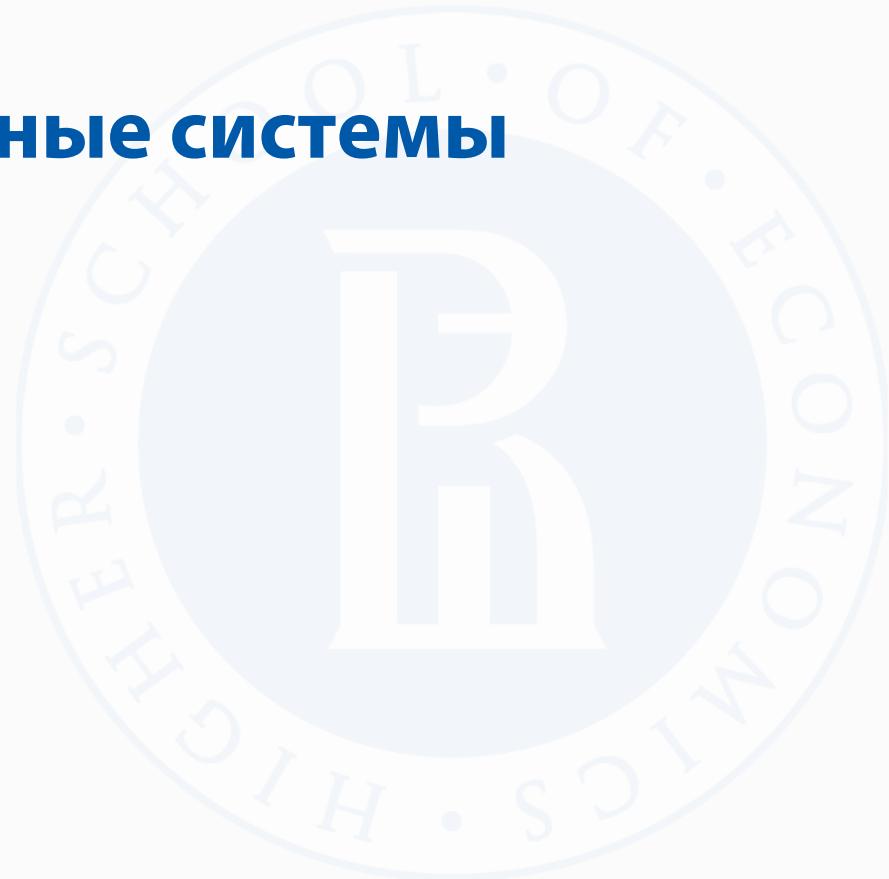


# **Рекомендательные системы**



# Search vs Discovery



Говорят, что Интернет покидает эпоху **поиска** и входит в эпоху **открытий**. В чем разница? Поиск — это когда вы ищете что-то. Открытие — это когда что-то замечательное, о существовании которого вы не знали, находит вас.

CNN Money

# Видеопрокат в Netflix

NETFLIX NEW EPISODES

Browse ▾ Kids DVD

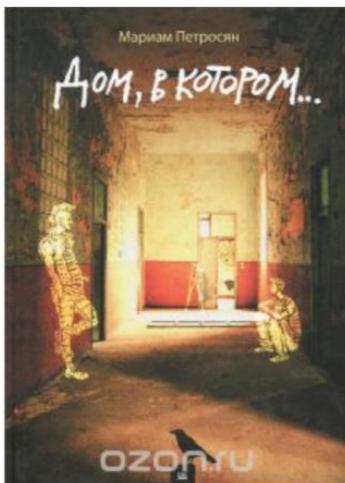
Because you watched New Girl



TV Mysteries



# Покупки на Ozon



## Дом, в котором...

ID 24277965

Новинка    Бестселлер

★★★★★ (155 отзывов)    566    189    У меня это есть

Автор: Мария Петросян

Издательство: Гаятри/Livebook

ISBN 978-5-904584-69-6; 2015 г.

Язык: Русский

[Дополнительные характеристики ▾](#)

Рекомендуем также



Дом, в котором... В  
3 томах (комплект)

509,60 ₽

[В корзину](#)



Тринадцатая  
сказка

332 ₽

[В корзину](#)



Дом странных  
детей

326,40 ₽

[В корзину](#)



Дом, в котором...  
164,90 ₽

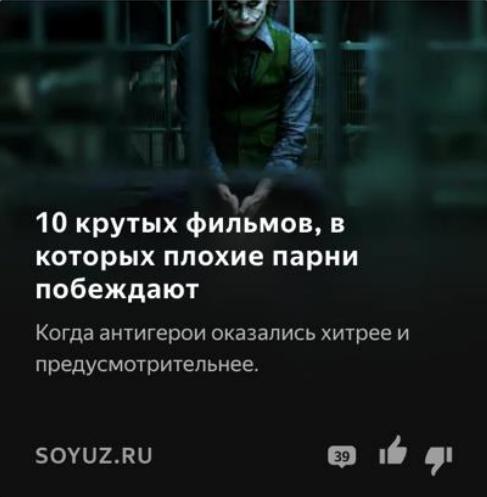
[Скачать](#)



Убить  
пересмешника...  
287,20 ₽

[В корзину](#)

# Чтение в Яндекс Дзен



**10 крутых фильмов, в которых плохие парни побеждают**

Когда антигерои оказались хитрее и предусмотрительнее.

SOYUZ.RU

39      

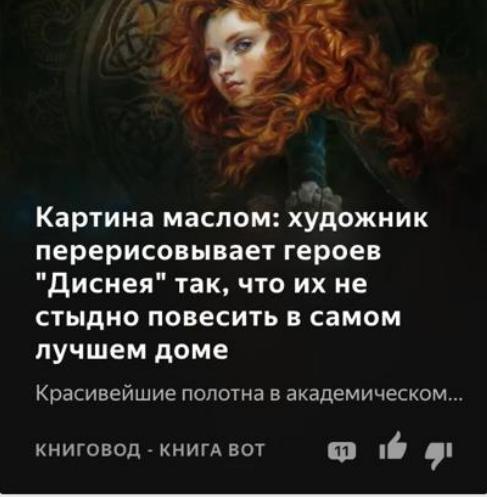


**7 убийств Бэтмена**

Герой, которого мы не заслужили, и который нам и не нужен

RUCONSOLE

3      



**Картина маслом: художник перерисовывает героев "Диснея" так, что их не стыдно повесить в самом лучшем доме**

Красивейшие полотна в академическом...

книговод - книга вот

11      

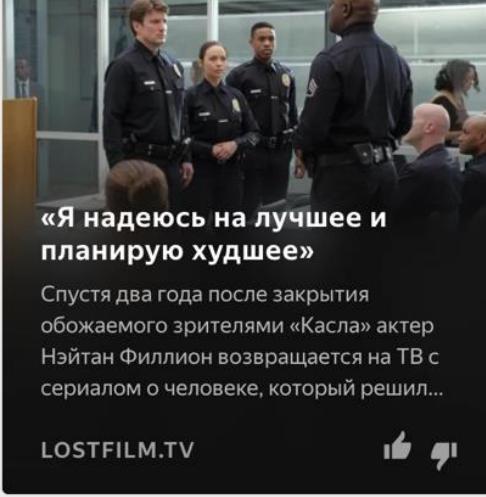


**Ридли Скотт начал работу над продолжением «Гладиатора»**

Ридли Скотт и студия Paramount начали работу над сиквелом «Гладиатора», сообщает Deadline. Сценарий...

КИНОПОИСК

1      



**«Я надеюсь на лучшее и планирую худшее»**

Спустя два года после закрытия обожаемого зрителями «Касла» актер Нэйтан Филлион возвращается на ТВ с сериалом о человеке, который решил...

LOSTFILM.TV

1

# Зачем нужны рекомендательные системы?



Вокруг слишком много информации,  
а рекомендации помогают найти нужное:

**NETFLIX** 2/3 фильмов просматриваются  
из рекомендаций



# Зачем нужны рекомендательные системы?



Вокруг слишком много информации,  
а рекомендации помогают найти нужное:

## NETFLIX

2/3 фильмов просматриваются  
из рекомендаций



## Google news

рекомендации генерируют  
на 38% больше кликов

# Зачем нужны рекомендательные системы?



Вокруг слишком много информации,  
а рекомендации помогают найти нужное:

## NETFLIX

2/3 фильмов просматриваются  
из рекомендаций



## Google news

рекомендации генерируют  
на 38% больше кликов

## amazon

35% покупок совершаются  
через рекомендации

# \$1,000,000 за улучшение алгоритма на 10%

www.netflixprize.com/leaderboard

**NETFLIX**

## Netflix Prize

**COMPLETED**

Home Rules Leaderboard Update

## Leaderboard

Showing Test Score. [Click here to show quiz score](#)

Display top  leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	<a href="#">BellKor's Pragmatic Chaos</a>	0.8567	10.06	2009-07-26 18:18:28
2	<a href="#">The Ensemble</a>	0.8567	10.06	2009-07-26 18:38:22
3	<a href="#">Grand Prize Team</a>	0.8582	9.90	2009-07-10 21:24:40
4	<a href="#">Opera Solutions and Vandelay United</a>	0.8588	9.84	2009-07-10 01:12:31
5	<a href="#">Vandelay Industries !</a>	0.8591	9.81	2009-07-10 00:32:20
6	<a href="#">PragmaticTheory</a>	0.8594	9.77	2009-06-24 12:06:56
7	<a href="#">BellKor in RioChaos</a>	0.8601	9.70	2009-05-13 08:14:09

# Рекомендательная система

Товар



Пользователь



	2		2	4	5	
	5		4			1
			5		2	
		1		5		4
			4			2
	4	5		1		

# Рекомендательная система

Товар



Пользователь



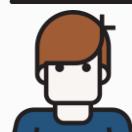
2



5



2



1



5



4

4

5

5

1

1

5

2

1

4

2

Рейтинг



# Рекомендательная система

Товар



Пользователь



2		2	4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

Рейтинг



Что тут?

# Два типа рейтингов

It's ok

I love it

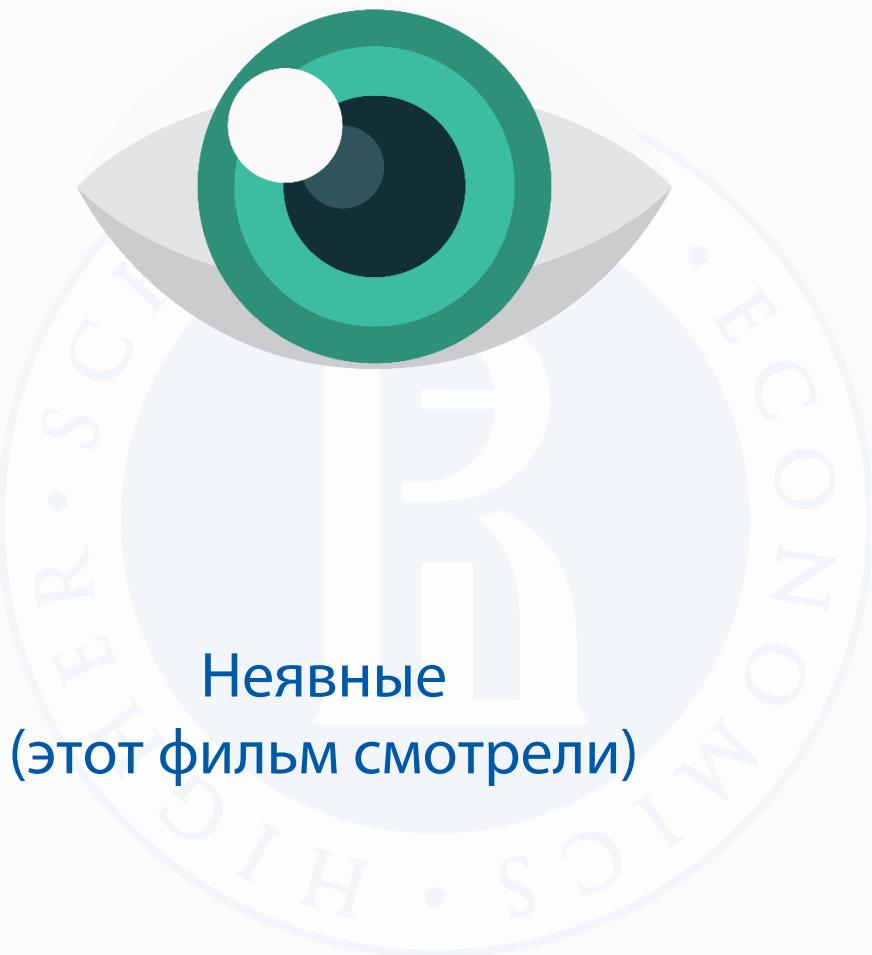
I like it



I hate it!

I don't like it

Явные  
(от 1 до 5)



Неявные  
(этот фильм смотрели)

# Резюме



Рекомендации помогают пользователям находить интересное им



# Резюме



Рекомендации помогают пользователям находить интересное им



В общем виде задача рекомендательной системы — это заполнение пропусков в матрице оценок

# Резюме



Рекомендации помогают пользователям находить интересное им



В общем виде задача рекомендательной системы — это заполнение пропусков в матрице оценок



Рейтинги бывают двух видов (явные и неявные) и для них нужны разные алгоритмы

# Резюме



Рекомендации помогают пользователям находить интересное им



В общем виде задача рекомендательной системы — это заполнение пропусков в матрице оценок



Рейтинги бывают двух видов (явные и неявные) и для них нужны разные алгоритмы

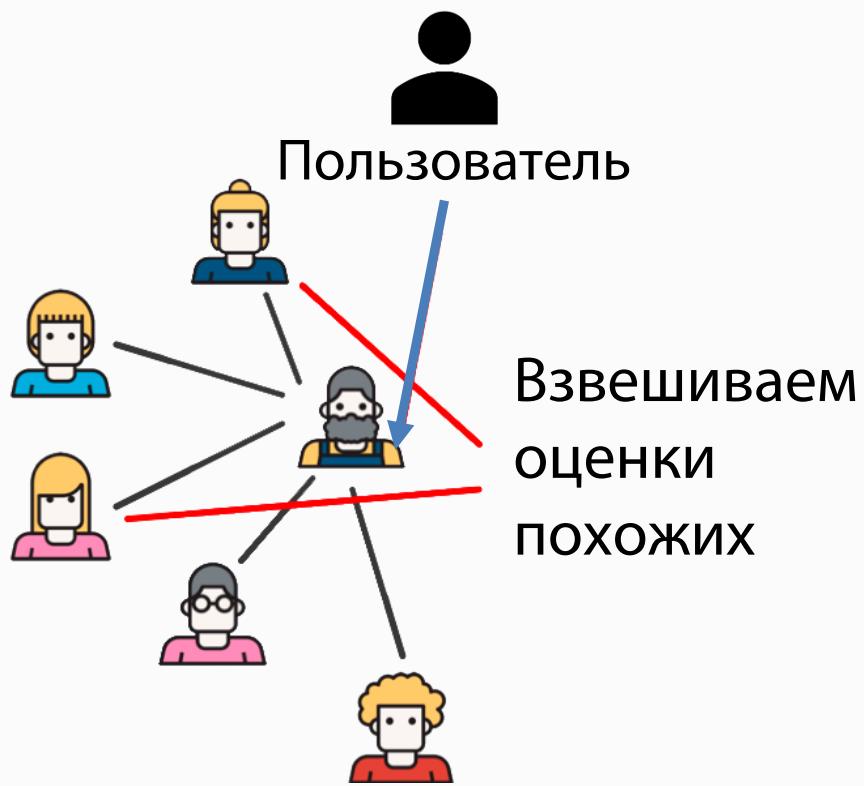


Далее: рассмотрим, как масштабировать классическую коллаборативную фильтрацию

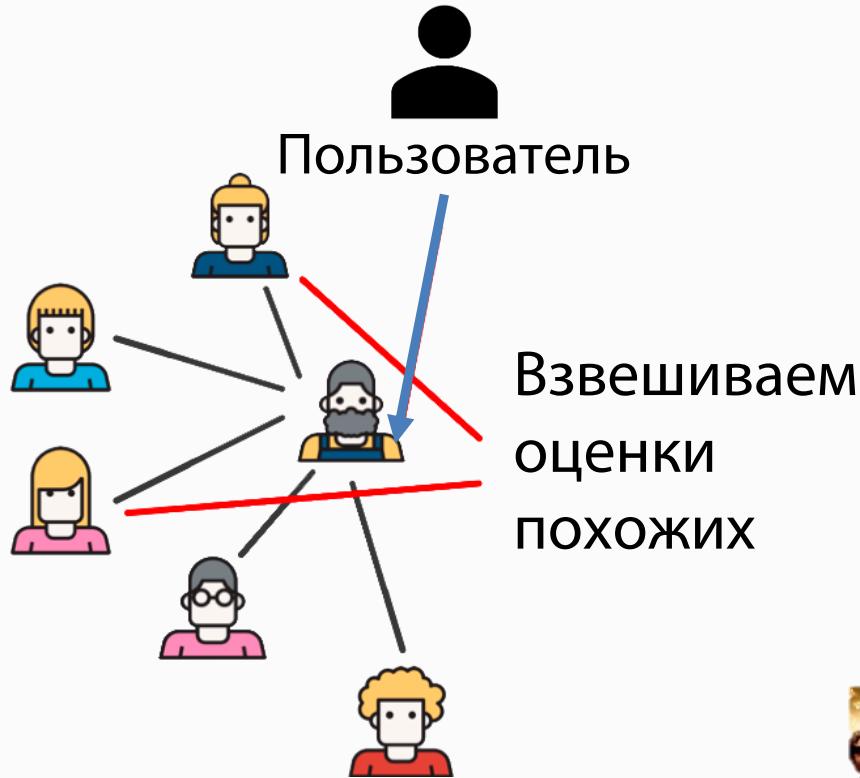
# **Коллаборативная фильтрация (Collaborative Filtering)**



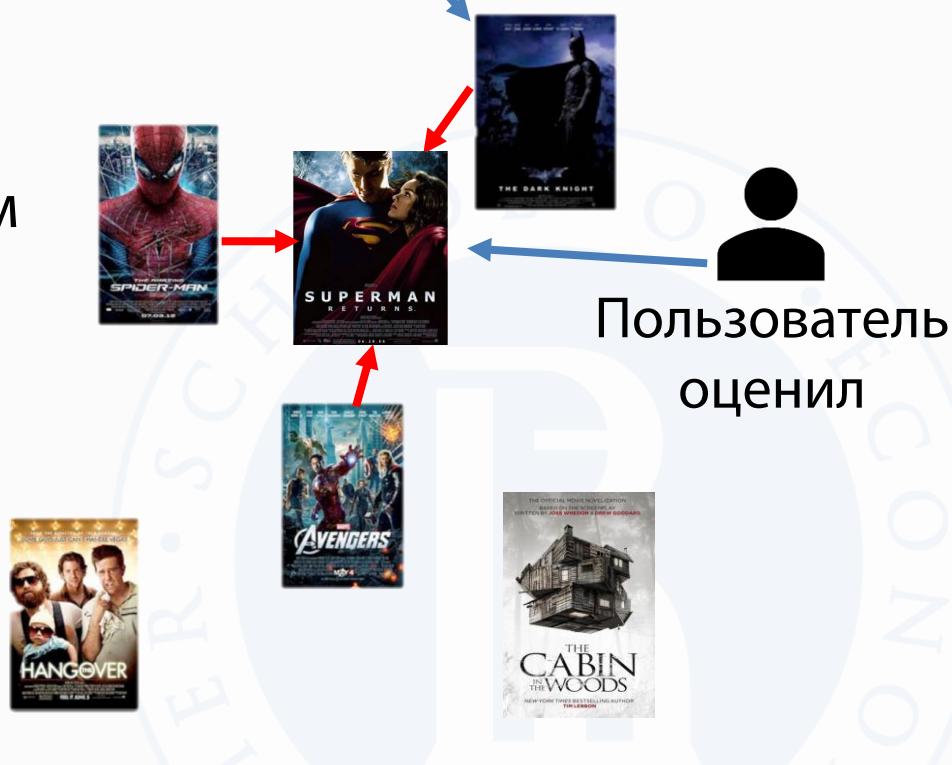
# User-based и item-based



# User-based и item-based



Рекомендуем похожий фильм



# User-based или item-based?

→ Рейтингов у пользователя мало →  
в пересечении с другими еще меньше →  
неуверенные похожести пользователей

# User-based или item-based?

- Рейтингов у пользователя мало →  
в пересечении с другими еще меньше →  
неуверенные похожести пользователей
- При появлении новой оценки похожести могут  
сильно измениться → не получится посчитать  
заранее

# Прикинем на примере

- 10000 рейтингов
- 1000 пользователей
- 100 товаров
- Рейтинги распределены равномерно в таблице

# Прикинем на примере

- 10000 рейтингов
  - 1000 пользователей
  - 100 товаров
  - Рейтинги распределены равномерно в таблице
- Сколько общих оценок в среднем у двух случайных пользователей?
- А у двух случайных товаров?

# Прикинем на примере

- 10000 рейтингов
- 1000 пользователей
- 100 товаров
- Рейтинги распределены равномерно в таблице

- Сколько общих оценок в среднем у двух случайных пользователей? (1)
- А у двух случайных товаров? (10)

# Будем использовать item-based CF

- Посчитаем для каждого товара в оффлайне 1000 самых похожих на него, сложим в Key-Value хранилище (например, Redis)

# Будем использовать item-based CF

- Посчитаем для каждого товара в оффлайне 1000 самых похожих на него, сложим в Key-Value хранилище (например, Redis)
- В онлайне возьмем оцененные пользователем товары (последние 100) и для них сходим в Key-Value за похожими

# Будем использовать item-based CF

- Посчитаем для каждого товара в оффлайне 1000 самых похожих на него, сложим в Key-Value хранилище (например, Redis)
- В онлайне возьмем оцененные пользователем товары (последние 100) и для них сходим в Key-Value за похожими
- Посчитаем итоговые рекомендации по формуле

$$\hat{r}_{ui} = \frac{\sum_j s(i,j)r_{uj}}{\sum_j |s(i,j)|}$$

# Резюме



Классическая колаборативная фильтрация работает на основе похожестей



# Резюме



Классическая колаборативная фильтрация работает на основе похожестей



Будем использовать item-based CF (надежнее похожести и их можно посчитать заранее в онлайне)

# Резюме



Классическая колаборативная фильтрация работает на основе похожестей



Будем использовать item-based CF (надежнее похожести и их можно посчитать заранее в онлайне)



Далее: поговорим о том, как считать похожести товаров на больших данных

# **Похожесть товаров**



# Похожесть товаров для явных рейтингов



	2		2	4	5	
	5		4			1
			5		2	
		1		5		4
			4			2
	4	5		1		

Похожи?

# Похожесть товаров для явных рейтингов



	2		2	4	5	
	5		4			1
			5		2	
		1		5		4
			4			2
	4	5		1		

Посмотрим только на общих пользователей.  
Как сравнить векторы?

# Косинусная мера

→ Важен только угол между векторами:

$$s(i, j) = \frac{\sum_{u \in U} r_{ui} r_{uj}}{\sqrt{\sum_{u \in U} r_{ui}^2} \sqrt{\sum_{u \in U} r_{uj}^2}}$$

# Косинусная мера

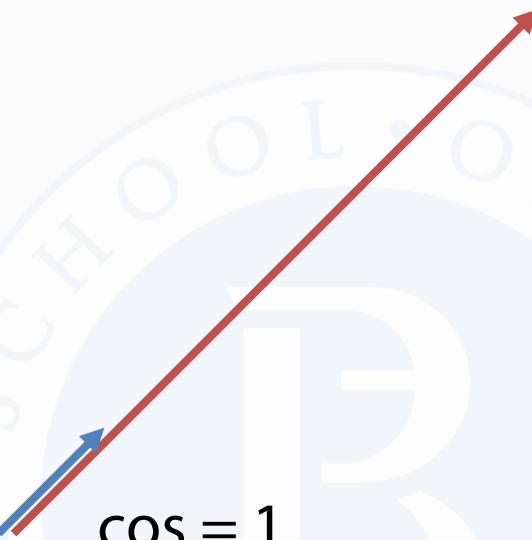
→ Важен только угол между векторами:

$$s(i, j) = \frac{\sum_{u \in U} r_{ui} r_{uj}}{\sqrt{\sum_{u \in U} r_{ui}^2} \sqrt{\sum_{u \in U} r_{uj}^2}}$$

→ Суммирование должно быть только по общим пользователям!

# Проблема

→ Товары [1,1] и [5,5] считает максимально близкими!



# Adjusted cosine similarity

→ Работает лучше (поправка на средний рейтинг пользователя):

$$s(i, j) = \frac{\sum_{u \in U} (r_{ui} - r_u)(r_{uj} - r_u)}{\sqrt{\sum_{u \in U} (r_{ui} - r_u)^2} \sqrt{\sum_{u \in U} (r_{uj} - r_u)^2}}$$

# Adjusted cosine similarity

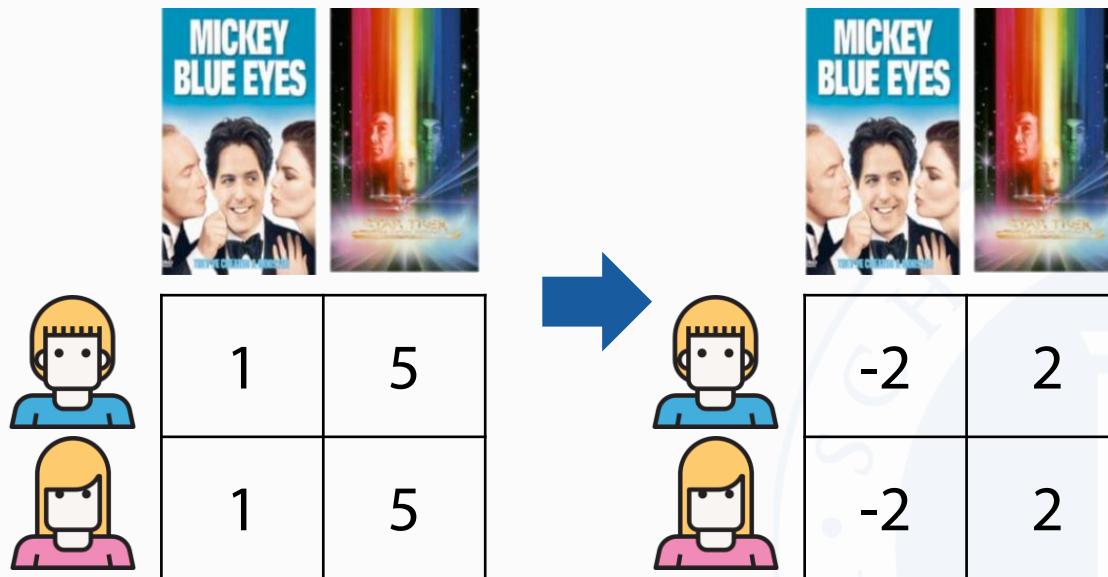
→ Работает лучше (поправка на средний рейтинг пользователя):

$$s(i, j) = \frac{\sum_{u \in U} (r_{ui} - r_u)(r_{uj} - r_u)}{\sqrt{\sum_{u \in U} (r_{ui} - r_u)^2} \sqrt{\sum_{u \in U} (r_{uj} - r_u)^2}}$$

→ Суммирование должно быть только по общим пользователям!

# И проблемы больше нет

→ Товары [1,1] и [5,5]:



$$\cos = -1$$

# Посчитаем все числители на MapReduce



Будем обновлять расчеты раз в день при помощи инвертированного индекса:

# Посчитаем все числители на MapReduce



Будем обновлять расчеты раз в день при помощи инвертированного индекса:

$$s(i, j) = \frac{\sum_{u \in U} (r_{ui} - r_u)(r_{uj} - r_u)}{\dots}$$



	2		2	4	5	
	5		4			1
			5		2	
		1		5		4
			4			2
	4	5		1		

Map шаг:

$$(1, 3) \rightarrow (5 - 3.3) * (4 - 3.3)$$

$$(1, 6) \rightarrow (5 - 3.3) * (1 - 3.3)$$

$$(3, 6) \rightarrow (4 - 3.3) * (1 - 3.3)$$

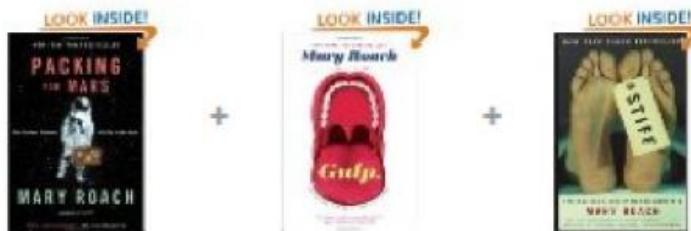
Reduce шаг:

Суммируем по ключу

# Похожести для неявных рейтингов

→ Например, рассмотрим покупку (неявный рейтинг)

## Frequently Bought Together



Price for all three: \$30.62

[Add all three to Cart](#)

[Add all three to Wish List](#)

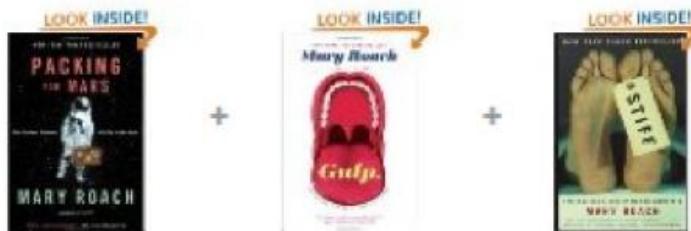
[Show availability and shipping details](#)

- This item: **Packing for Mars: The Curious Science of Life in the Void** by Mary Roach Hardcover \$10.38
- Gulp: Adventures on the Alimentary Canal** by Mary Roach Paperback \$10.34
- Stiff: The Curious Lives of Human Cadavers** by Mary Roach Paperback \$9.90

# Похожести для неявных рейтингов

→ Например, рассмотрим покупку (неявный рейтинг)

## Frequently Bought Together



Price for all three: \$30.62

Add all three to Cart

Add all three to Wish List

Show availability and shipping details

- This item: **Packing for Mars: The Curious Science of Life in the Void** by Mary Roach Hardcover \$10.38
- Gulp: Adventures on the Alimentary Canal** by Mary Roach Paperback \$10.34
- Stiff: The Curious Lives of Human Cadavers** by Mary Roach Paperback \$9.90

→ Что такое «часто покупают вместе»?

# Количество совместных покупок?

- Бестселлеры типа «Гарри Поттер» покупают очень много людей

# Количество совместных покупок?

- Бестселлеры типа «Гарри Поттер» покупают очень много людей
- Получается, что «Гарри Поттер» часто покупают со всеми остальными книгами

# Количество совместных покупок?

- Бестселлеры типа «Гарри Поттер» покупают очень много людей
- Получается, что «Гарри Поттер» часто покупают со всеми остальными книгами
- Но не рекомендовать же всем «Гарри Поттера» к любой книге?

# Мера Жаккара для множеств



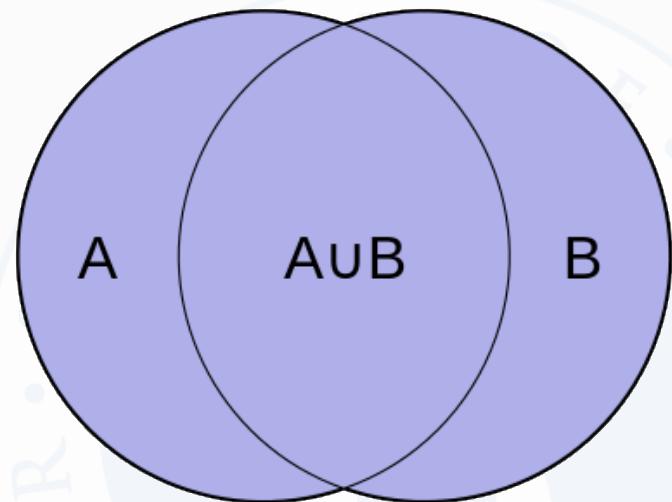
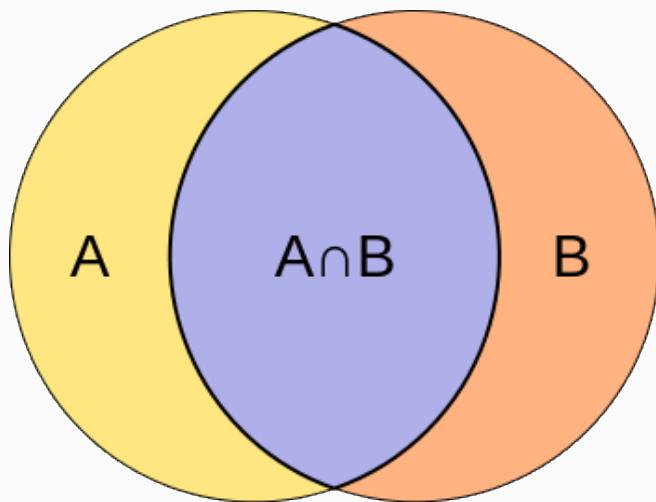
Учитывает популярности сравниваемых товаров:

# Мера Жаккара для множеств



Учитывает популярности сравниваемых товаров:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$



— пользователи, купившие товар А

# На MapReduce считается точно так же

- Числитель в мере Жаккара — это скалярное произведение столбцов в матрице покупок:

# На MapReduce считается точно так же



Числитель в мере Жаккара — это скалярное произведение столбцов в матрице покупок:

	1		1	1	1	
	1		1			1
			1		1	
		1		1		1
			1			1
	1	1		1		

$$J(A, B) = \frac{|A \cap B|}{\dots}$$

Map шаг:

(1, 3) → 1

(1, 6) → 1

(3, 6) → 1

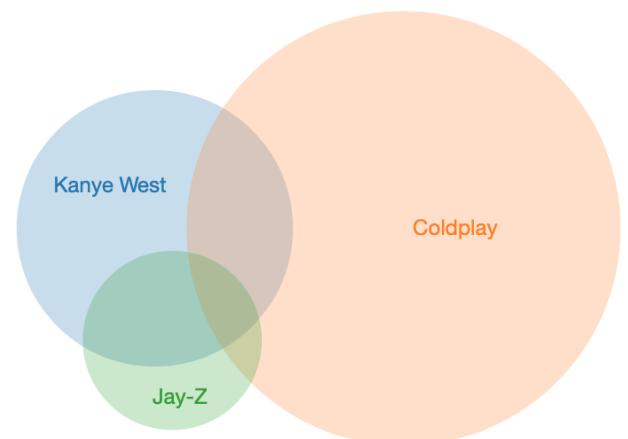
Reduce шаг:

Суммируем по ключу

# Пример

→ Самые похожие на Kanye West по пересечению:

Artist	Overlap
Coldplay	8061
Jay-Z	6851
The Beatles	6139
Radiohead	6135
Eminem	5454
<a href="#">more ...</a>	

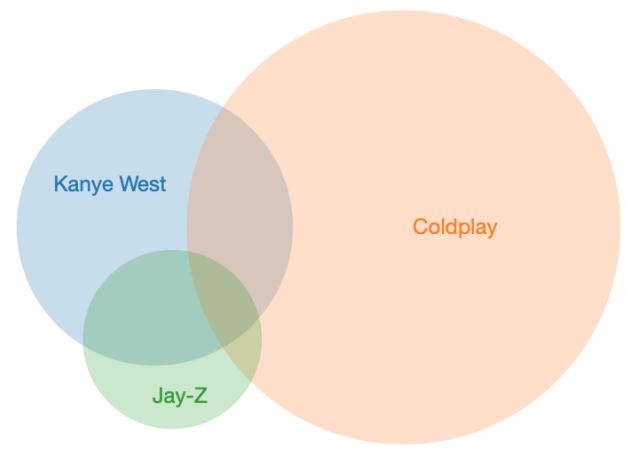


# Пример

→ Самые похожие на Kanye West по пересечению:

Artist	Overlap
Coldplay	8061
Jay-Z	6851
The Beatles	6139
Radiohead	6135
Eminem	5454

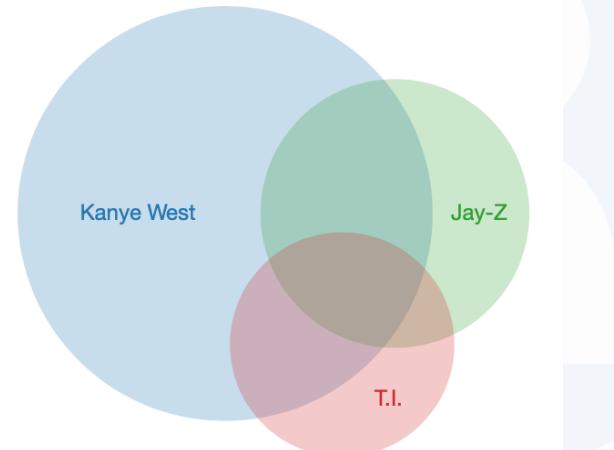
[more ...](#)



→ Самые похожие на Kanye West по мере Жаккара:

Artist	Jaccard
Jay-Z	0.217
T.I.	0.163
Lupe Fiasco	0.156
Nas	0.156
Common	0.139

[more ...](#)



# Резюме



CF дает неплохие рекомендации при большом количестве оценок

# Резюме



CF дает неплохие рекомендации при большом количестве оценок



Плохо работает при сильной разреженности матрицы оценок (неуверенные похожести)

# Резюме



CF дает неплохие рекомендации при большом количестве оценок



Плохо работает при сильной разреженности матрицы оценок (неуверенные похожести)



Два пользователя должны оценивать одинаковые товары, оценка **сильно похожих** товаров не учитывается в их близости

# Резюме



CF дает неплохие рекомендации при большом количестве оценок



Плохо работает при сильной разреженности матрицы оценок (неуверенные похожести)



Два пользователя должны оценивать одинаковые товары, оценка **сильно похожих** товаров не учитывается в их близости



Хорошо масштабируется на большие данные

# **Матричные факторизации (Matrix Factorization)**



# Уменьшение размерности

→ Матрица рейтингов  
разреженная (много  
пропусков)



2		2	4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

# Уменьшение размерности

→ Матрица рейтингов  
разреженная (много  
пропусков)

→ Оценки товаров  
коррелируют  
(пользователям  
часто нравятся одни  
и те же пары товаров)



2		2	4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

# Уменьшение размерности

- Матрица рейтингов разреженная (много пропусков)
- Оценки товаров коррелируют (пользователям часто нравятся одни и те же пары товаров)
- Можно сжать матрицу оценок!



	2		2	4	5	
	5		4			1
			5		2	
		1		5		4
			4			2
	4	5		1		

# SVD для плотной матрицы

$$R = U\Sigma V^T$$

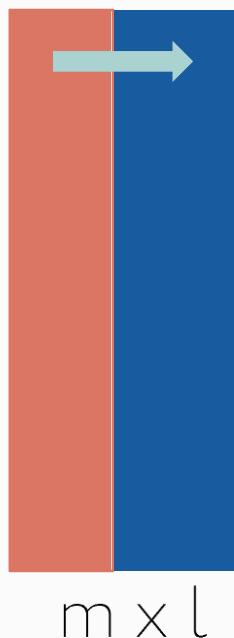
$$UU^T = U^T U = I \quad \leftarrow$$

$$VV^T = V^T V = I$$

единичная  
матрица



=

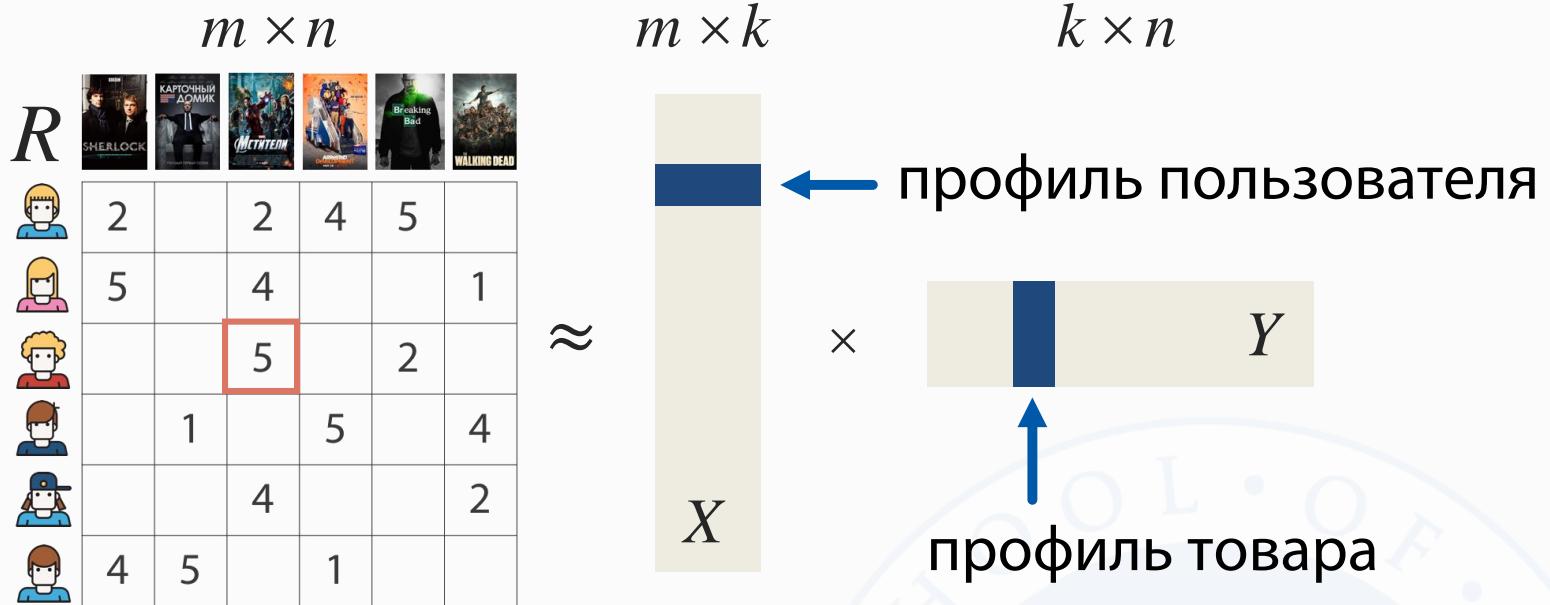


$$\begin{matrix} & \times & \end{matrix} \begin{matrix} \text{l} \times \text{l} \\ \times \end{matrix} \begin{matrix} \text{l} \times \text{n} \end{matrix}$$

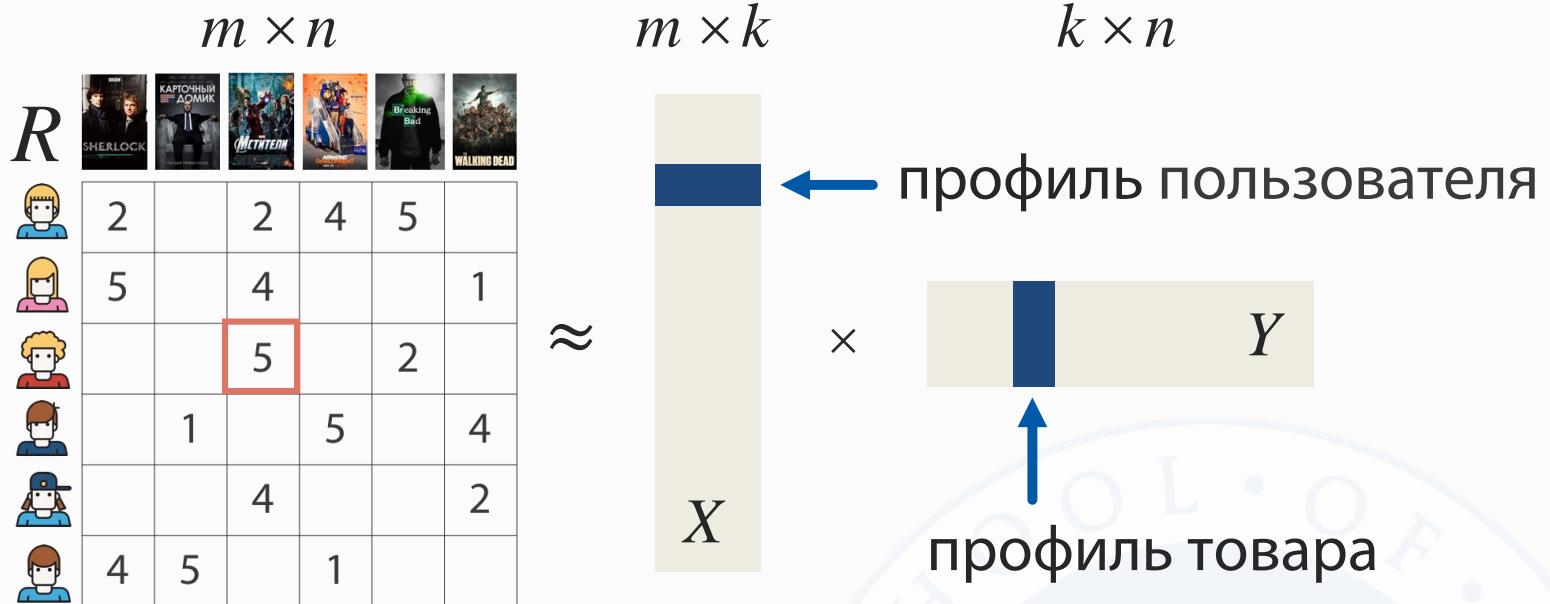


Векторы отсортированы по убыванию вклада  
в приближение матрицы!

# Funk SVD для разреженной матрицы

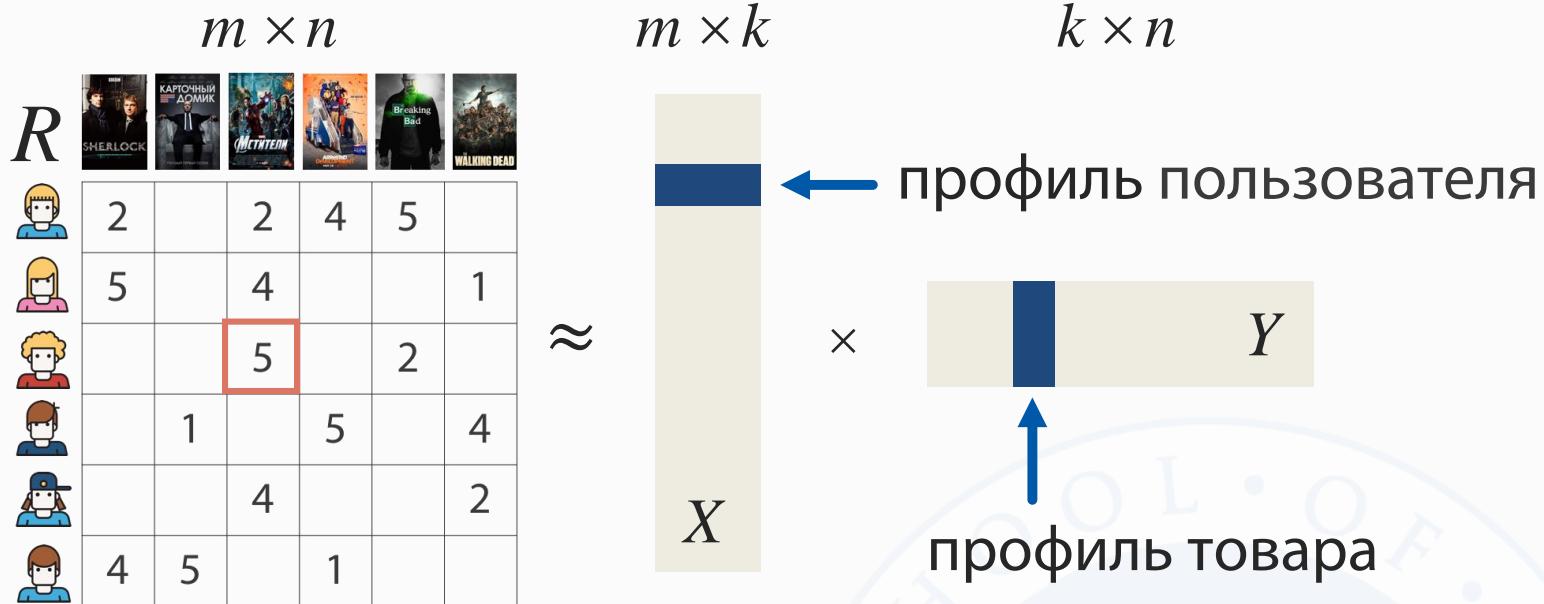


# Funk SVD для разреженной матрицы



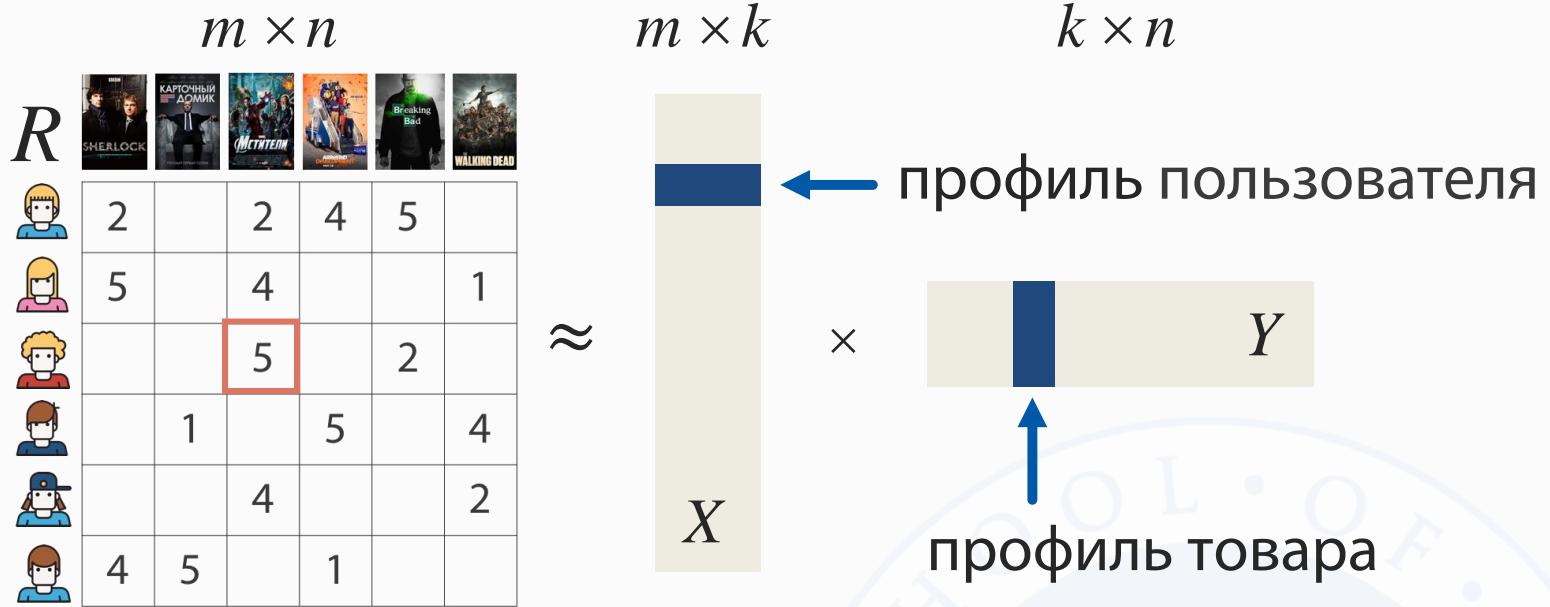
- Пользователи и товары погружаются в новое пространство (малой размерности  $k$ )

# Funk SVD для разреженной матрицы



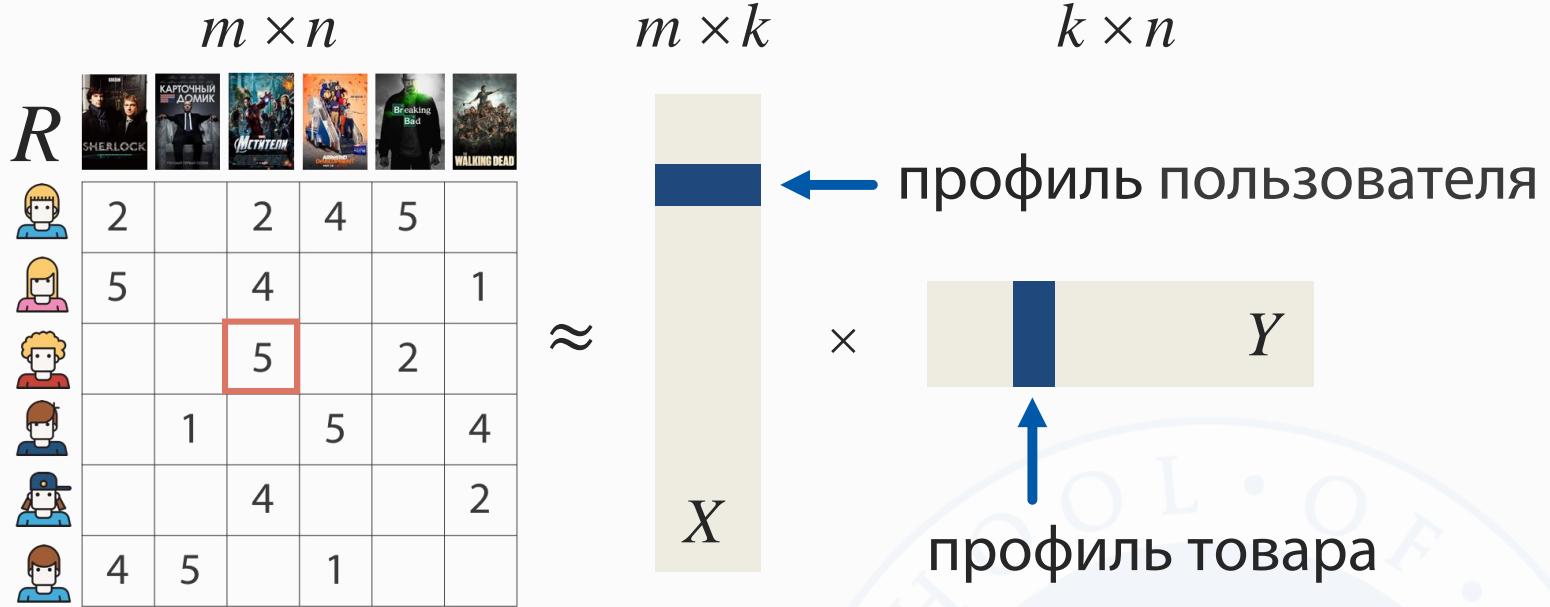
- Пользователи и товары погружаются в новое пространство (малой размерности  $k$ )
- Координаты в этом пространстве — «характеристики» товаров и пользователей

# Funk SVD для разреженной матрицы



- Пользователи и товары погружаются в новое пространство (малой размерности  $k$ )
- Координаты в этом пространстве — «характеристики» товаров и пользователей
- Похожесть в этом пространстве — это скалярное произведение

# Funk SVD для разреженной матрицы



Оптимизируем при помощи SGD:

$$\min_{X,Y} \sum_{(u,i) \in R} (r_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2)$$

# Интерпретация профилей товаров

Фантастичность

Funk SVD сам  
разместил товары  
в пространстве ( $k = 2$ )



# Можно оценить похожести товаров

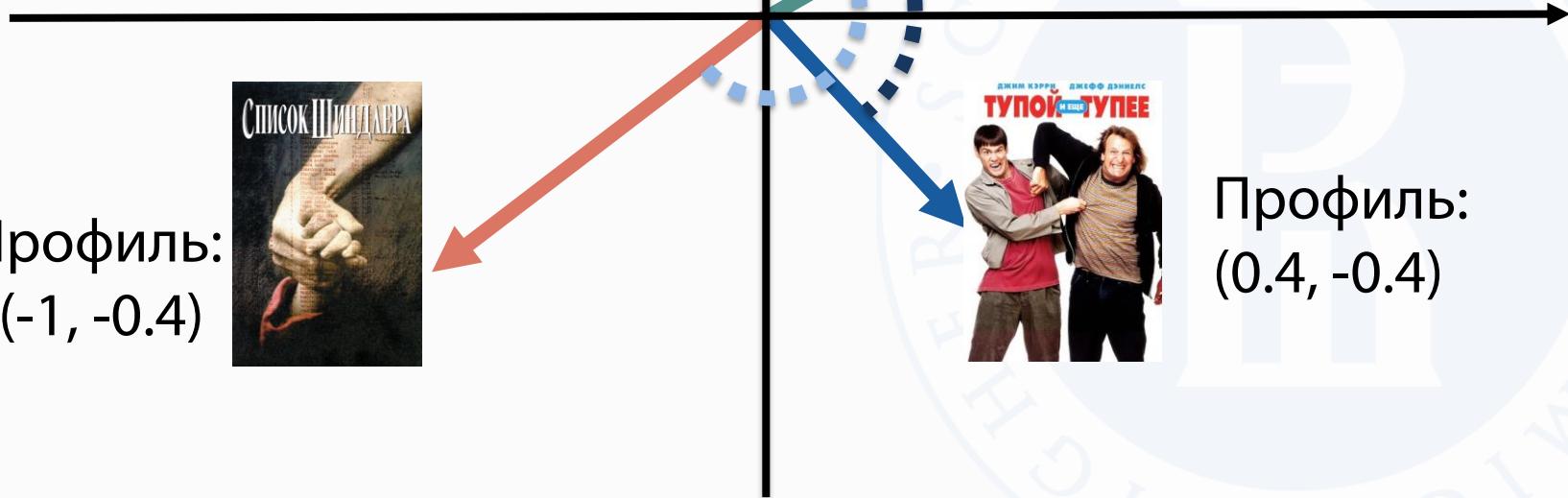
Фантастичность



$\cos$  между  
векторами



Профиль:  
(0.8, 0.4)



Профиль:  
(-1, -0.4)



Профиль:  
(0.4, -0.4)



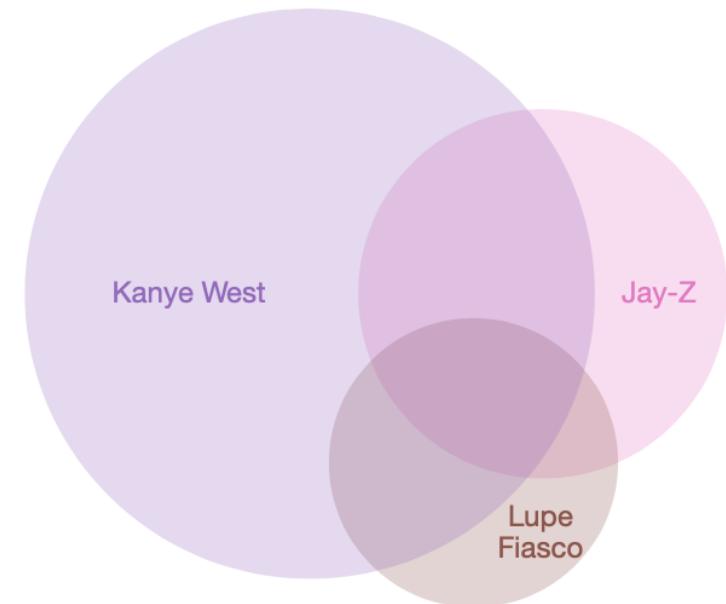
Комедийность

# Пример похожестей товаров



Самые похожие на Kanye West  
по косинусу между профилями исполнителей:

Artist	Implicit ALS
Lupe Fiasco	0.950
Jay-Z	0.928
T.I.	0.913
Lil Wayne	0.895
N*E*R*D	0.891
more ...	



# Профиль для нового пользователя



# Профиль для нового пользователя



# Рекомендация для пользователя



# Резюме



Матричные факторизации (MF) не считают похожести товаров напрямую (как в CF)

# Резюме



Матричные факторизации (MF) не считают похожести товаров напрямую (как в CF)



MF пытаются описать пользователей и товары маленьким набором характеристик, объясняющих оценки

# Резюме



Матричные факторизации (MF) не считают похожести товаров напрямую (как в CF)



MF пытаются описать пользователей и товары маленьким набором характеристик, объясняющих оценки



Эти характеристики могут быть не интерпретируемыми

# Резюме



Матричные факторизации (MF) не считают похожести товаров напрямую (как в CF)



MF пытаются описать пользователей и товары маленьkim набором характеристик, объясняющих оценки



Эти характеристики могут быть не интерпретируемыми

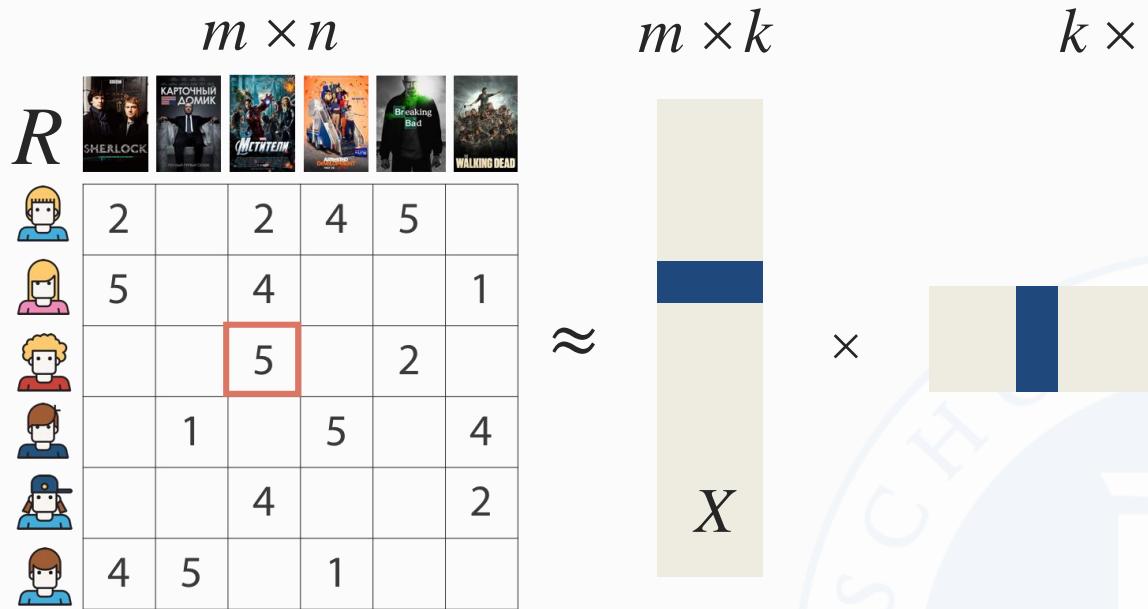


К сожалению, SGD сложно распараллелить, но [далее](#) мы решим эту проблему!

**ALS и iALS**



# Alternating Least Squares (ALS)



$$\min_{X,Y} \sum_{(u,i) \in R} (r_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2)$$

# Alternating Least Squares (ALS)

$$\min_{X,Y} \sum_{(u,i) \in R} (r_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2)$$

# Alternating Least Squares (ALS)

$$\min_{X,Y} \sum_{(u,i) \in R} (r_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2)$$

→ Инициализируем  $X$  и  $Y$  случайными значениями

# Alternating Least Squares (ALS)

$$\min_{X,Y} \sum_{(u,i) \in R} (r_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2)$$

- Инициализируем  $X$  и  $Y$  случайными значениями
- В цикле:
  - Фиксируем матрицу  $X$  (пользователей)

# Alternating Least Squares (ALS)

$$\min_{X,Y} \sum_{(u,i) \in R} (r_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2)$$

→ Инициализируем  $X$  и  $Y$  случайными значениями

→ В цикле:

- Фиксируем матрицу  $X$  (пользователей)
- Находим оптимальную матрицу  $Y$  (решаем гребневую регрессию для каждого товара)



User 1 profile

User 2 profile

User 3 profile

×

Item profile

5

≈ 1

4

- И наоборот

# Implicit ALS (iALS) для неявных оценок

Как долго читал статью

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \text{ (по умолчанию)} \end{cases}$$

Понравилось?

Если не читал,  
то уверенность  
в  $p_{ui} = 0$  маленькая

$$c_{ui} = 1 + \alpha r_{ui}$$

$$\min \sum_{u,i} c_{ui} (p_{ui} - x_u^T y_i)^2 + \lambda \left( \sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

Взвешенные потери

Сумма по всем ячейкам (пропусков нет)

# iALS: считается так же быстро

Решение регрессии

Диагональ с весами  
оценок пользователя

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u)$$

# iALS: считается так же быстро

Решение регрессии

Диагональ с весами  
оценок пользователя

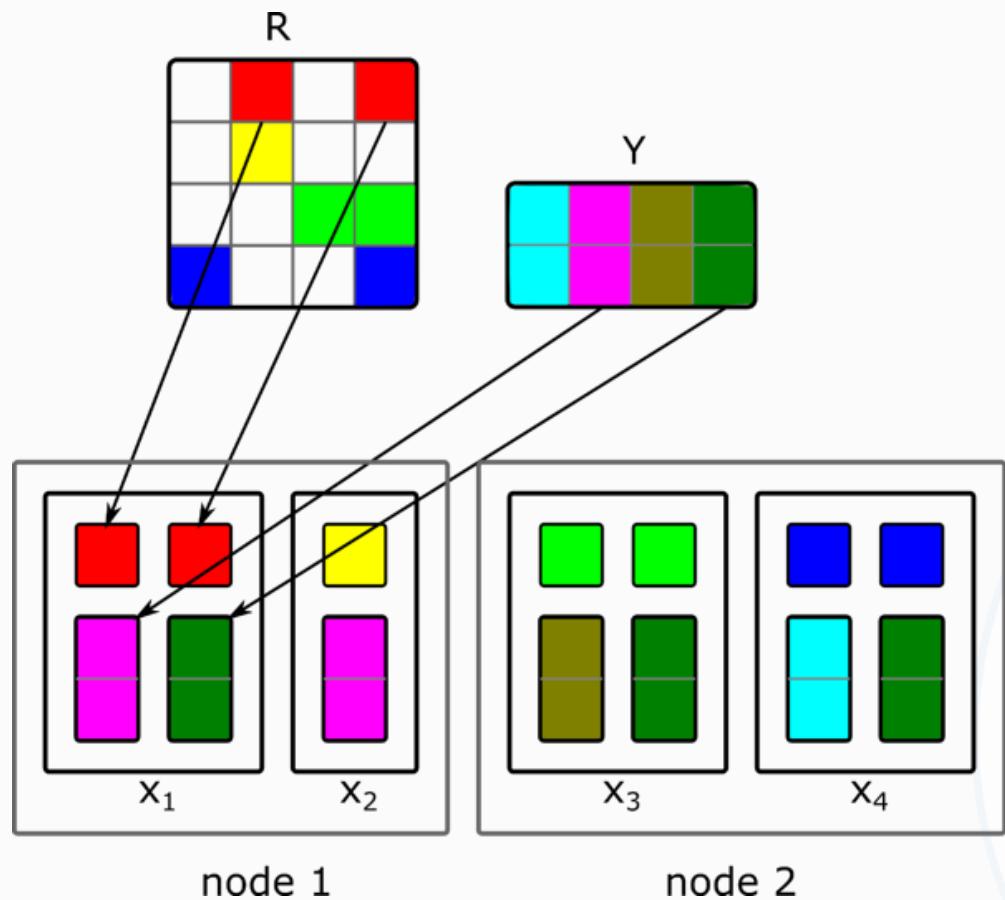
$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u)$$

$$Y^T C^u Y = Y^T Y + Y^T (C^u - I) Y$$

Считается один раз!

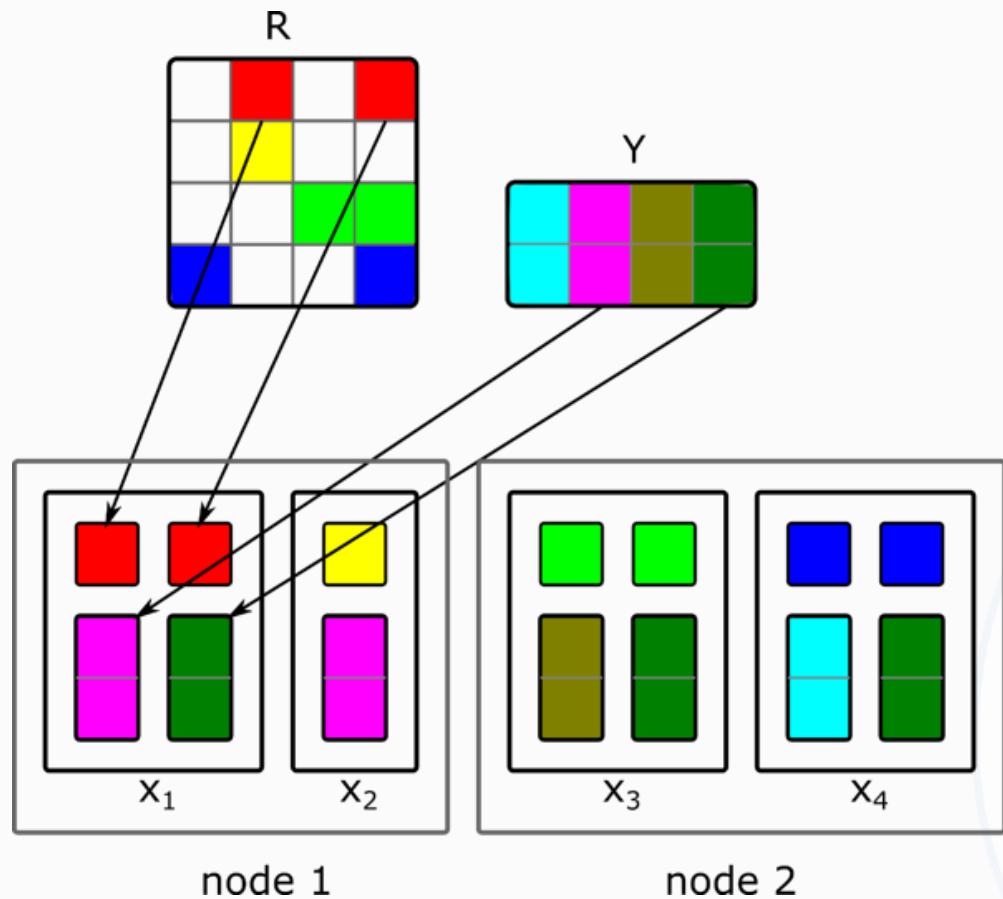
Не ноль только  
на прочтенных!

# Наивный параллельный ALS

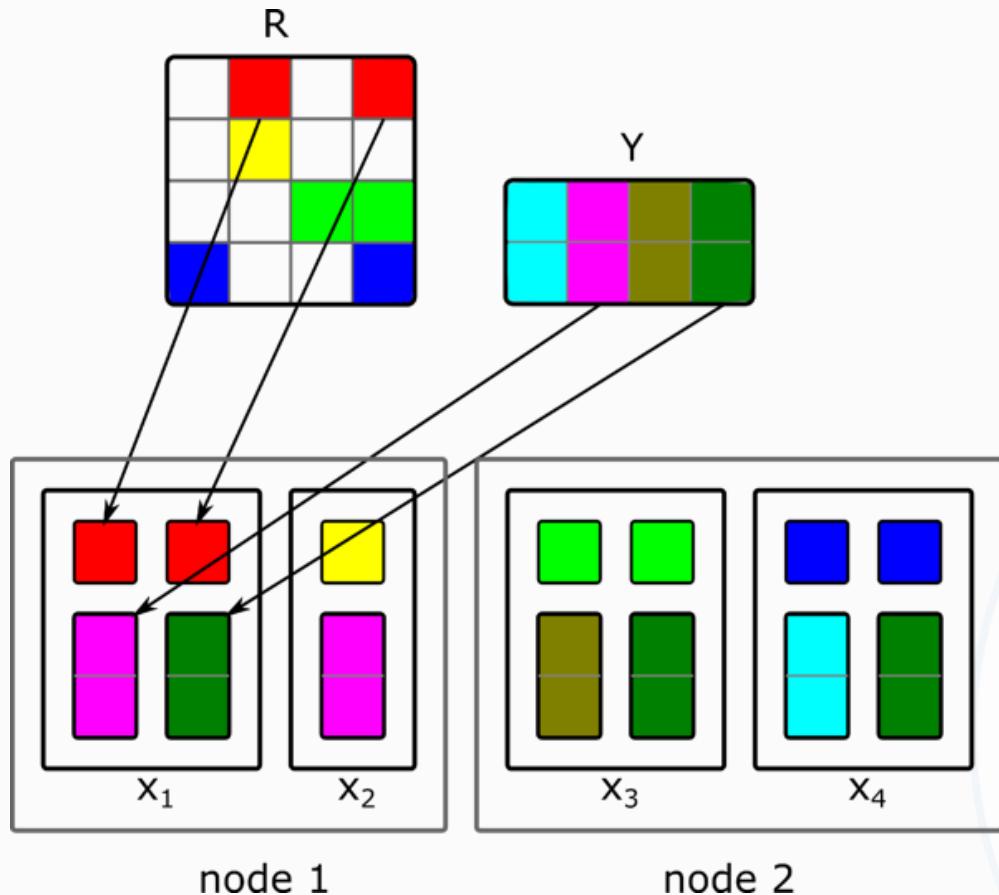


# Наивный параллельный ALS

- $R = XY$ , обновляем  $X$

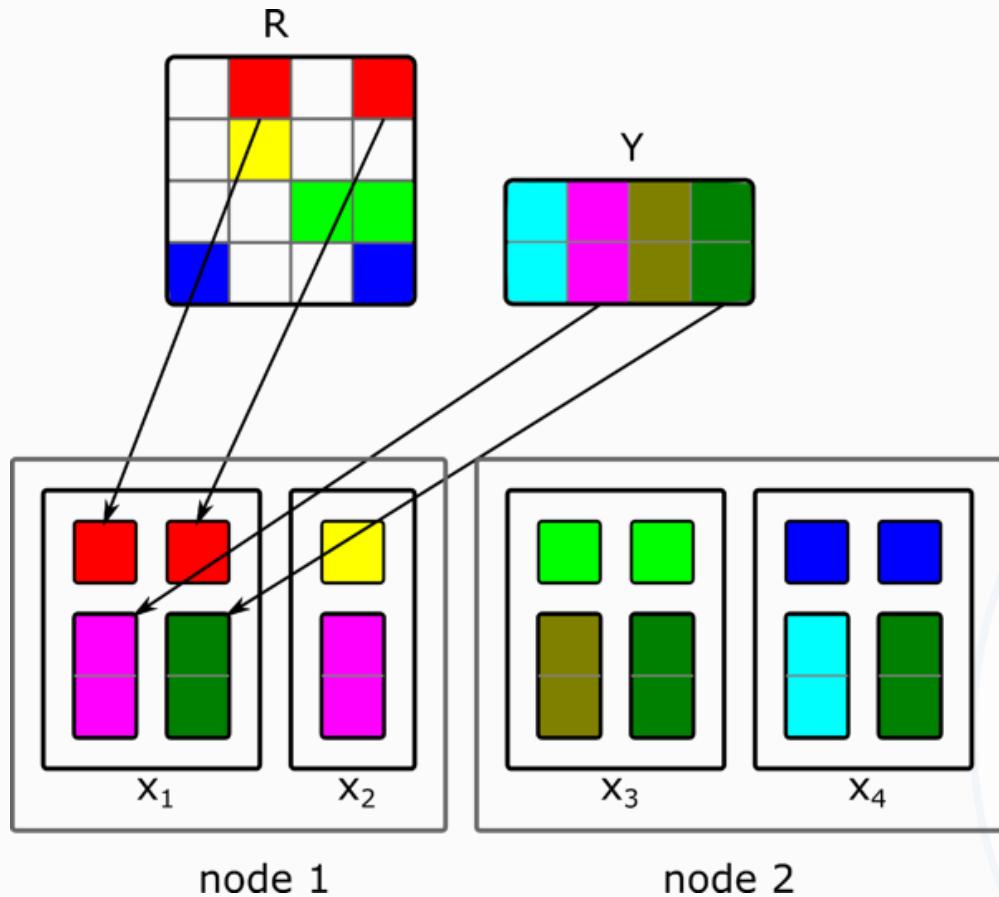


# Наивный параллельный ALS



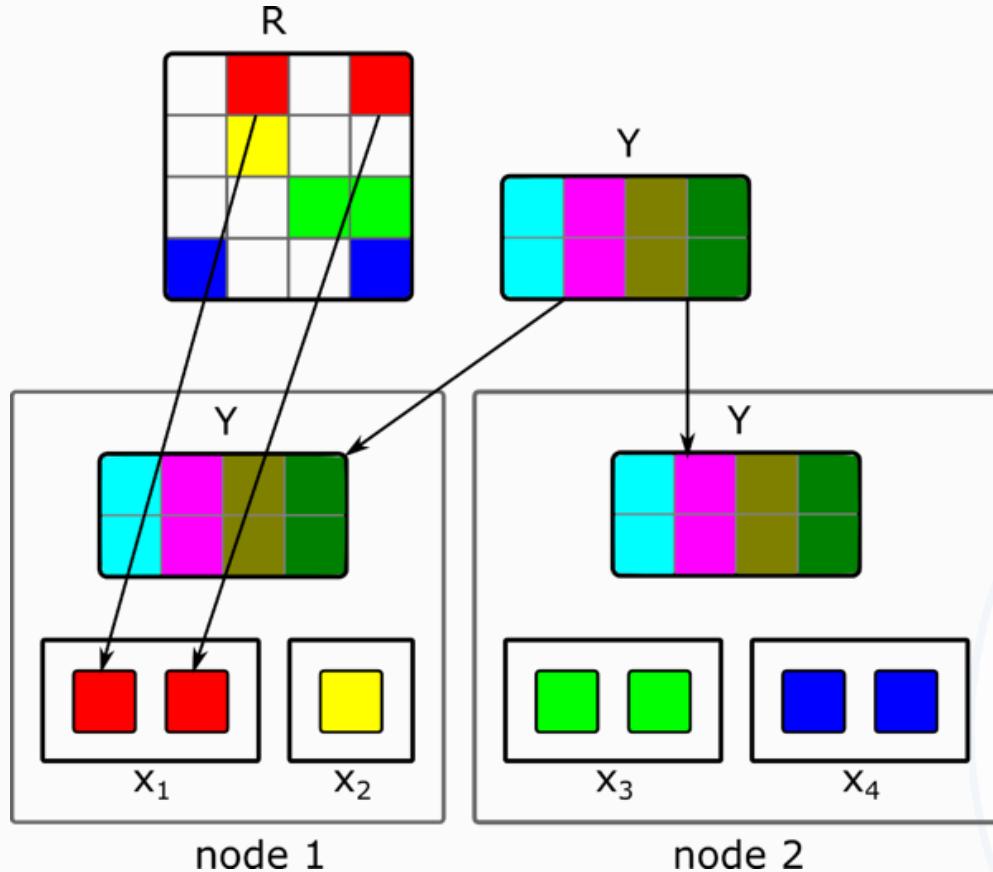
- $R = XY$ , обновляем  $X$
- Join  $R$  и  $Y$  по товарам  
→  $(u, r_{ui}, y_i)$
- Группируем по  $u$
- На каждой машине пересчитываем  $x_u$

# Наивный параллельный ALS



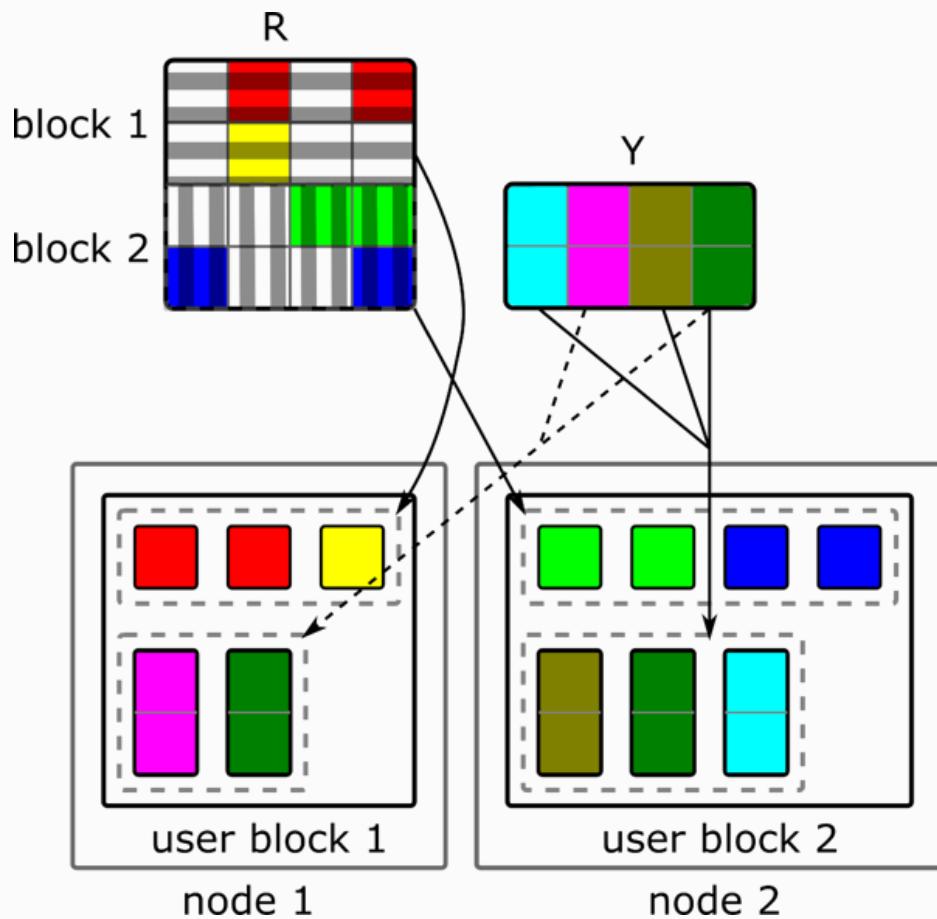
- $R = XY$ , обновляем  $X$
- Join  $R$  и  $Y$  по товарам  
→  $(u, r_{ui}, y_i)$
- Группируем по  $u$
- На каждой машине пересчитываем  $x_u$
- Один и тот же вектор  $y_i$  может понадобиться нескольким пользователям на машине, и будет отправлен несколько раз

# Реализация с broadcast Y



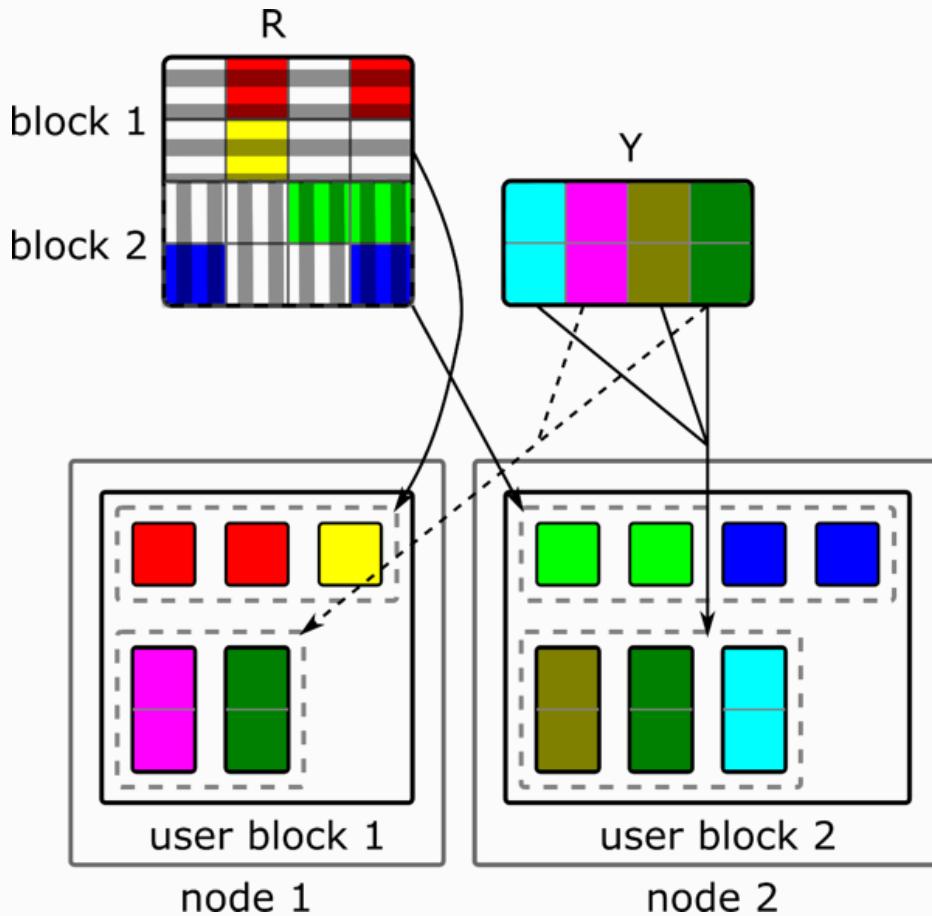
- Быстрее
- Имеет предел масштабирования ( $Y$  должен влезать в память)

# Блочный ALS (как в Spark ML)



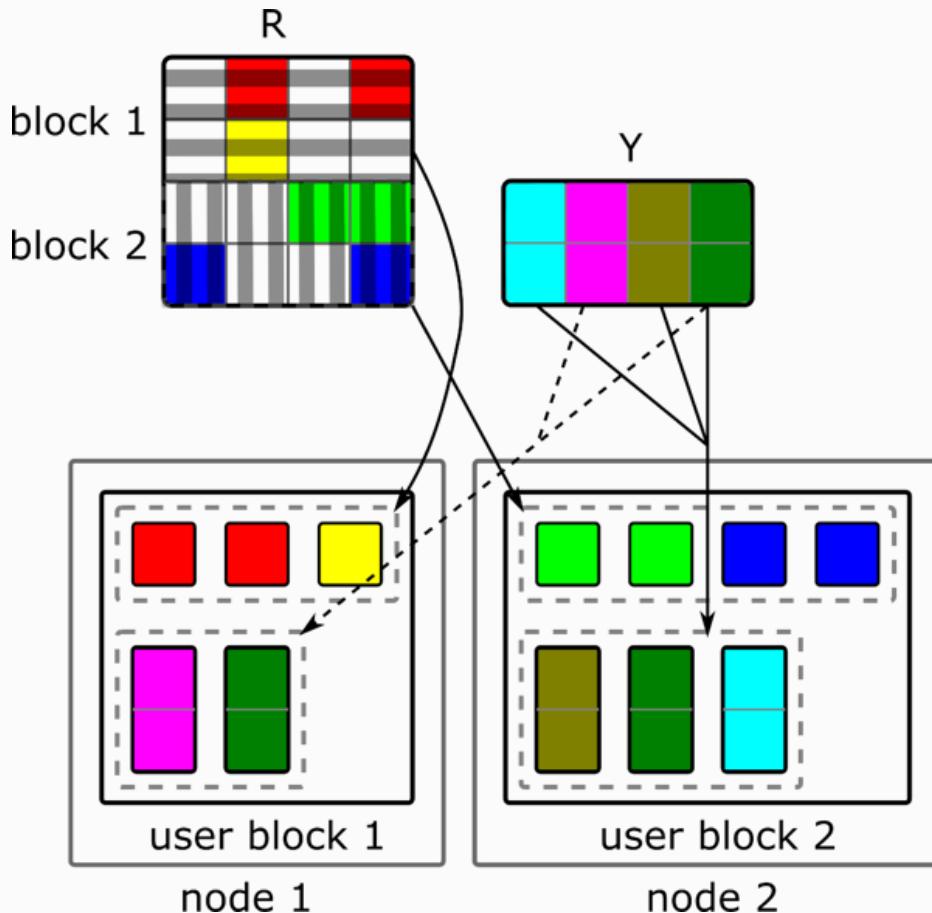
- Пользователи делятся на блоки, каждый блок обрабатывается на своей машине

# Блочный ALS (как в Spark ML)



- Пользователи делятся на блоки, каждый блок обрабатывается на своей машине
- Один раз вычисляем, какие товары понадобятся в блоках

# Блочный ALS (как в Spark ML)



- Пользователи делятся на блоки, каждый блок обрабатывается на своей машине
- Один раз вычисляем, какие товары понадобятся в блоках
- Пересылаются только нужные части матриц  
**в память машин**

# Блочный ALS (как в Spark ML)



Числом блоков можно управлять (параметр numBlocks)

# Блочный ALS (как в Spark ML)

- Числом блоков можно управлять (параметр numBlocks)
- Чем их больше, тем выше шанс, что подматрица товаров для каждого блока влезет в память

# Блочный ALS (как в Spark ML)

- Числом блоков можно управлять (параметр numBlocks)
- Чем их больше, тем выше шанс, что подматрица товаров для каждого блока влезет в память

```
from pyspark.ml.recommendation import ALS

als = ALS(maxIter=5, regParam=0.01,
userCol="userId", itemCol= "itemId",
ratingCol= "rating", numBlocks=10)

model = als.fit(training)
```

# Резюме



ALS лучше параллелится, потому что каждый шаг распадается на много независимых подзадач

# Резюме



ALS лучше параллелится, потому что каждый шаг распадается на много независимых подзадач



iALS так же быстро работает с неявными оценками (нужно только заранее посчитать  $Y^T Y$ )

# Резюме



ALS лучше параллелится, потому что каждый шаг распадается на много независимых подзадач



iALS так же быстро работает с неявными оценками (нужно только заранее посчитать  $Y^T Y$ )



В Spark ML реализован блочный ALS/iALS

# Резюме



ALS лучше параллелится, потому что каждый шаг распадается на много независимых подзадач



iALS так же быстро работает с неявными оценками (нужно только заранее посчитать  $Y^T Y$ )



В Spark ML реализован блочный ALS/iALS



Далее: поговорим о том, как оценить качество рекомендательной системы

# **Метрики качества**



# Какая метрика лучше?

$$MAE = \frac{1}{|R|} \sum_{(u,i) \in R} |r_{ui} - \hat{r}_{ui}|$$

$$RMSE = \sqrt{\frac{1}{|R|} \sum_{(u,i) \in R} (r_{ui} - \hat{r}_{ui})^2}$$

					MAE	RMSE
true	5	4	3	5		
A	5	2	1	3	1.5	1.73
B	5	4	3	1	1	2

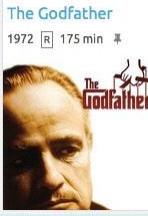
# Какая метрика лучше?

→ Вообще говоря, вряд ли нам нужно прям точно угадать рейтинг, нужно отсортировать правильно!

					MAE	RMSE
true	5	4	3	5		
A	5	2	1	3	1.5	1.73
B	5	4	3	1	1	2

# Какая метрика лучше?

→ Вообще говоря, вряд ли нам нужно прям точно угадать рейтинг, нужно отсортировать правильно!

					MAE	RMSE
true	5	4	3	5		
A	5	2	1	3	1.5	1.73
B	5	4	3	1	1	2

→ Вариант А сортирует лучше!

# Метрики ранжирования

→ Point-wise (поточечные):

$$f(user, item) \rightarrow \mathbb{R} \quad L(y_{i,j}, f_{i,j}) = \frac{1}{2}(y_{i,j} - f_{i,j})^2$$

# Метрики ранжирования

→ Point-wise (поточечные):

$$f(user, item) \rightarrow \mathbb{R} \quad L(y_{i,j}, f_{i,j}) = \frac{1}{2}(y_{i,j} - f_{i,j})^2$$

→ Pair-wise (попарные):

$$f(user, item_1, item_2) \rightarrow \mathbb{R} \quad \sum_{y_i > y_j} f_i \leq f_j$$

# Метрики ранжирования

→ Point-wise (поточечные):

$$f(user, item) \rightarrow \mathbb{R}$$

$$L(y_{i,j}, f_{i,j}) = \frac{1}{2}(y_{i,j} - f_{i,j})^2$$

→ Pair-wise (парные):

$$f(user, item_1, item_2) \rightarrow \mathbb{R}$$

$$\sum_{y_i > y_j} f_i \leq f_j$$

→ List-wise (списковые):

$$f(user, item_1, \dots, item_n) \rightarrow \mathbb{R}$$

NDCG

# Discounted Cumulative Gain (DCG)

$$DCG = \sum_{i=1}^k \frac{2^{rel_i} - 1}{\log_2(i + 1)}$$

Идеальная сортировка:

5	4	3	2	1	DCG
$31/\log_2(2)$	$15/\log_2(3)$	$7/\log_2(4)$	$3/\log_2(5)$	$1/\log_2(6)$	45.64

Ошибка в начале:

3	4	5	2	1	DCG
$7/\log_2(2)$	$15/\log_2(3)$	$31/\log_2(4)$	$3/\log_2(5)$	$1/\log_2(6)$	33.64

Ошибка в конце:

5	4	1	2	3	DCG
$31/\log_2(2)$	$15/\log_2(3)$	$1/\log_2(4)$	$3/\log_2(5)$	$7/\log_2(6)$	44.96

# Normalized Cumulative Gain (NDCG)

$$NDCG = \frac{DCG}{Ideal\ DCG}$$

Идеальная сортировка:

5	4	3	2	1	NDCG
$31/\log_2(2)$	$15/\log_2(3)$	$7/\log_2(4)$	$3/\log_2(5)$	$1/\log_2(6)$	1.00

Ошибка в начале:

3	4	5	2	1	NDCG
$7/\log_2(2)$	$15/\log_2(3)$	$31/\log_2(4)$	$3/\log_2(5)$	$1/\log_2(6)$	0.73

Ошибка в конце:

5	4	1	2	3	NDCG
$31/\log_2(2)$	$15/\log_2(3)$	$1/\log_2(4)$	$3/\log_2(5)$	$7/\log_2(6)$	0.98

# Пример

					MAE	RMSE	NDCG
true	5	4	3	5			
A	5	2	1	3	1.5	1.73	1.00
B	5	4	3	1	1	2	0.93

Вариант А сортирует лучше!

# Резюме



3 основных типа метрик: поточечные, попарные,  
списковые

# Резюме



3 основных типа метрик: поточечные, попарные, списковые



Нужно искать такую, которая будет лучше всего коррелировать с онлайн качеством (количество покупок, время смотрения)