственных чисел сравнивается с распределением длин обломков трости единичной длины, сломанной в  $n - 1$  случайно выбранной точке.

$$l_i = E(L_i) = \frac{1}{n} \sum_{j=i}^n \frac{1}{j}.$$

По правилу сломанной трости  $k$ -й собственный вектор сохраняется в списке главных компонент, если

$$\frac{\lambda_1}{\text{tr } C} > l_1 \text{ and } \frac{\lambda_2}{\text{tr } C} > l_2 \text{ and } \dots \frac{\lambda_k}{\text{tr } C} > l_k.$$



# Неотрицательное разложение

Иногда используют неотрицательное матричное разложение, в котором матрица  $V$  с неотрицательными элементами раскладывается на произведение двух матриц  $W$  и  $H$  меньшего ранга, элементы которых тоже неотрицательны.

$$\begin{bmatrix} & \\ & \\ & \\ & \end{bmatrix}^W \times \begin{bmatrix} & & & & & \\ & & & & & \end{bmatrix}^H \approx \begin{bmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix}^V$$

Это разложение, в отличие от сингулярного, не обладает свойством ортогональности, но зато проще в вычислении и используется в ситуациях, когда данным логически свойственна неотрицательность

# Латентно-семантический анализ

---

Латентно-семантический анализ текста (ЛСА) - это процесс выделения в текстах и словах неявных “тематик” для его дальнейшего анализа.

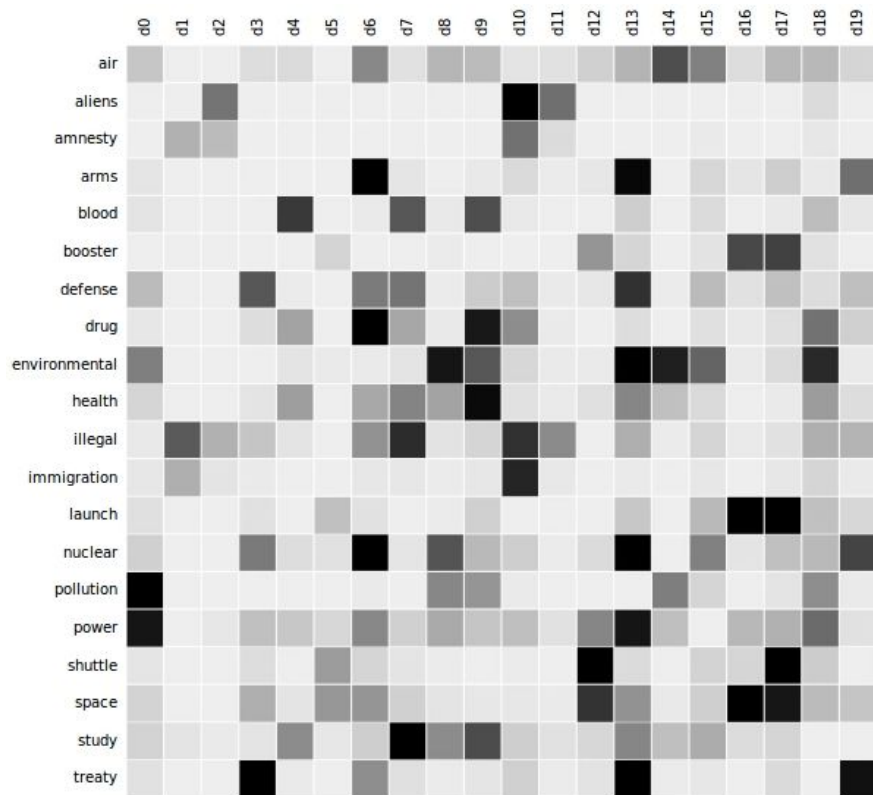
	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
voyage	1	0	0	1	1	0
trip	0	0	0	1	0	1

Основная идея метода основана на том, что в текстах с похожей тематикой часто встречаются одни и те же слова

# Латентно-семантический анализ

Рассматривается терм-документная матрица, в которой столбцы соответствуют документам, а строки - словам, и в ячейке находится число, равное частоте вхождения слова в соответствующий документ

Метод ЛСА состоит в построении низкорангового приближения этой матрицы ранга  $k$ . Для этого можно использовать сингулярное разложение, или NMF (для сохранения логики частотности). Полученные  $k$ -мерные представления текстов и слов и будут их разложением на неявные признаки.






























# Рекомендательные системы

## Дано:

- $U$  - множество пользователей
- $I$  - множество объектов рекомендации
- $r_{ui}$  - оценка  $i$ -того товара  $u$ -тым пользователем

## Надо:

Для каждого пользователя найти  $k$  товаров, которые будут ему наиболее интересны

# Модели, основанные на схожести

---

Два пользователя считаются “похожими”, если у них близкие оценки для одних объектов.

Более формально, будем считать схожесть двух пользователей по корреляции Пирсона.

$$w_{uv} = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}},$$

Аналогично два объекта считаются “похожими”, если для них большое количество пользователей оставили похожие оценки.

$$w_{ij} = \frac{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)^2} \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \bar{r}_j)^2}}$$

# Выбор рекомендации

---

Теперь, когда подсчитаны “похожести” объектов и пользователей, надо понять, как выбирать рекомендации.

## 1. На основе пользователей

- Похожие пользователи:  $U(u_0) = \{v \in U \mid w_{u_0v} > \alpha\}$ .
- Ранжируем объекты по частоте выбора этими пользователями  $p_i = \frac{|\{u \in U(u_0) \mid \exists r_{ui}\}|}{|U(u_0)|}$ .
- Выбираем первые  $k$

## 2. На основе объектов

- Выбираем объекты, похожие на интересы пользователя:  $I(u_0) = \{i \in I \mid \exists r_{u_0i_0}, w_{i_0i} > \alpha\}$ .
- Смотрим, какой объект ближе, по оценившим его людям:  $p_i = \max_{i_0: \exists r_{u_0i_0}} w_{i_0i}$ .
- Выбираем первые  $k$

# Модели с неявными данными

---

Модели оценивания по схожести имеют проблему - необходимо хранить большую матрицу отношений людей и объектов

1. Построим для каждого пользователя и объекта векторные представления  $p_u \in \mathbb{R}^d$  и  $q_i \in \mathbb{R}^d$  фиксированной размерности, которые описывают некоторые скрытые признаки.

2. “Совместимость” пользователя и объекта будем считать как  $r_{ui} \approx \langle p_u, q_i \rangle$ . Тогда, если  $R$  - множество пар пользователь-объект, для которых известна оценка, то функционал ошибки следующий:

$$\sum_{(u,i) \in R} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle)^2 \rightarrow \min_{P, Q}$$

Заметим, что, если положить  $R'$  - центрированной матрицей оценок  $R$  и записать это в матричном виде:  $\|R' - P^T Q\|^2 \rightarrow \min_{P, Q}$

То мы получим знакомую нам задачу низкорангового приближения матриц!

# Обучение модели

---

Решать задачу низкорангового приближения точно тяжело, поэтому можно использовать:

1. Численные приближения матричных разложений
2. Стохастический градиентный спуск: для случайных  $(u, i)$

$$\begin{aligned}p_{uk} &:= p_{uk} + \eta q_{ik} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle), \\q_{ik} &:= q_{ik} + \eta p_{uk} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle).\end{aligned}$$

3. ALS / HALS

# Бонус! Разложение Холецкого

---

При решении оптимизационных задач часто приходится решать системы линейных уравнений, поэтому особого внимания достойны методы, ускоряющие их решение. Один из таких методов - разложение Холецкого.

Метод состоит в разложении симметричной положительно-определенной матрицы следующим образом:  $A = LL^T$ , где  $L$  - нижнетреугольная матрица с положительными элементами на диагонали. Матрицу  $L$  легко вычислить последовательно:

$$\begin{aligned}l_{11} &= \sqrt{a_{11}}, \\l_{j1} &= \frac{a_{j1}}{l_{11}}, \quad j \in [2, n], \\l_{ii} &= \sqrt{a_{ii} - \sum_{p=1}^{i-1} l_{ip}^2}, \quad i \in [2, n], \\l_{ji} &= \frac{1}{l_{ii}} \left( a_{ji} - \sum_{p=1}^{i-1} l_{ip} l_{jp} \right), \quad i \in [2, n-1], j \in [i+1, n].\end{aligned}$$



# Разложение Холецкого

---

Теперь, когда получено разложение Холецкого, решение системы  $Ax=b$  сводится к решению  $Lx=b$  и  $L^T y=x$ , а решение системы для треугольной матрицы тривиально.

$$\begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{pmatrix} \begin{pmatrix} 2 & 6 & -8 \\ 0 & 1 & 5 \\ 0 & 0 & 3 \end{pmatrix}.$$

Можно показать, что при решении систем уравнений через разложение Холецкого требуется примерно вдвое меньше арифметических операций, чем при решении стандартным методом Гаусса.

# СПИСОК ИСТОЧНИКОВ

---

- <https://heartbeat.fritz.ai/applications-of-matrix-decompositions-for-machine-learning-f1986d03571a>
- <https://github.com/esokolov/ml-course-hse/blob/master/2019-fall/lecture-notes/lecture12-factorizations.pdf>
- *William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery. 2.9 Cholesky Decomposition // Numerical Recipes in C. — 2nd edition. — Cambridge: Cambridge University Press. — ISBN 0-521-43108-5.*
- [https://www.researchgate.net/publication/277930278\\_Face\\_Recognition\\_using\\_Eigenfaces](https://www.researchgate.net/publication/277930278_Face_Recognition_using_Eigenfaces)
- <https://qastack.ru/stats/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues>