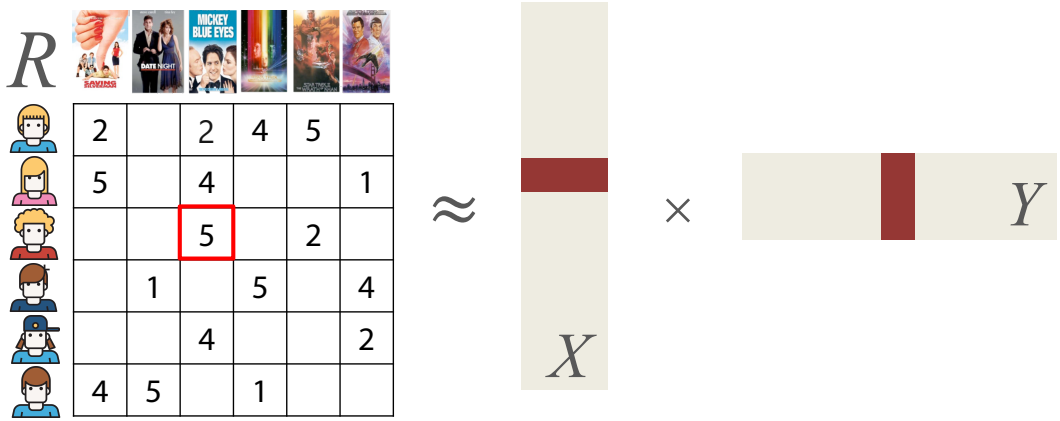
Week 4. Recommender Systems

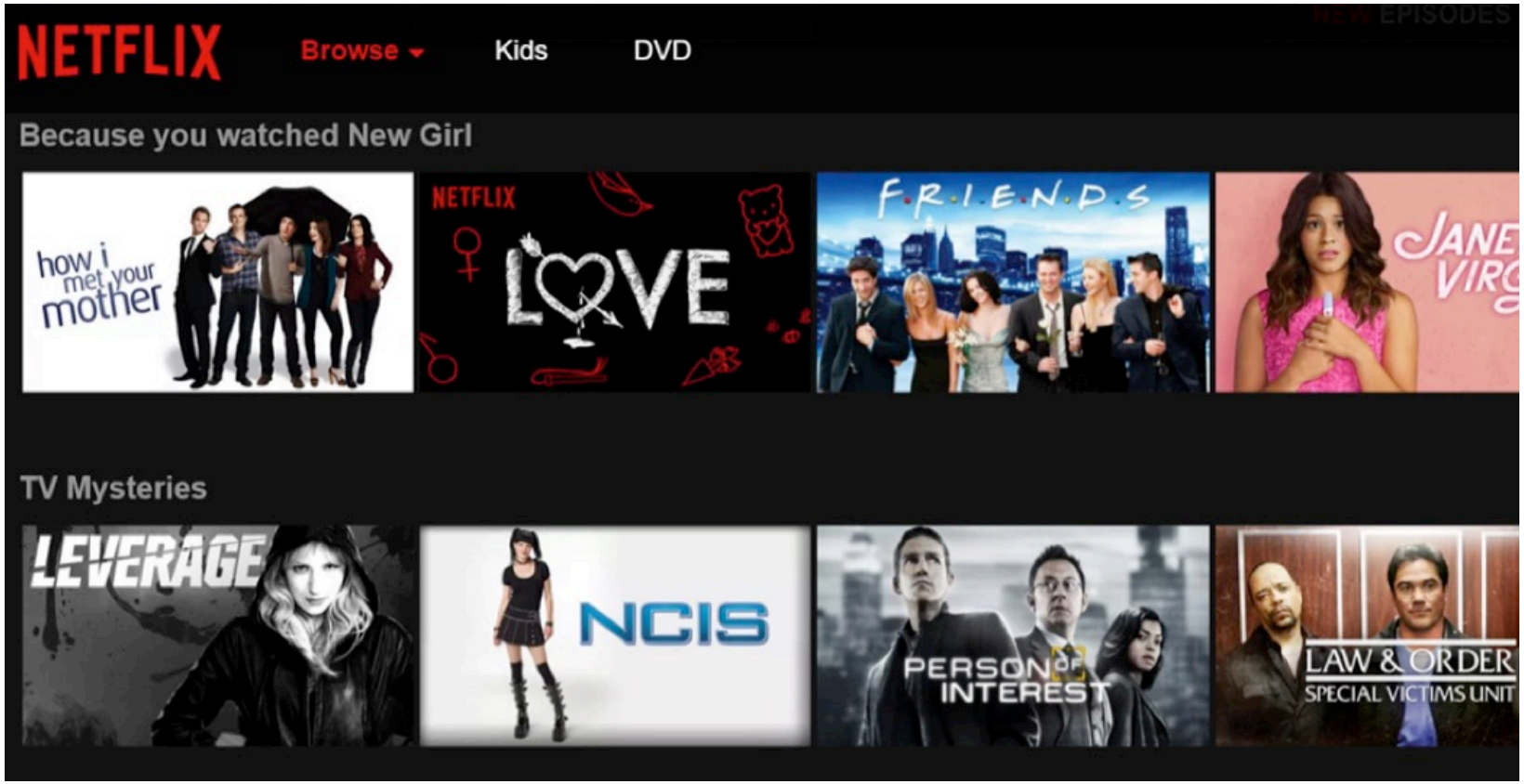
Recommender Systems

- **Memory-based**: item-item and user-user similarities

- **Model-based**: latent factors representation of interactions

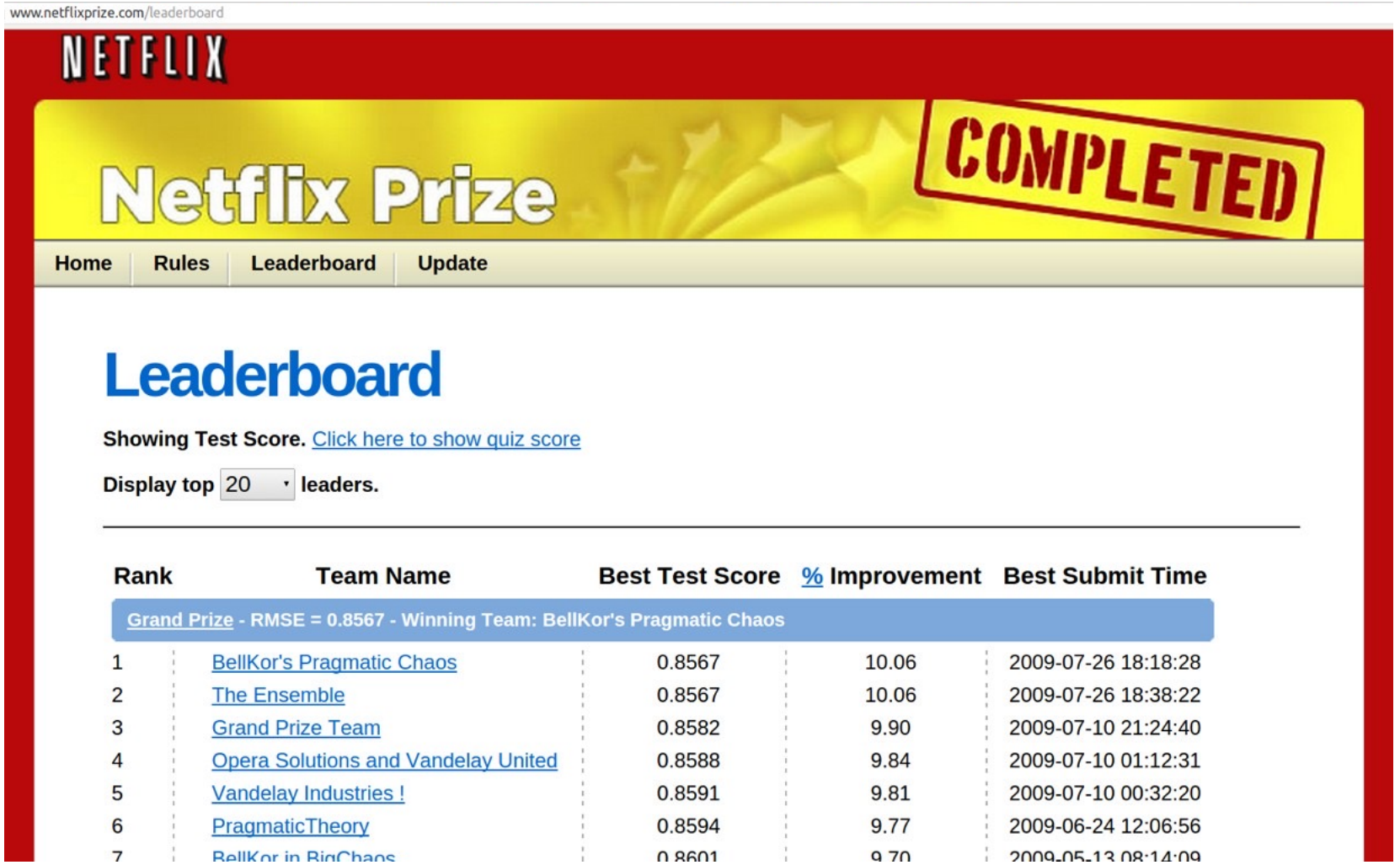


Netflix



65% of total consumption is attributed to recommended items

$1,000,000 for 10% improvement



www.netflixprize.com/leaderboard

NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update

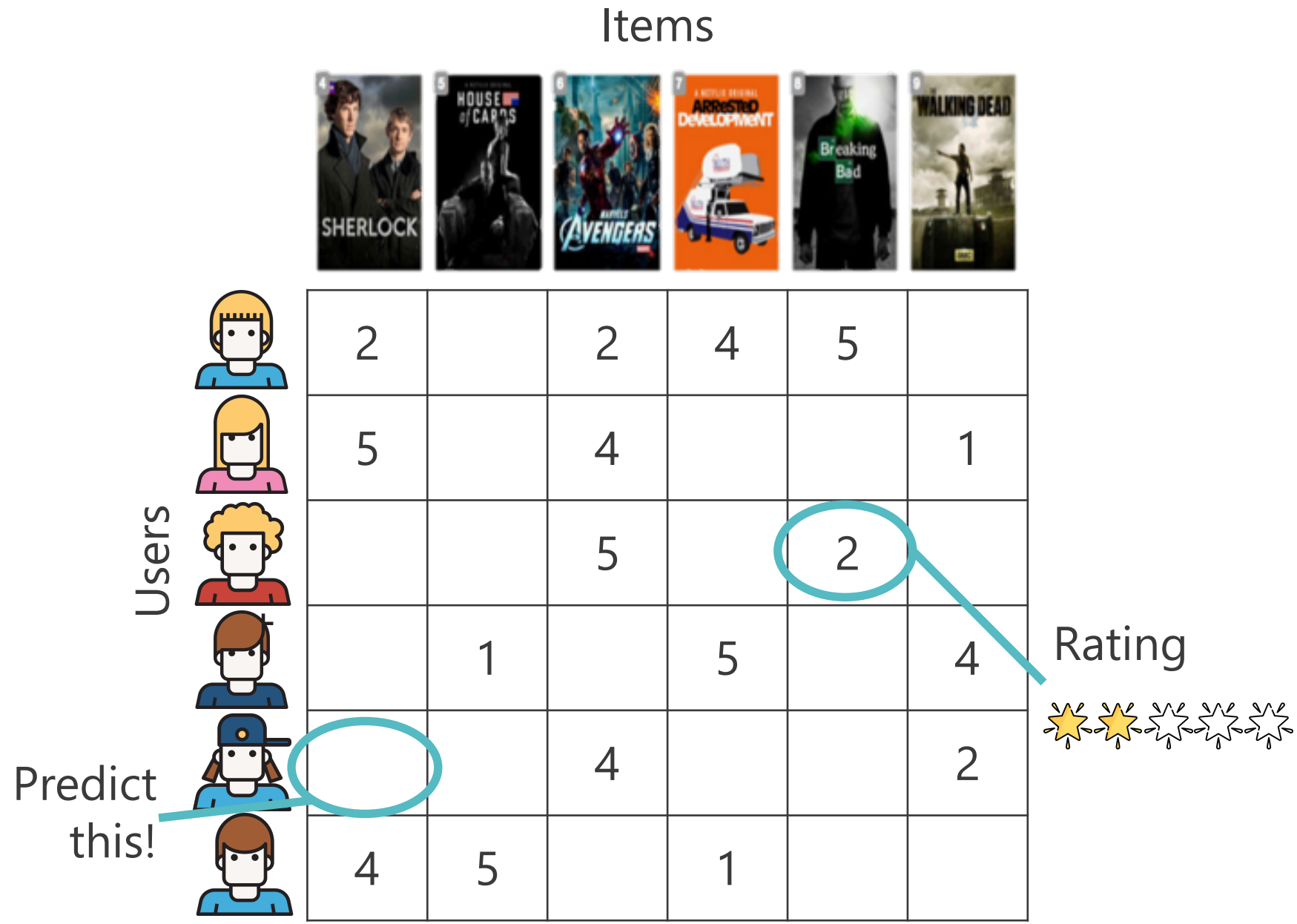
Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 **leaders.**

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

User-item matrix



Two types of ratings

It's ok

I love it

I like it

I hate it!

I don't like it

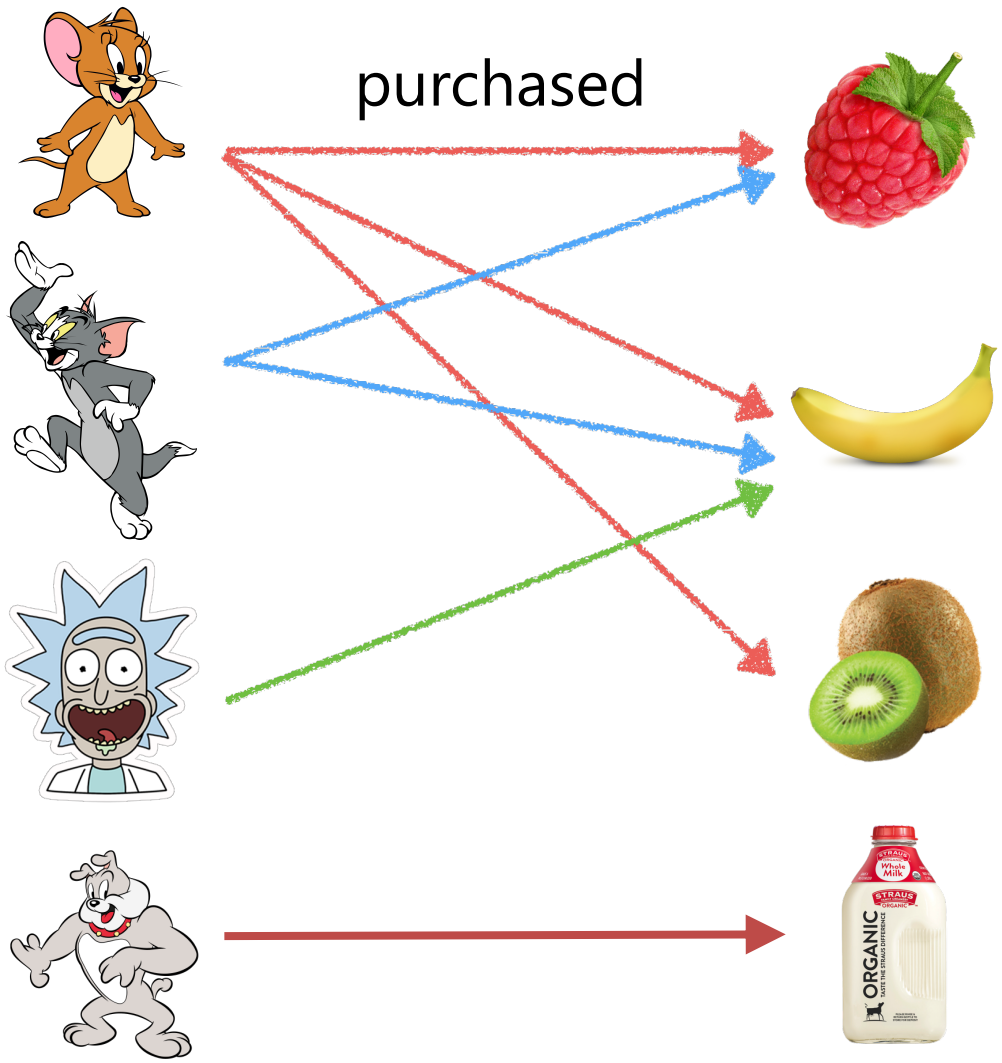
Explicit
(number of stars)

Implicit
(watched a movie)

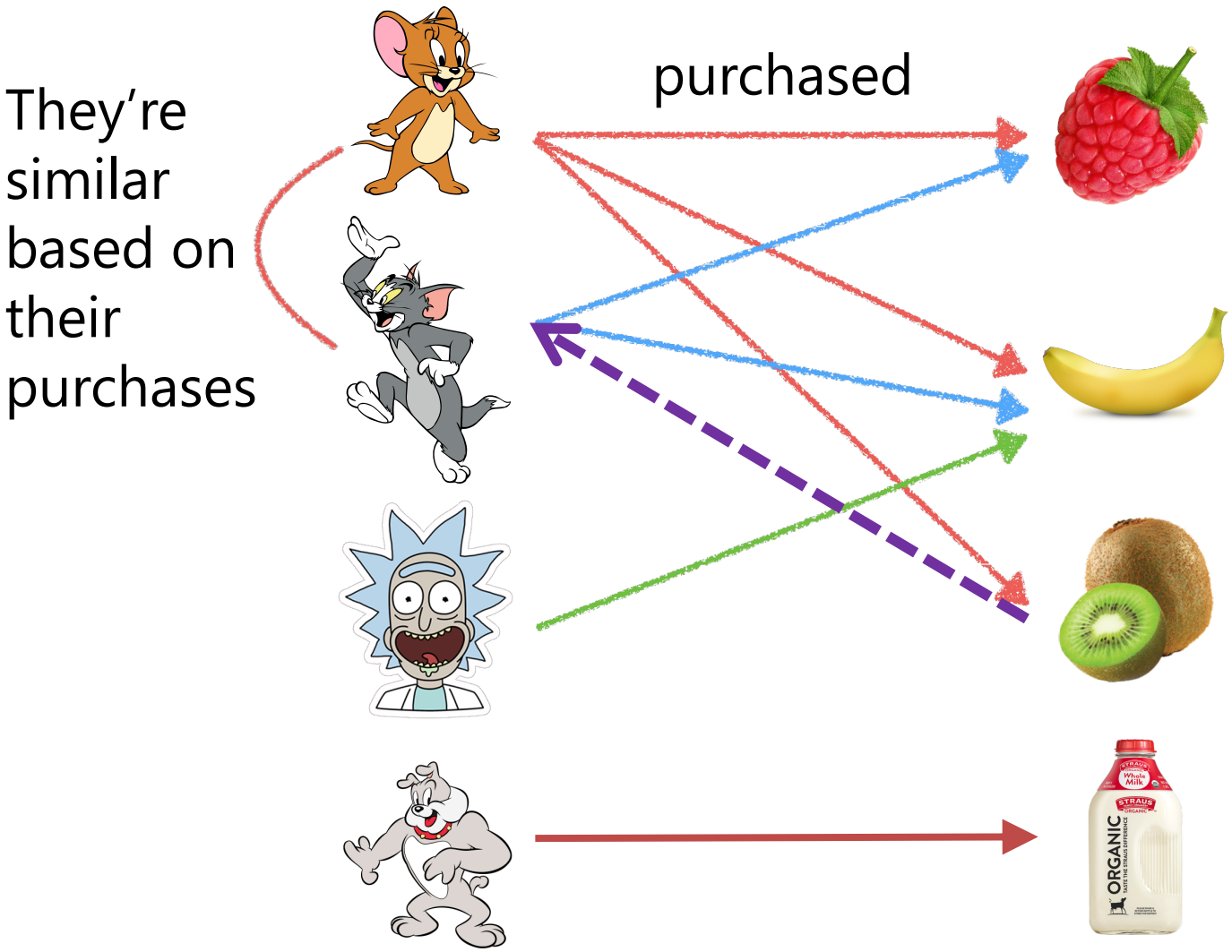
Memory-based collaborative filtering

Collaborative filtering

Can you produce recommendations for Tom?

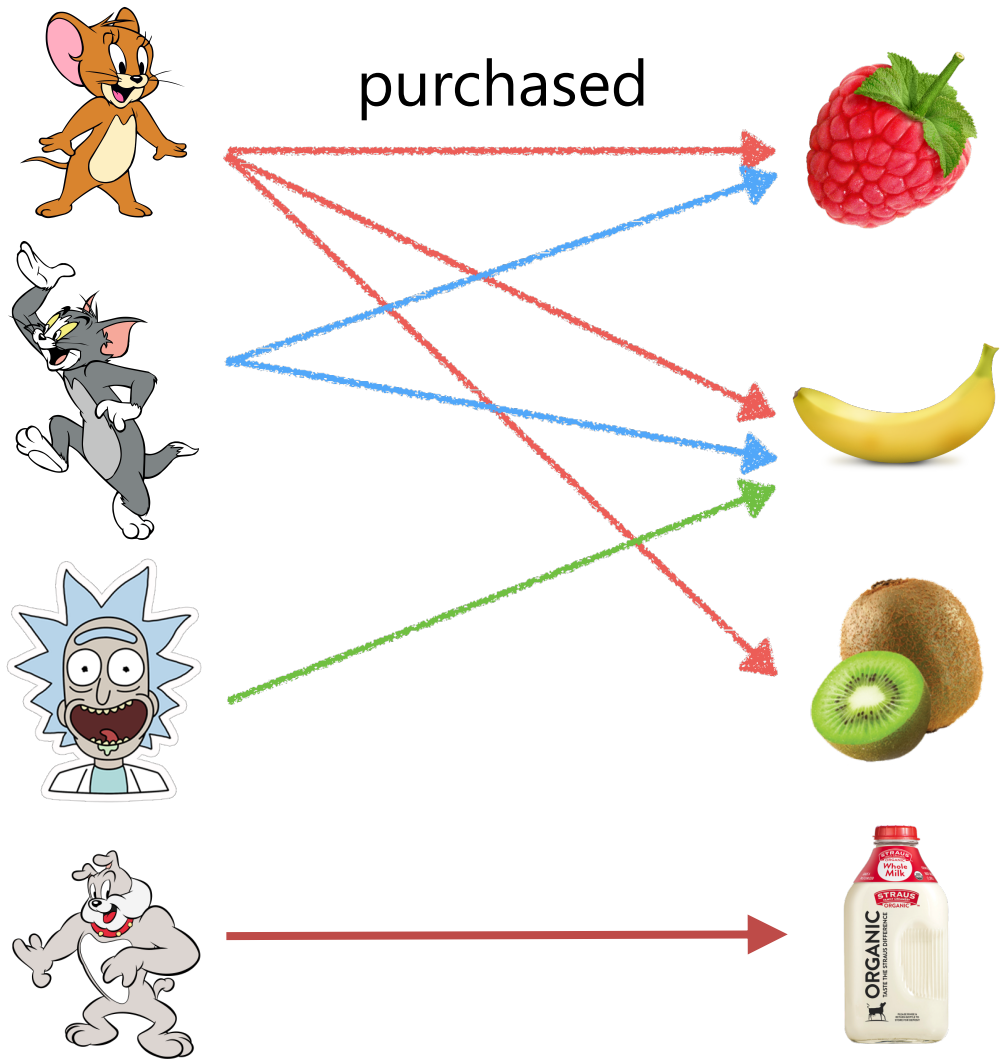


User-based collaborative filtering

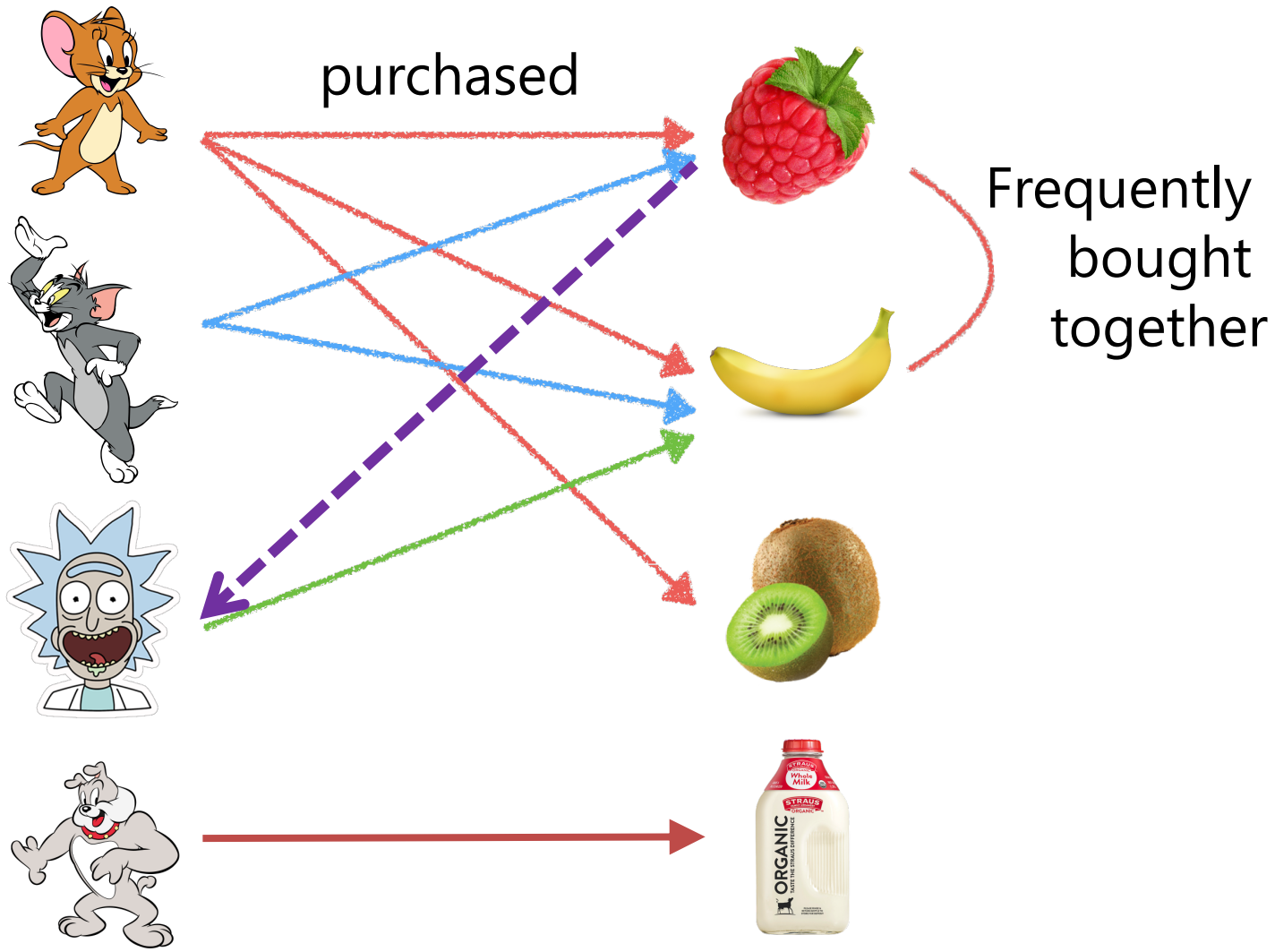


Collaborative filtering

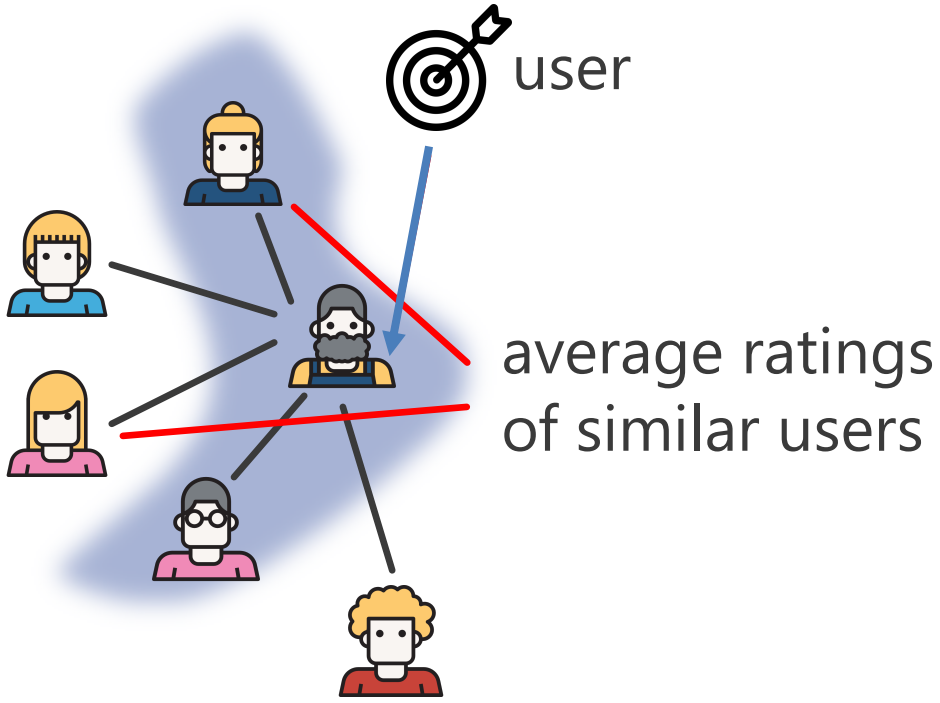
Can you produce recommendations for Rick?



Item-based collaborative filtering

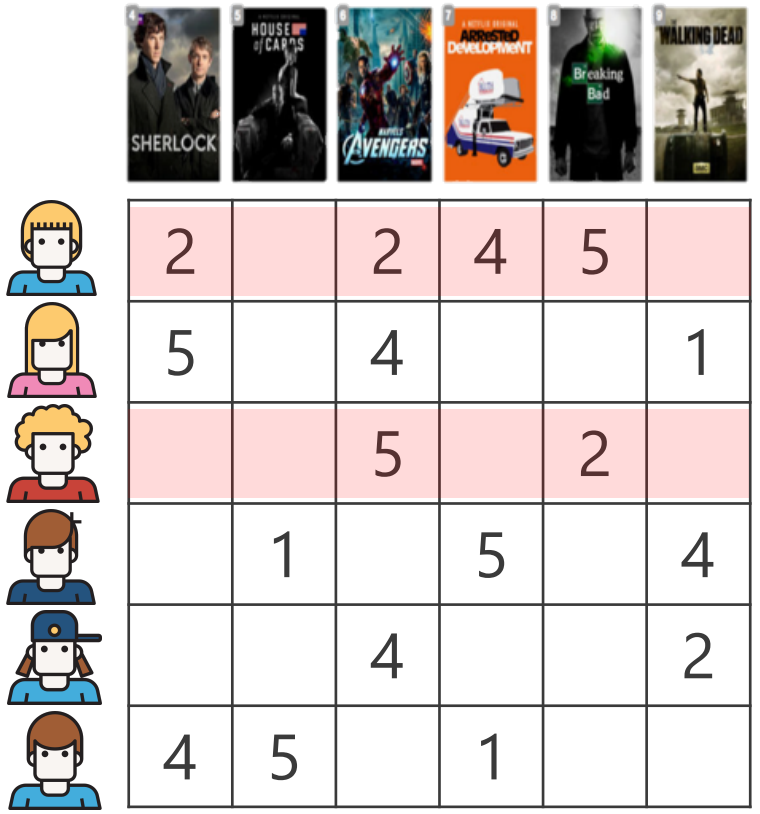


User-based & item-based



User-based collaborative filtering

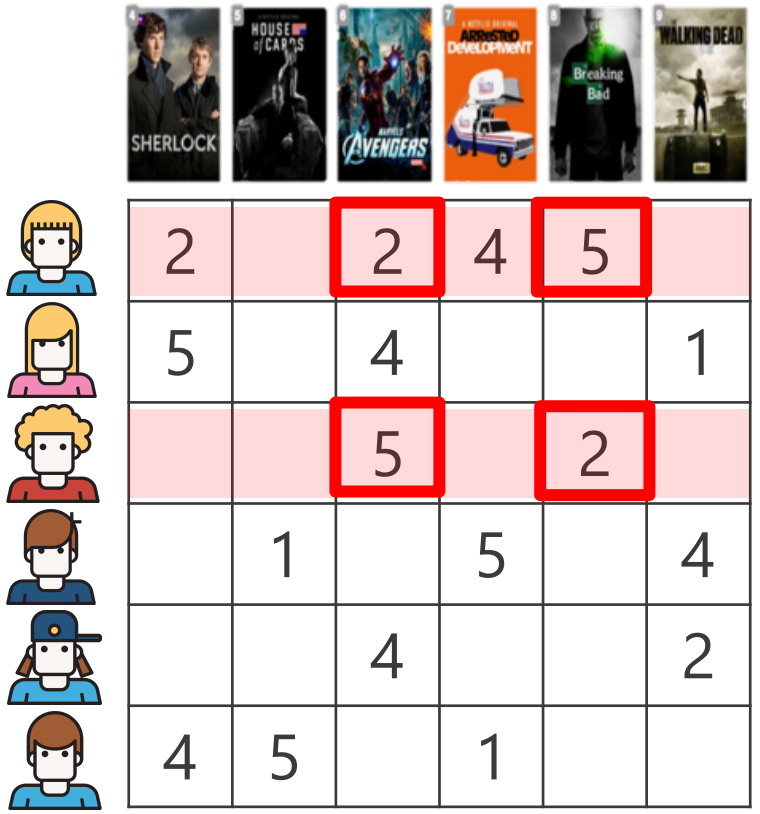
Similarity of users with explicit ratings



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

Are they similar?

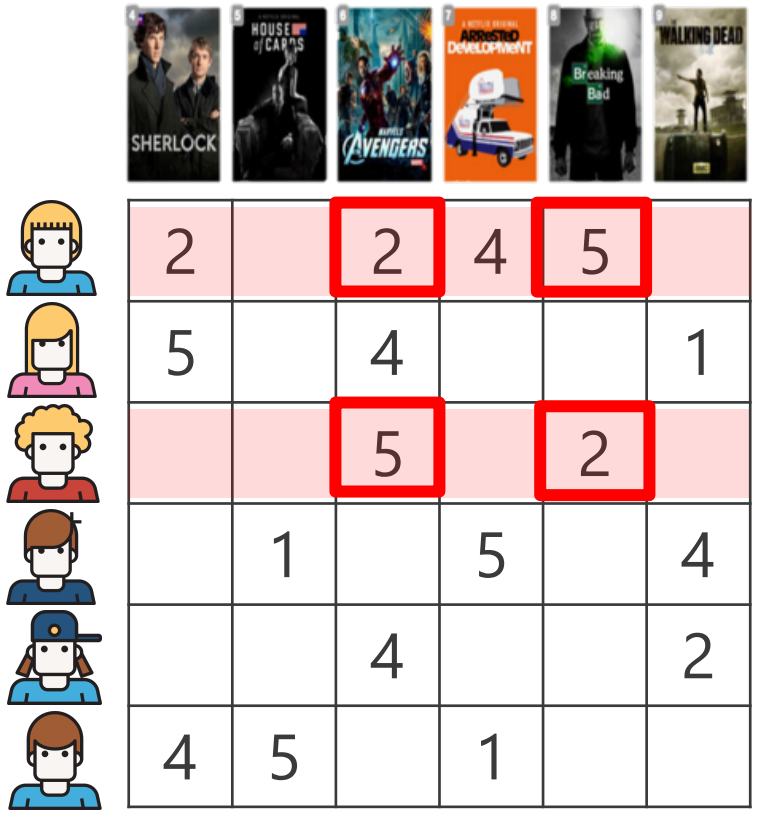
Similarity of users with explicit ratings



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

Let's look at common items.
Is there a correlation?

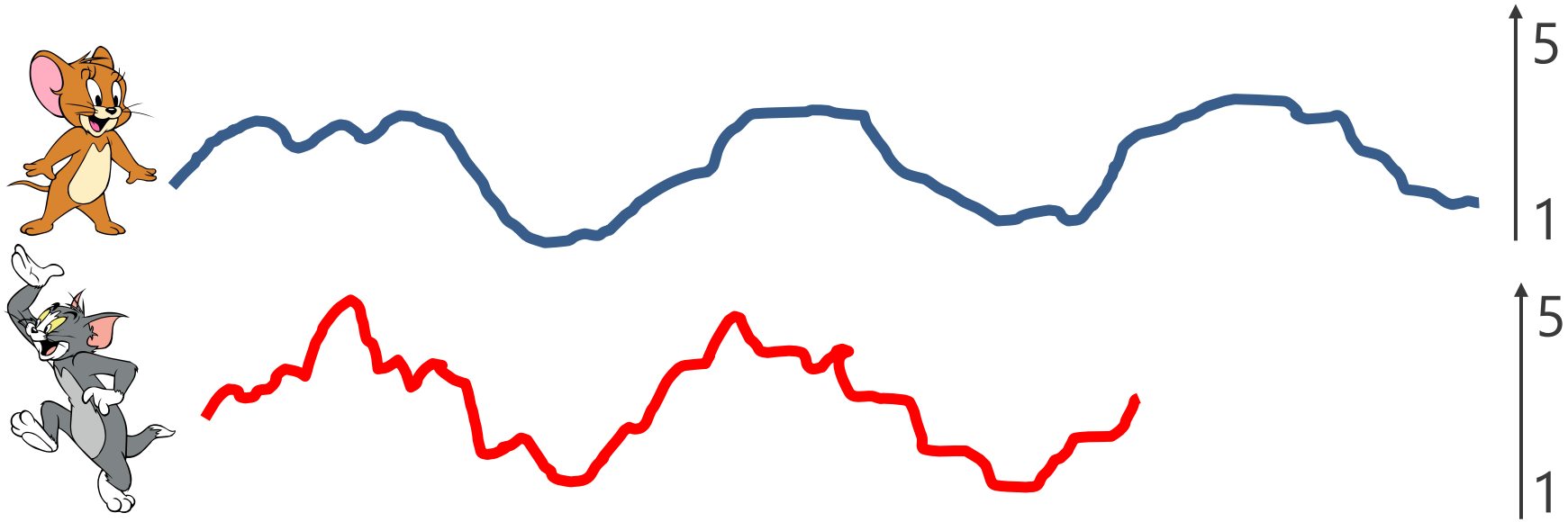
Similarity of users with explicit ratings



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

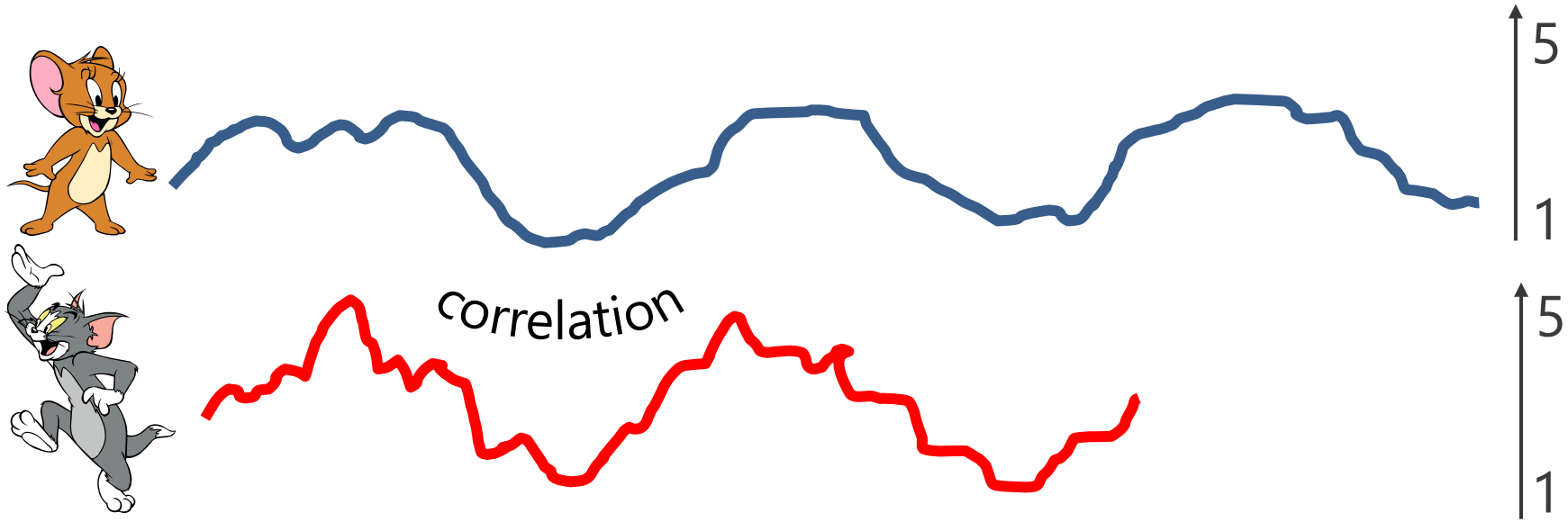
Let's look at common items.
Is there a correlation? **Yes, negative.**

Similarity of users with explicit ratings



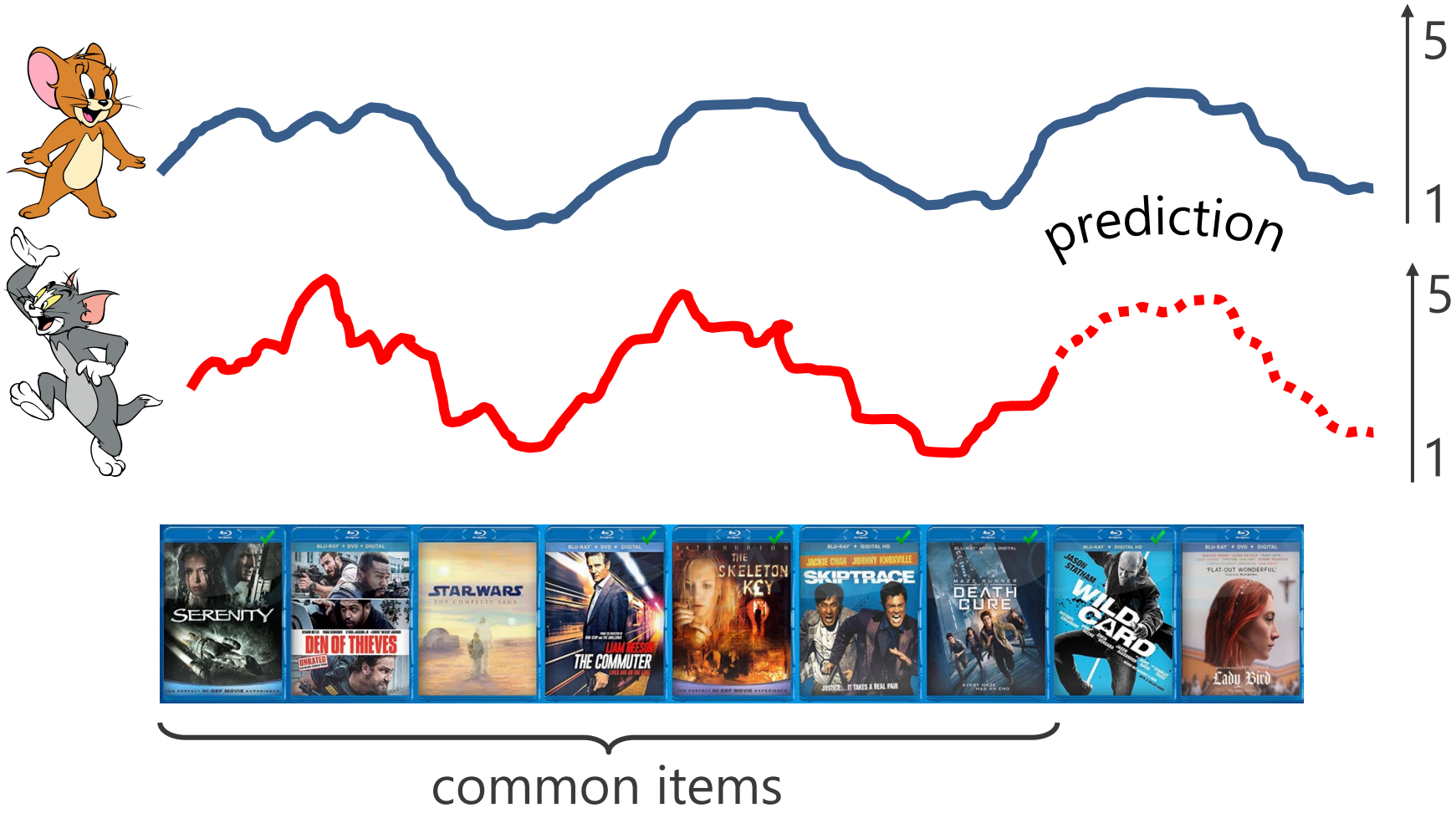
common items

Similarity of users with explicit ratings

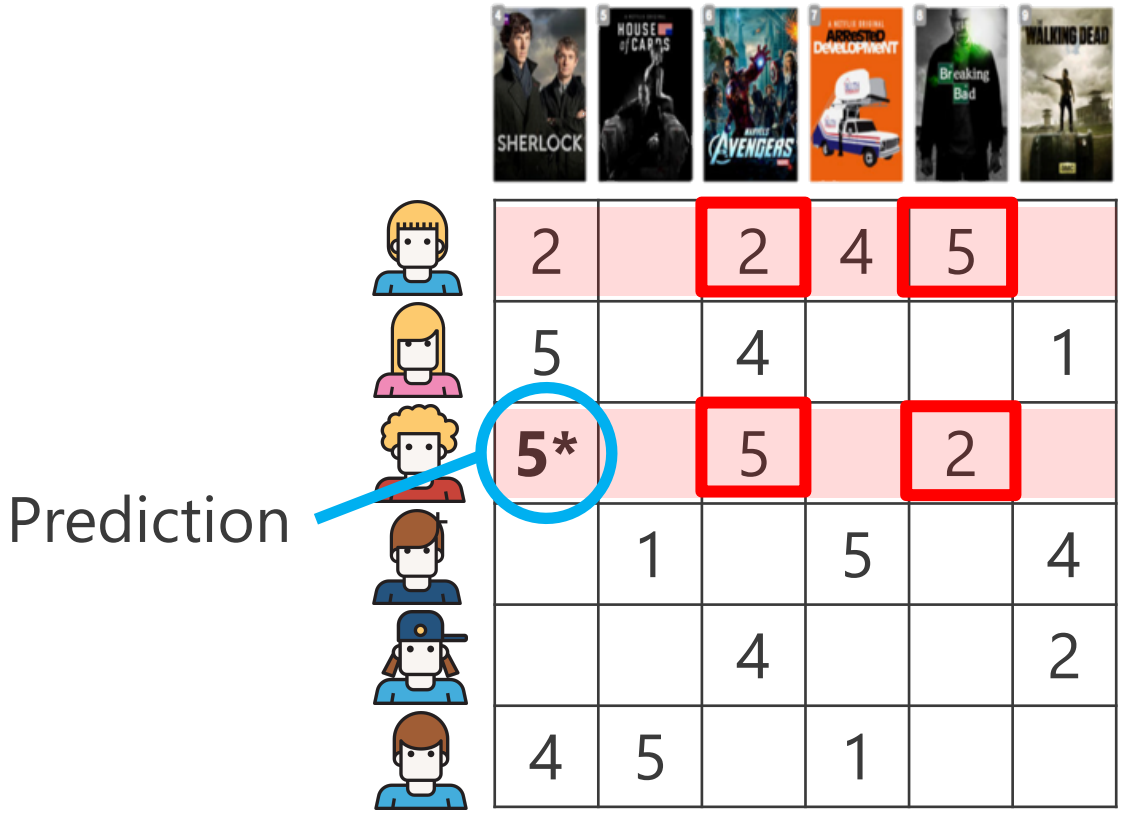


common items

Similarity of users with explicit ratings



Similarity of users with explicit ratings



User-based collaborative filtering

r_{ui} – user-item rating

r_u – average user rating

Similarity of two users (I – common items):

$$corr(u,v) = \frac{\sum_{i \in I}(r_{ui} - r_u)(r_{vi} - r_v)}{\sqrt{\sum_{i \in I}(r_{ui} - r_u)^2 \sum_{i \in I}(r_{vi} - r_v)^2}}$$

Prediction ($N(u)$ – neighboring users):

$$\widehat{r_{ui}} = r_u + \frac{\sum_{v \in N(u)} corr(u,v)(r_{vi} - r_v)}{\sum_{v \in N(u)} |corr(u,v)|}$$

User-based vs item-based

- 10,000 ratings
- 1000 users
- 100 items
- Ratings are uniformly distributed in the matrix

How many items do 2 random users have in common on average?

What about 2 random items?

User-based vs item-based

- 10,000 ratings
- 1000 users
- 100 items
- Ratings are uniformly distributed in the matrix

How many items do 2 random users have in common on average? (1)

What about 2 random items? (10)

Item-based CF has better chances at computing confident similarity estimates!

Item-based collaborative filtering

Similarity of items with implicit feedback

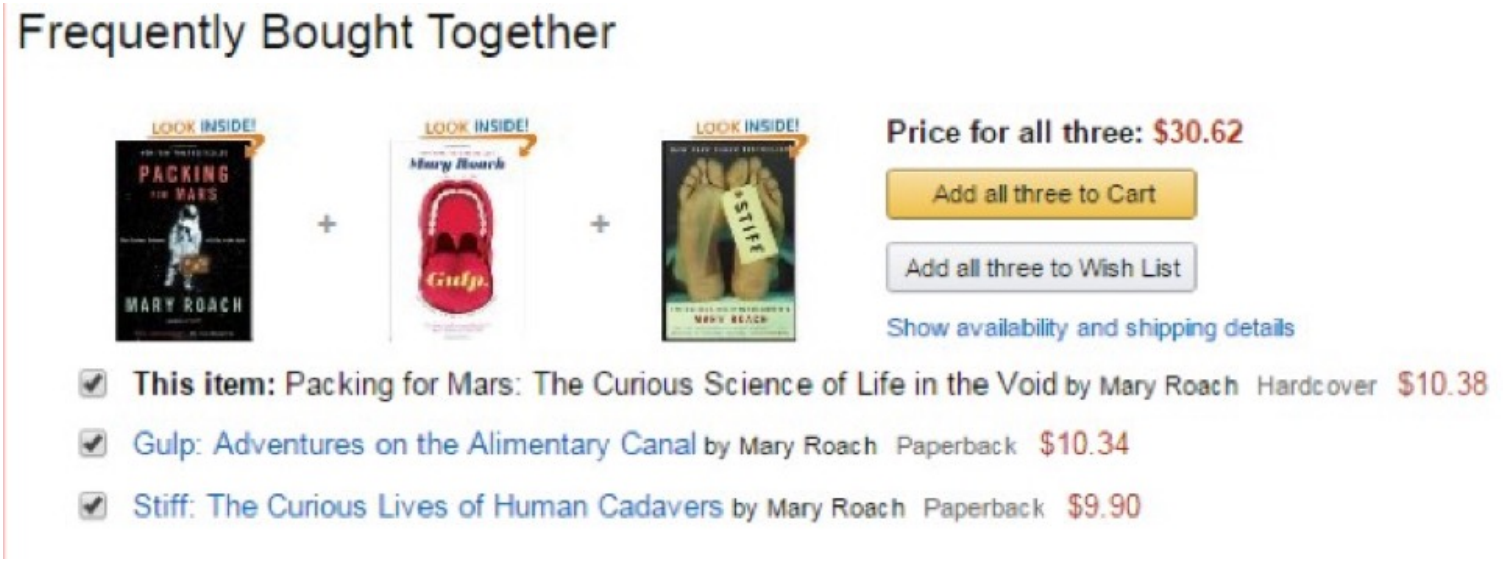
We don’t ask the user to provide a rating, but just observe what the user clicks, watches (a movie) or buys (a product).

If you don’t click, watch or buy an item, it doesn’t mean you don’t like it. That’s why it is an implicit feedback.

Similarity of items with implicit feedback

We don't ask the user to provide a rating, but just observe what the user clicks, watches (a movie) or buys (a product).

If you don't click, watch or buy an item, it doesn't mean you don't like it. That's why it is an implicit feedback.



How to calculate this one?

Frequently bought together

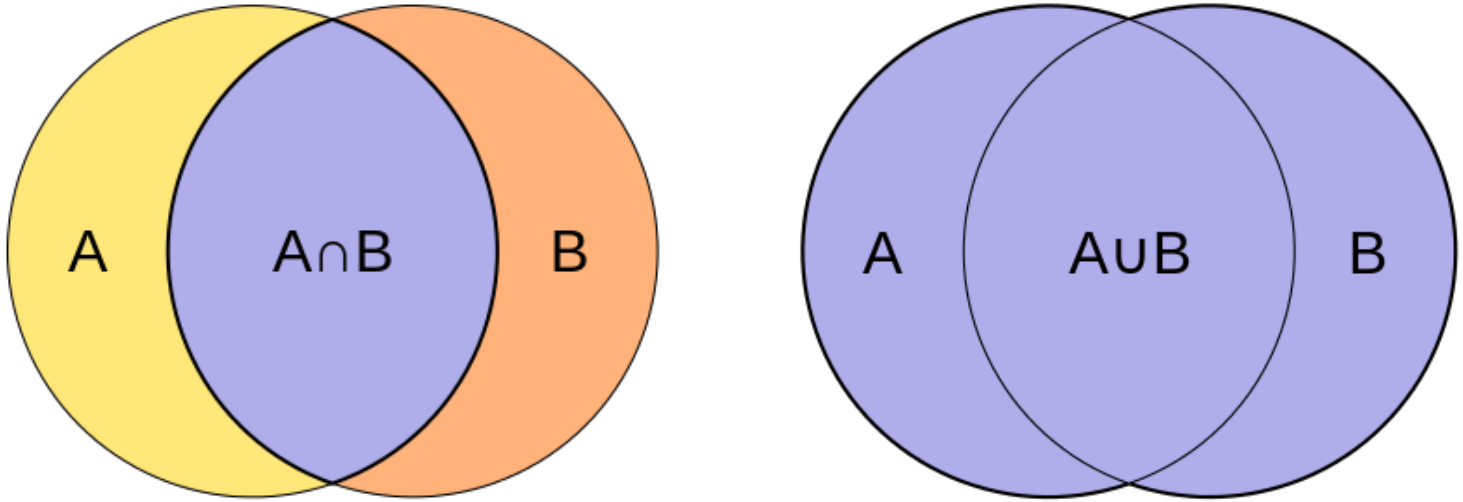
Naïve co-occurrences are bad:

- Best sellers like Harry Potter will co-occur with many other books (many people buy them both)
- It doesn't mean it's similar to every book, right?

Jaccard similarity of sets (*of users*)

Takes into account the popularity of items:

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



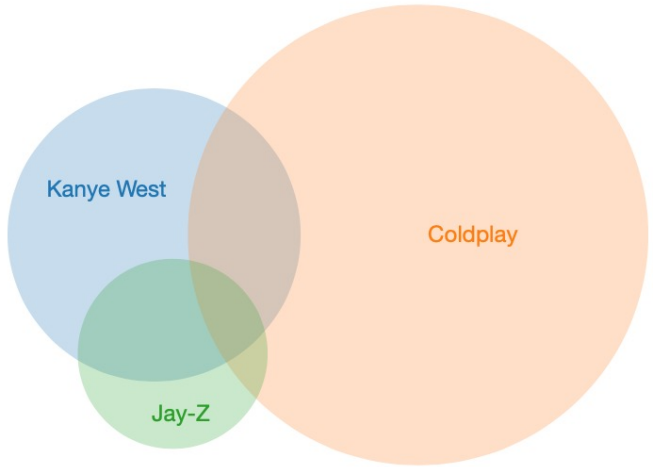
A – users who've bought item A

Example for user-artist interactions

Most similar to Kanye West by co-occurrence:

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

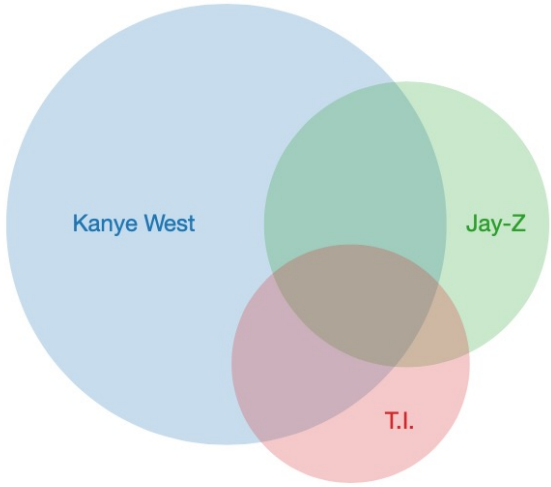
more ...



Most similar to Kanye West by Jaccard similarity:

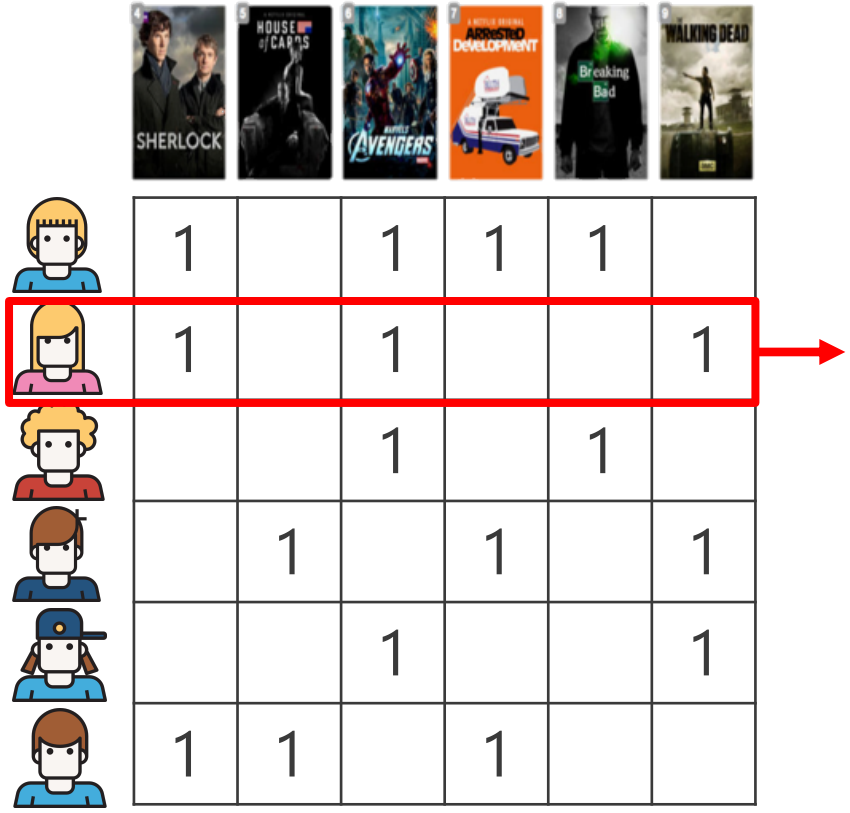
Artist	Jaccard
Jay-Z	0.217
T.I.	0.163
Lupe Fiasco	0.156
Nas	0.156
Common	0.139

more ...



https://www.benfrederickson.com/distance-metrics/

How to compute all item-item similarities



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

Map step:

(1, 3) → 1
(1, 6) → 1
(3, 6) → 1

Reduce step:
Sum up the values for each key (i, j)

Item-based collaborative filtering at scale

- More confident estimates of similarities.
  They don't change that fast.

- Viable solution (Amazon, 2003):
  - Precompute item-item similarities in offline
    (store in No-SQL)

  - Update user history with new clicks (store in No-SQL)

  - Find similar items in real-time based on his clicks
    (query No-SQL with clicked items)

https://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf

Summary

- Memory based collaborative filtering works well when you have lots of feedback (confident similarities).
- And it falls short when the feedback is very sparse.
- Item-based collaborative filtering is easier to implement in production environment & it works better
- Two users must rate *identical* items to compute user-user similarity, “Terminator” and “Terminator 2” are different items and won’t affect similarity.

Model-based collaborative filtering (matrix factorization)

Dimensionality reduction

- User-item matrix is **sparse** (many missing values)
- Item ratings are **correlated** (user likes similar items)
- **Hence matrix can be compressed!**

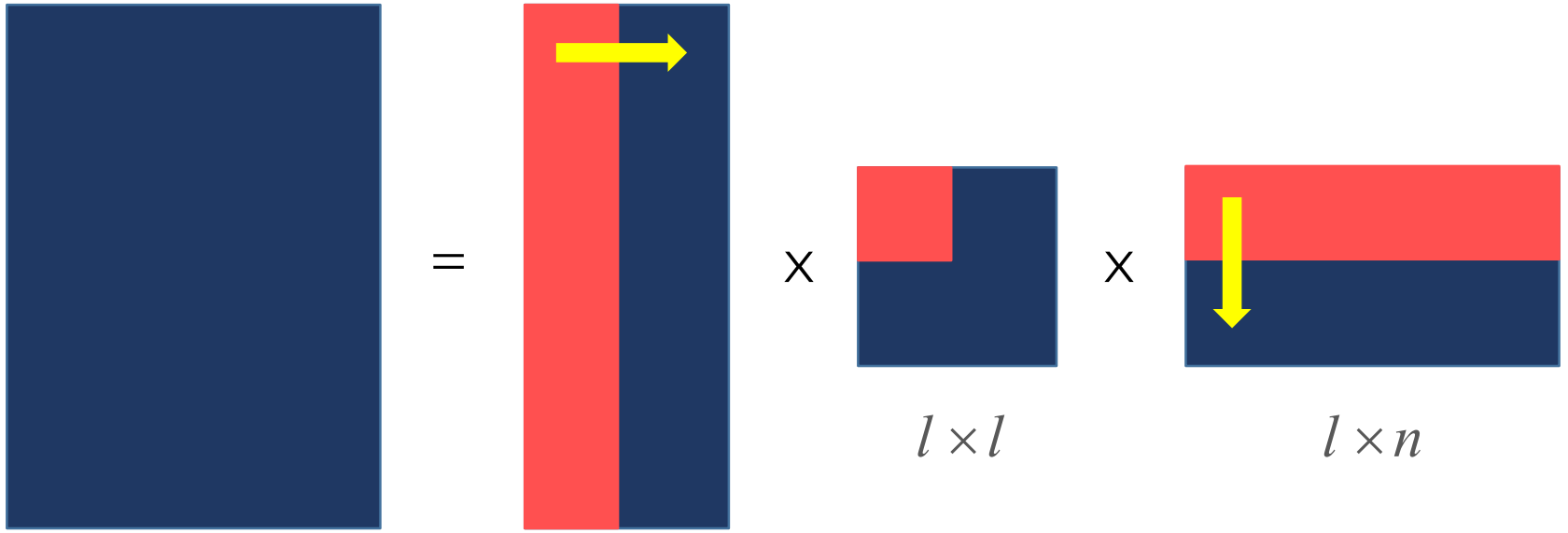
2		2	4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

SVD for dense matrix

$$R = U\Sigma V^{\mathrm{T}}$$

$$UU^T = U^TU = I$$ ← identity matrix

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



$m \times n$ $\quad$ $m \times l$ ← number of latent factors

$l \times l$ $\quad$ $l \times n$

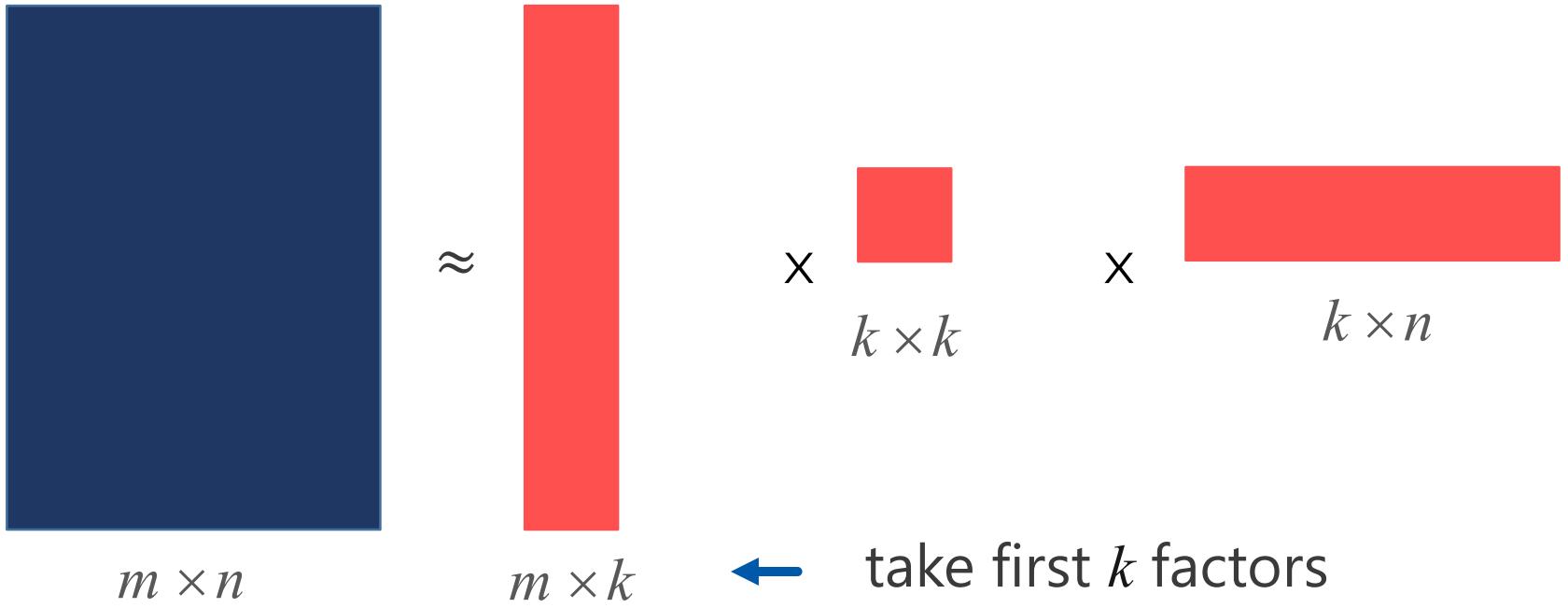
Factors are sorted based on their importance
in matrix reconstruction

SVD for dense matrix

$$R = U\Sigma V^{\mathrm{T}}$$

$$UU^T = U^TU = I$$ ← identity matrix

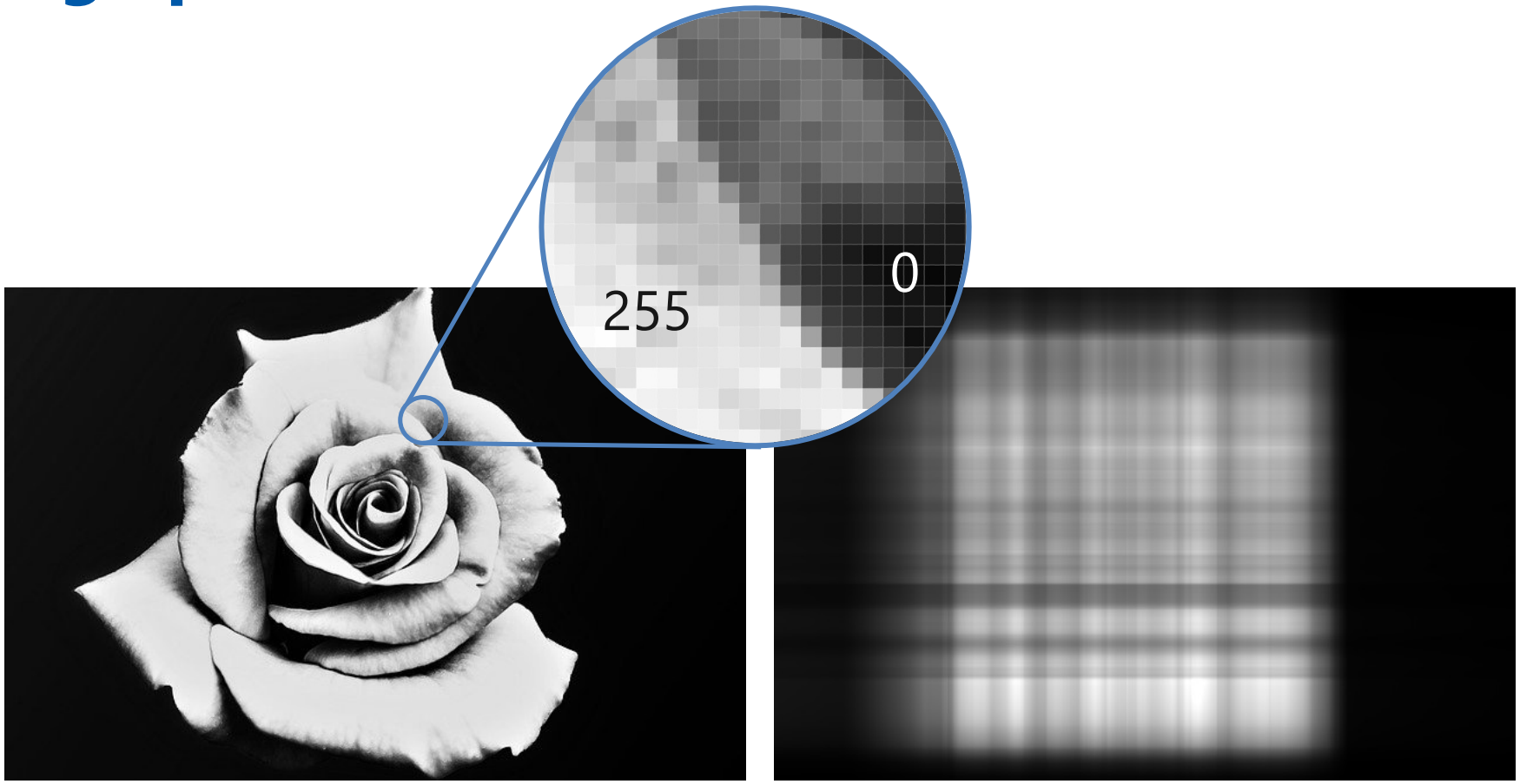
$$VV^T = V^TV = I$$



← take first k factors

sklearn.decomposition.TruncatedSVD

SVD for image pixels

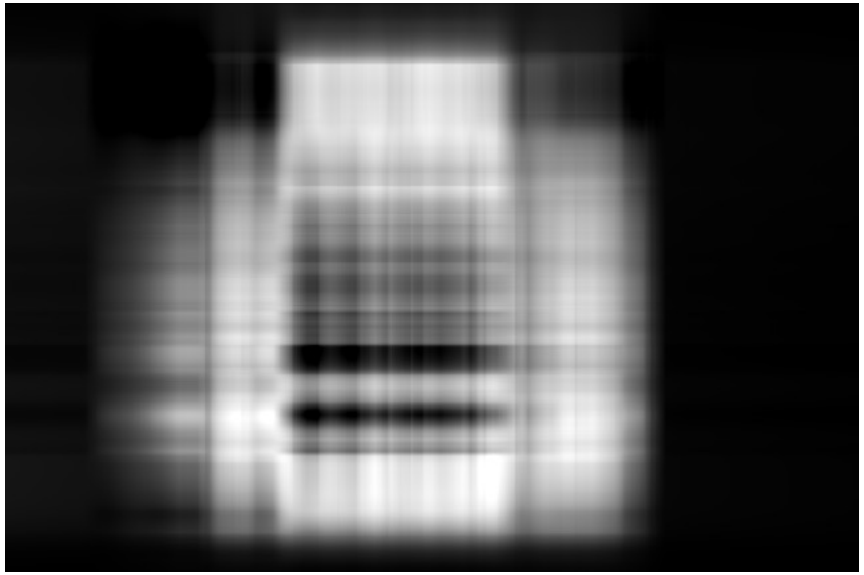


400 x 600 image
240000 values

400 x 1 / 1 x 1 / 1 x 600 matrices
1001 values

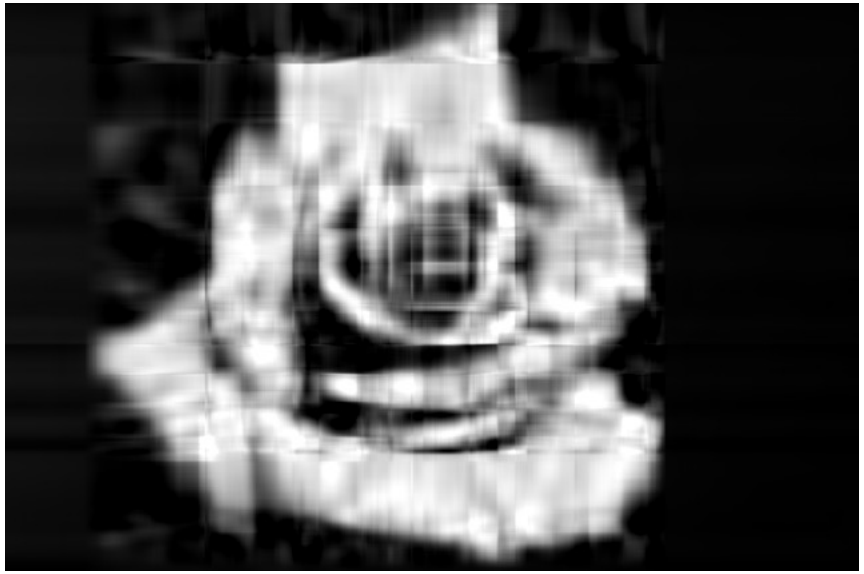
SVD for image pixels

400 x 600 image
240000 values



400 x 2 / 2 x 2 / 2 x 600 matrices
2004 values

SVD for image pixels

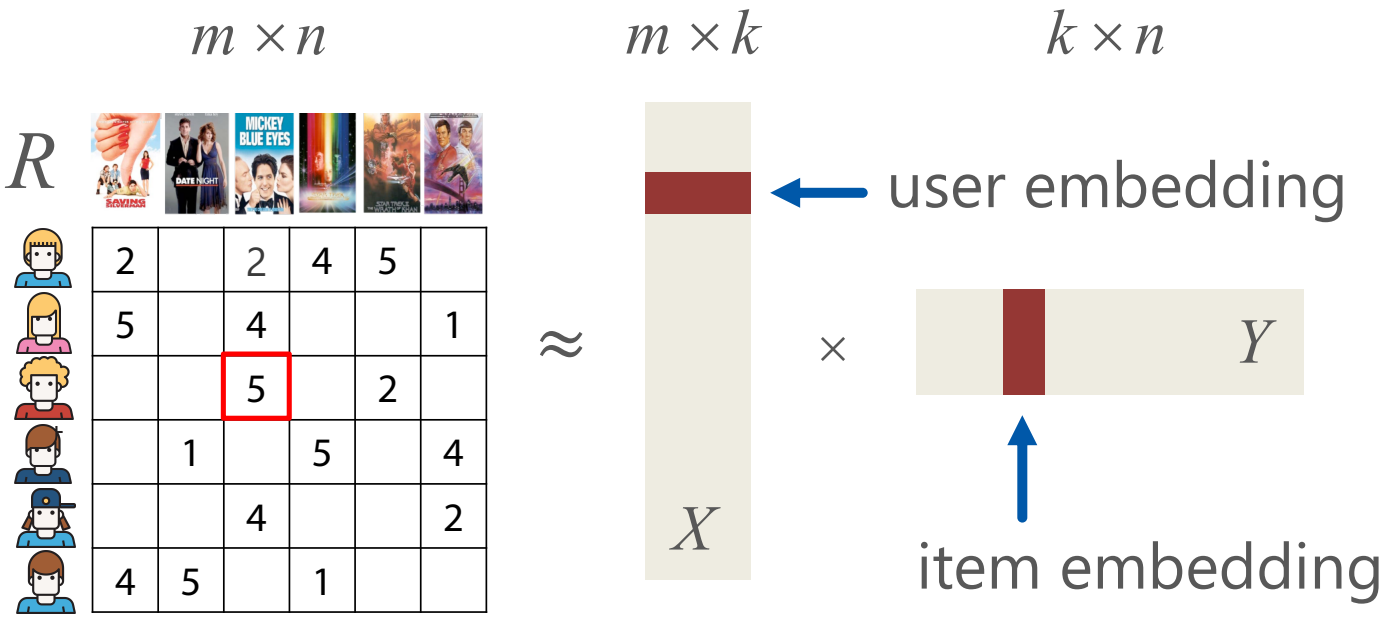


400 x 600 image
240000 values

400 x 9 / 9 x 9 / 9 x 600 matrices
9081 values

26x compression!

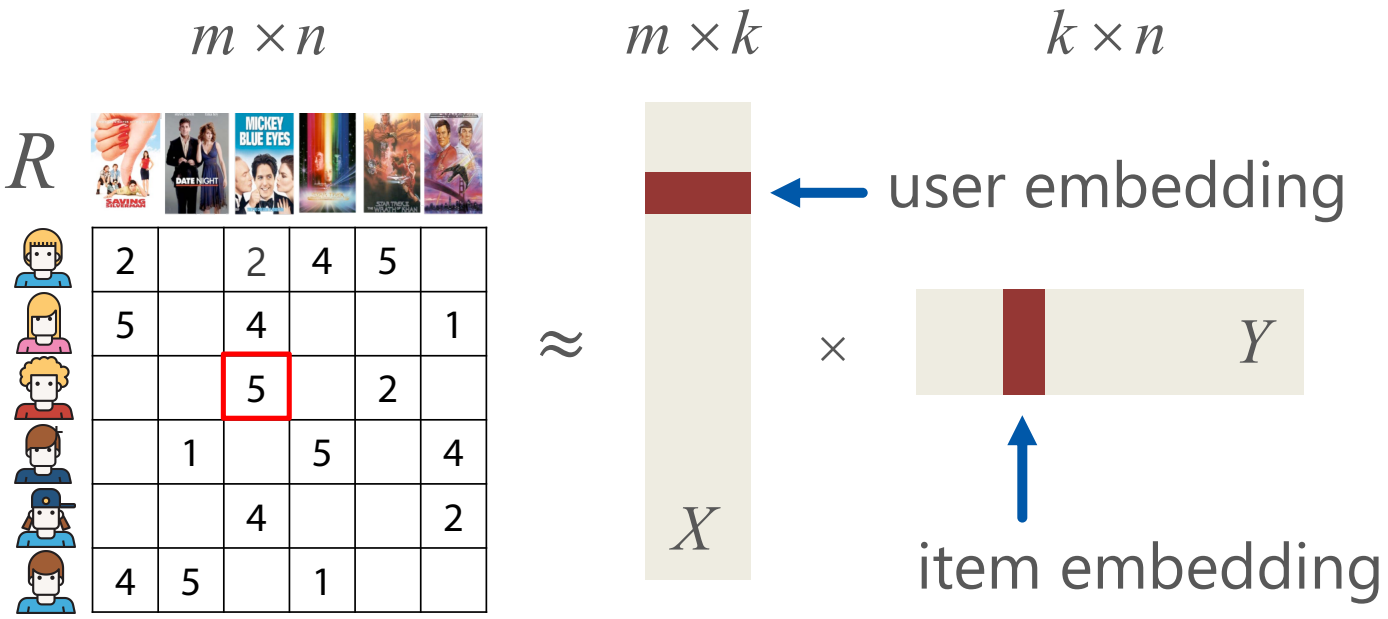
Funk SVD for sparse matrix



- Users and items are embedded into space of latent characteristics $\mathbb{R}^k$.

- We try to explain a rating with a dot product of these embeddings.

Funk SVD for sparse matrix

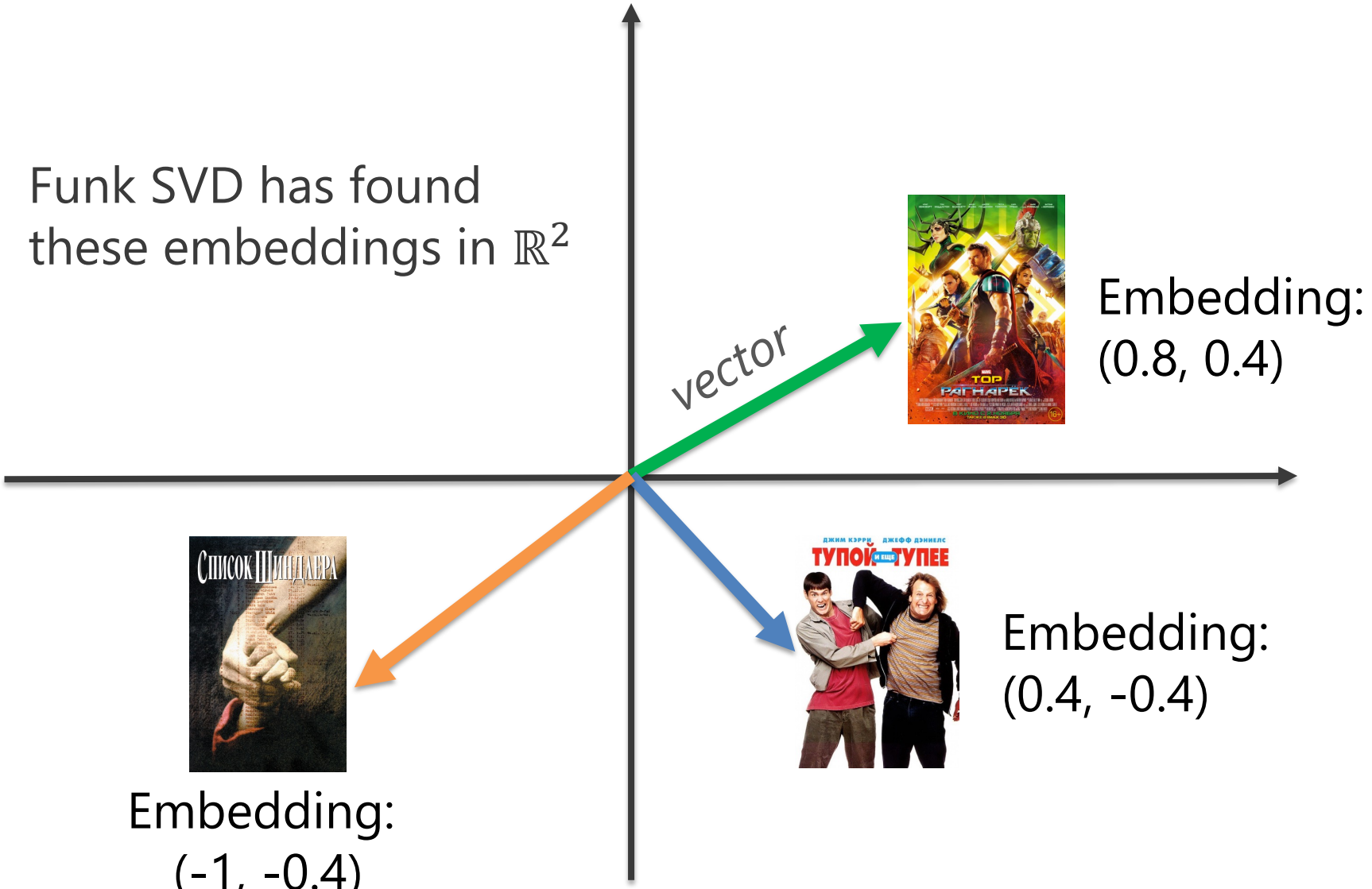


We train these embeddings using SGD:

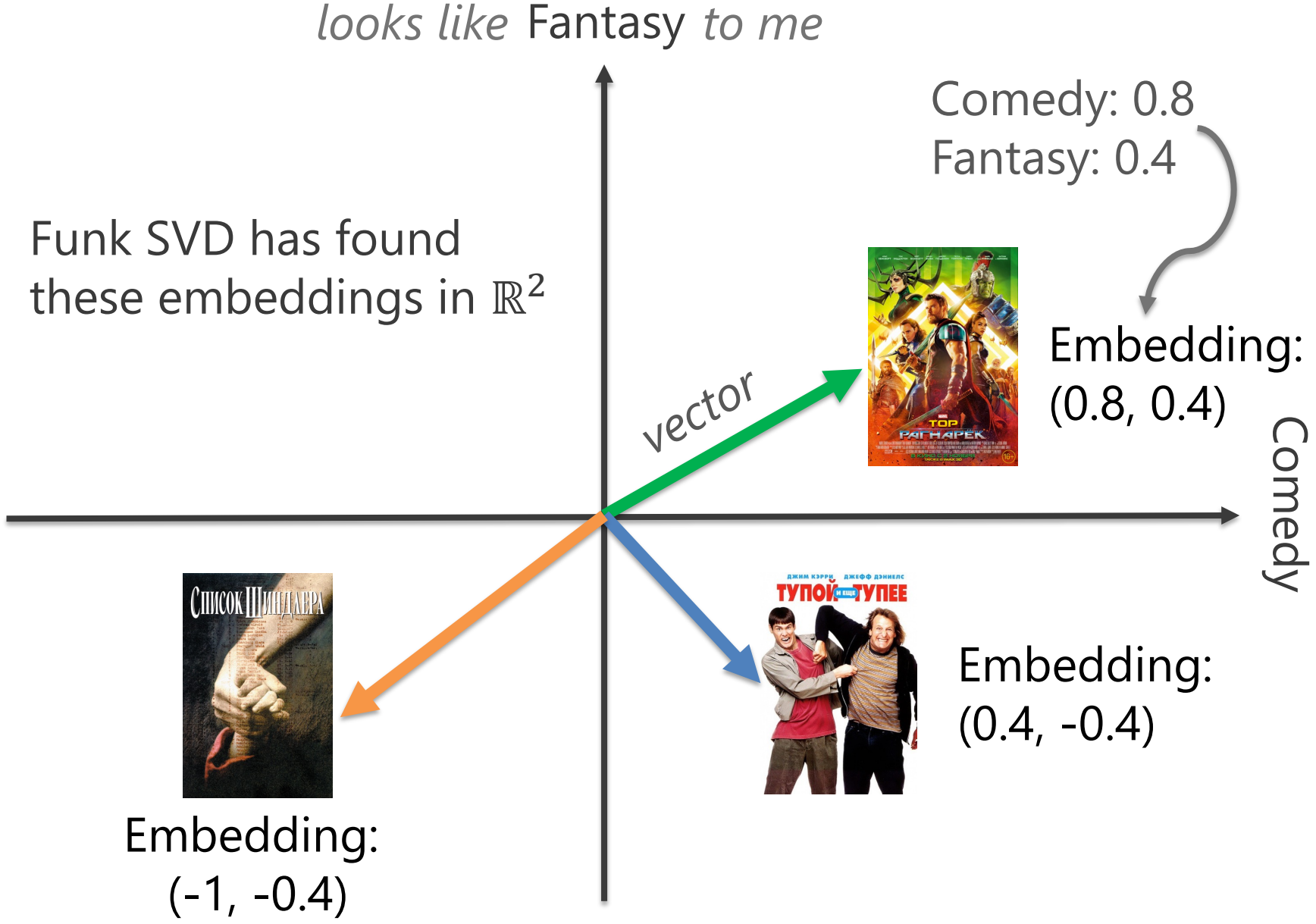
$$\min_{X,Y} \sum_{(u,i)\in R} (r_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2)$$

Interpreting item embeddings

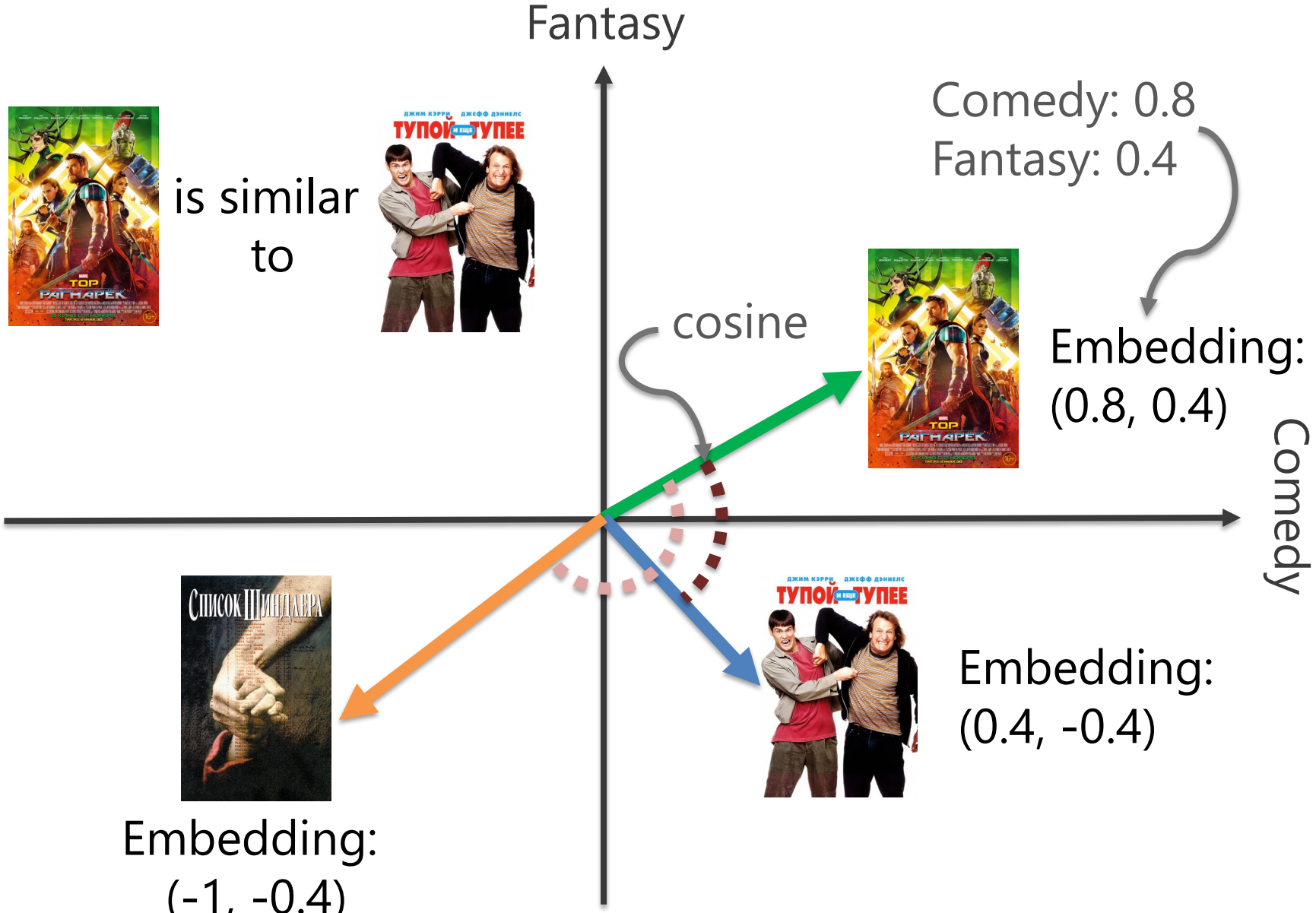
Funk SVD has found these embeddings in $\mathbb{R}^2$



Interpreting item embeddings



Interpreting item embeddings

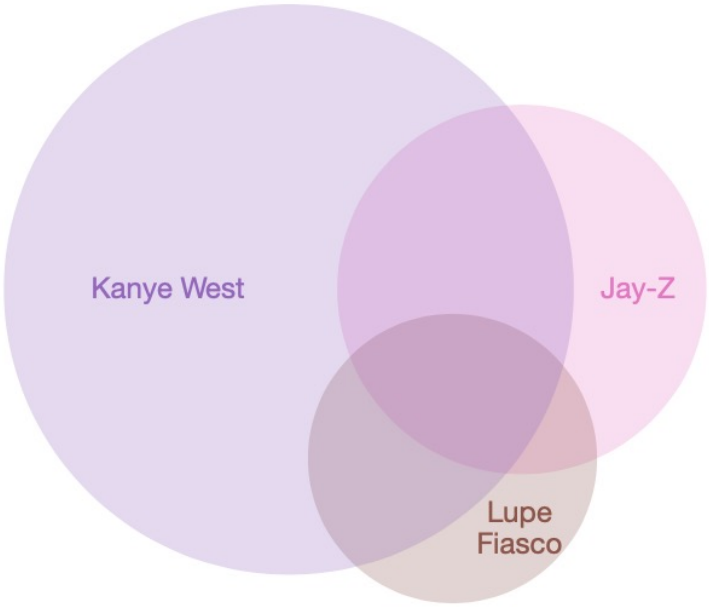


Similarity of artists

Most similar to Kanye West
based on cosine between artist embeddings:

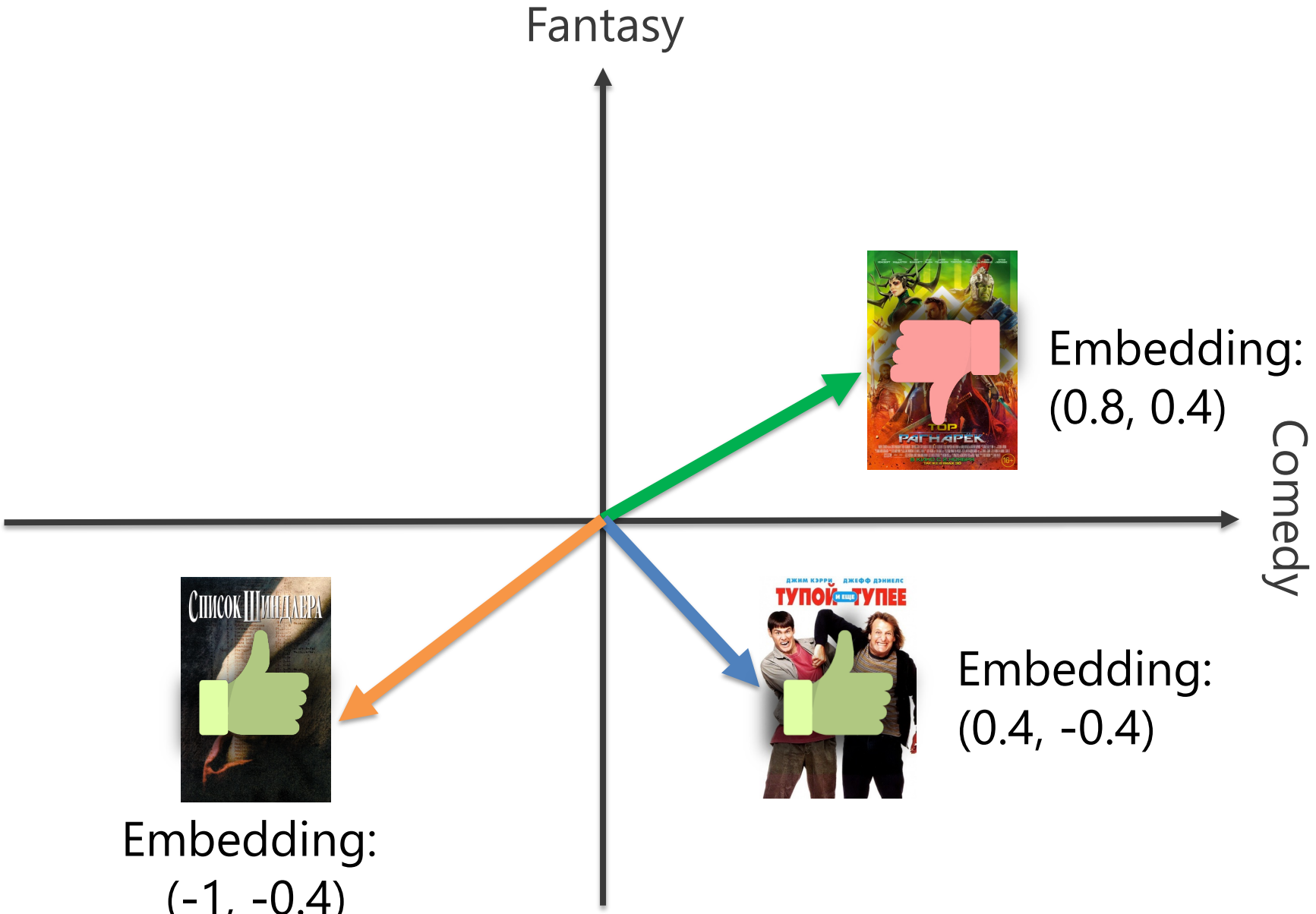
Artist	Implicit ALS
Lupe Fiasco	0.950
Jay-Z	0.928
T.I.	0.913
Lil Wayne	0.895
N*E*R*D	0.891

more ...



https://www.benfrederickson.com/matrix-factorization/

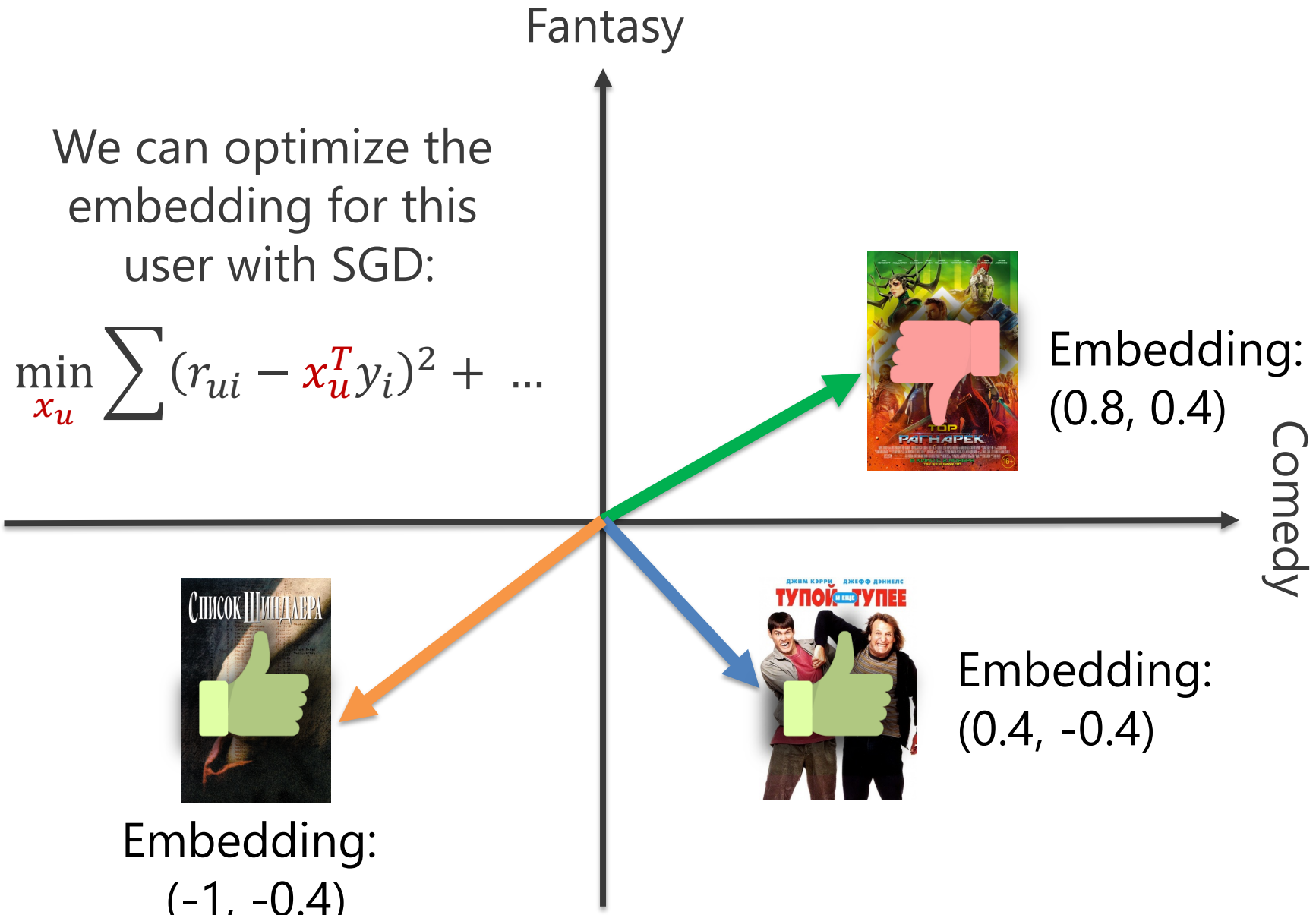
Handling new users



Handling new users

We can optimize the embedding for this user with SGD:

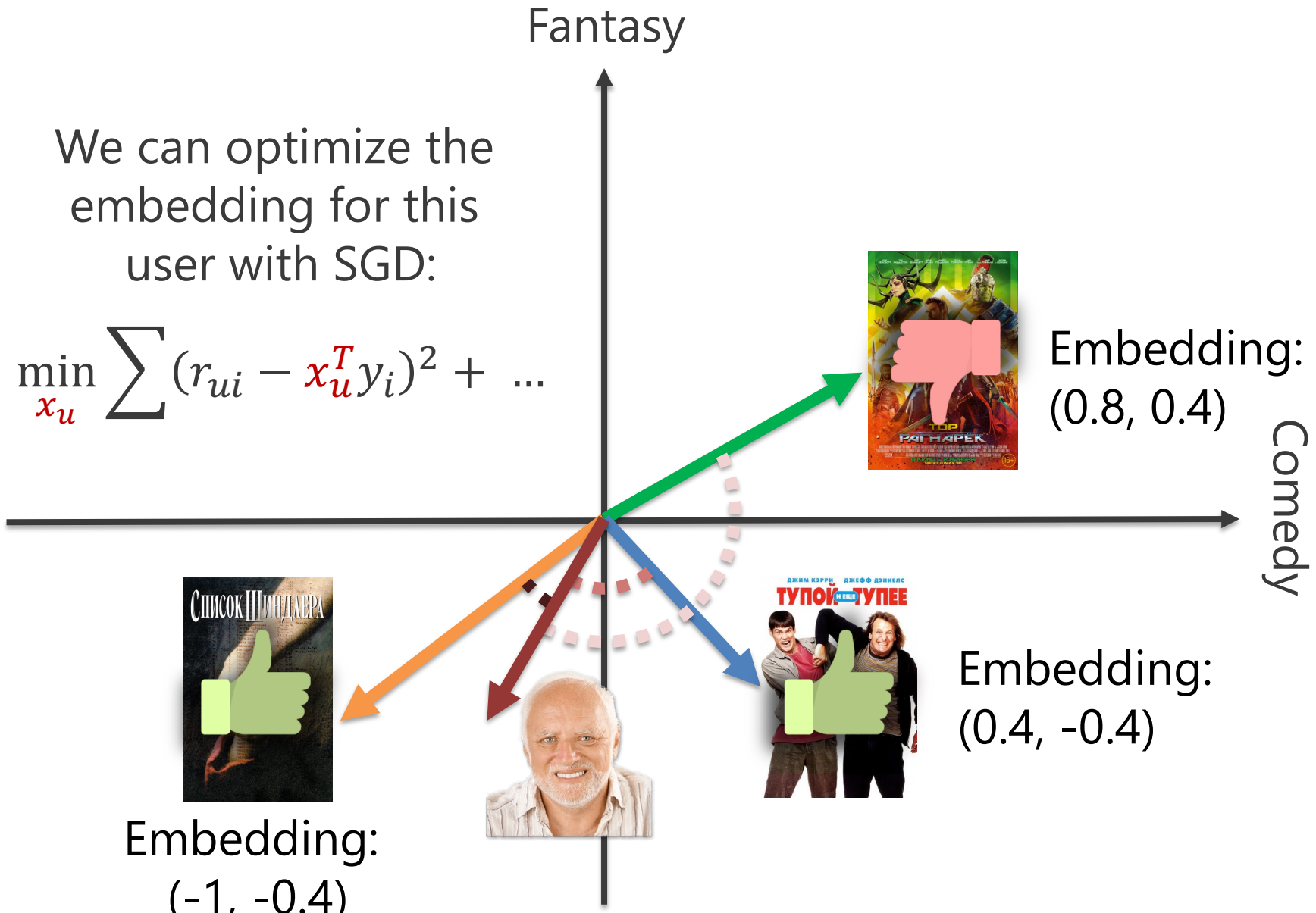
$$\min_{x_u} \sum (r_{ui} - x_u^T y_i)^2 + \ \ldots$$



Handling new users

We can optimize the embedding for this user with SGD:

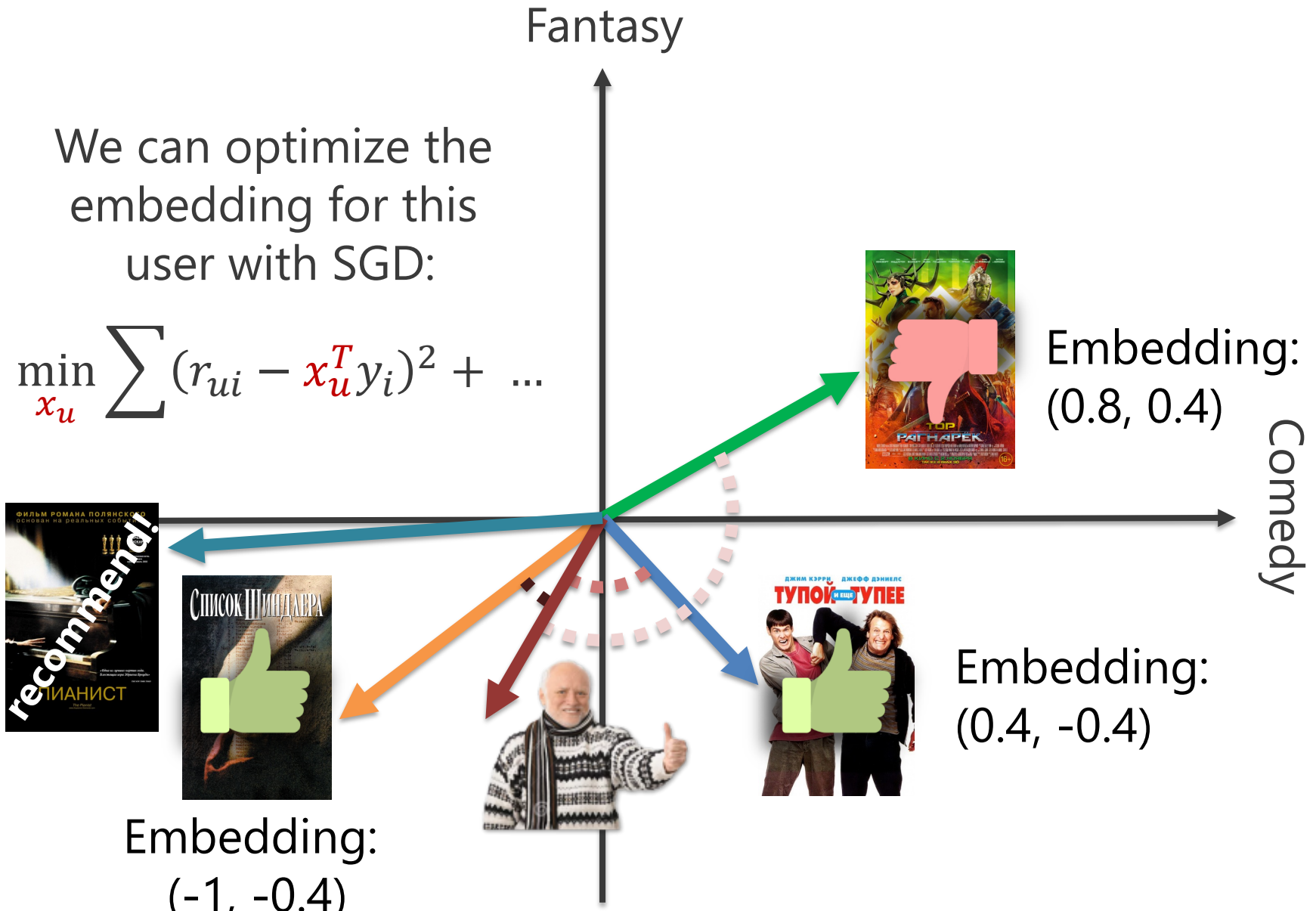
$$\min_{x_u} \sum (r_{ui} - x_u^T y_i)^2 + \ \ldots$$



Handling new users

We can optimize the embedding for this user with SGD:

$$\min_{x_u} \sum (r_{ui} - x_u^T y_i)^2 + \ \ldots$$

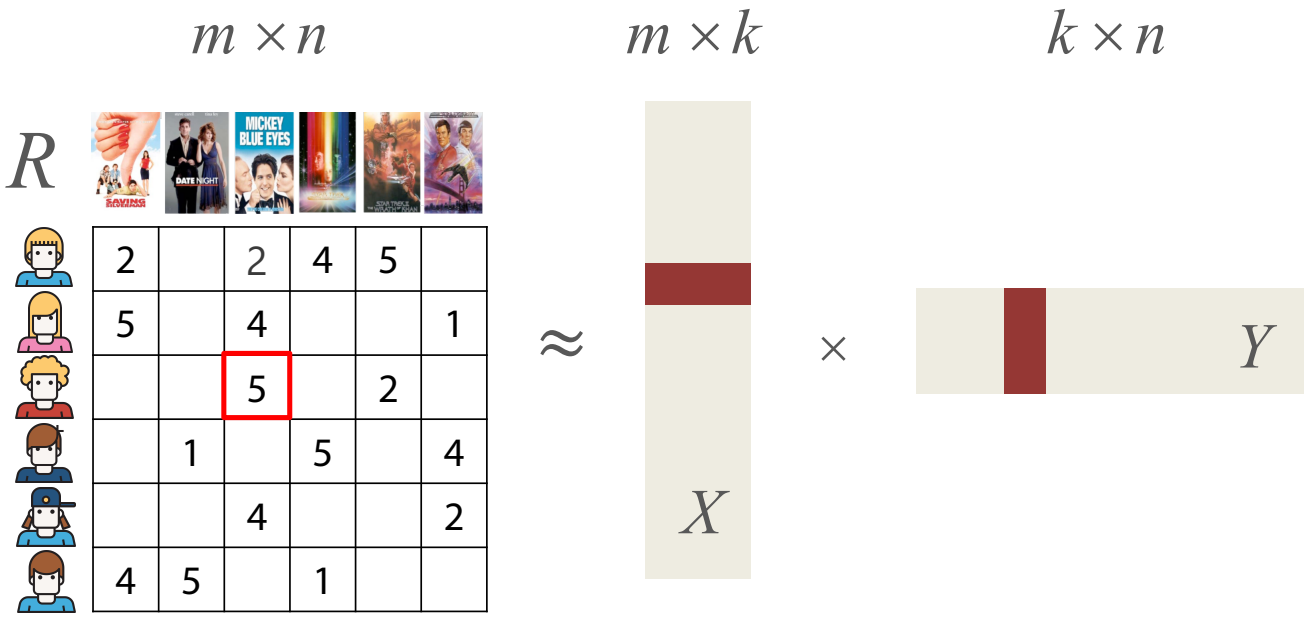


Summary

- Matrix factorizations don’t compute similarities explicitly.
- Matrix factorizations try to explain all interactions with latent characteristics.
- These characteristics can be hard to interpret, but we don’t need that interpretability anyway.
- SGD is a sequential algorithm by nature, it’s hard to scale it.
  We will solve this problem with ALS!

ALS and iALS

Alternating Least Squares (ALS)

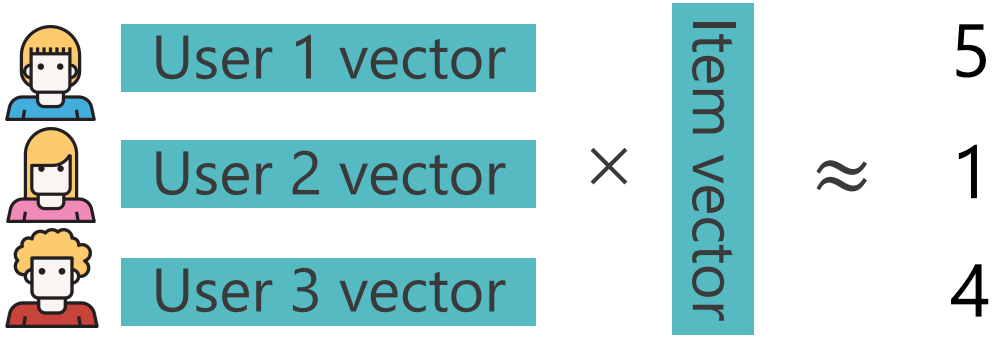


$$\min_{X,Y} \sum_{(u,i)\in R} (r_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2)$$

Alternating Least Squares (ALS)

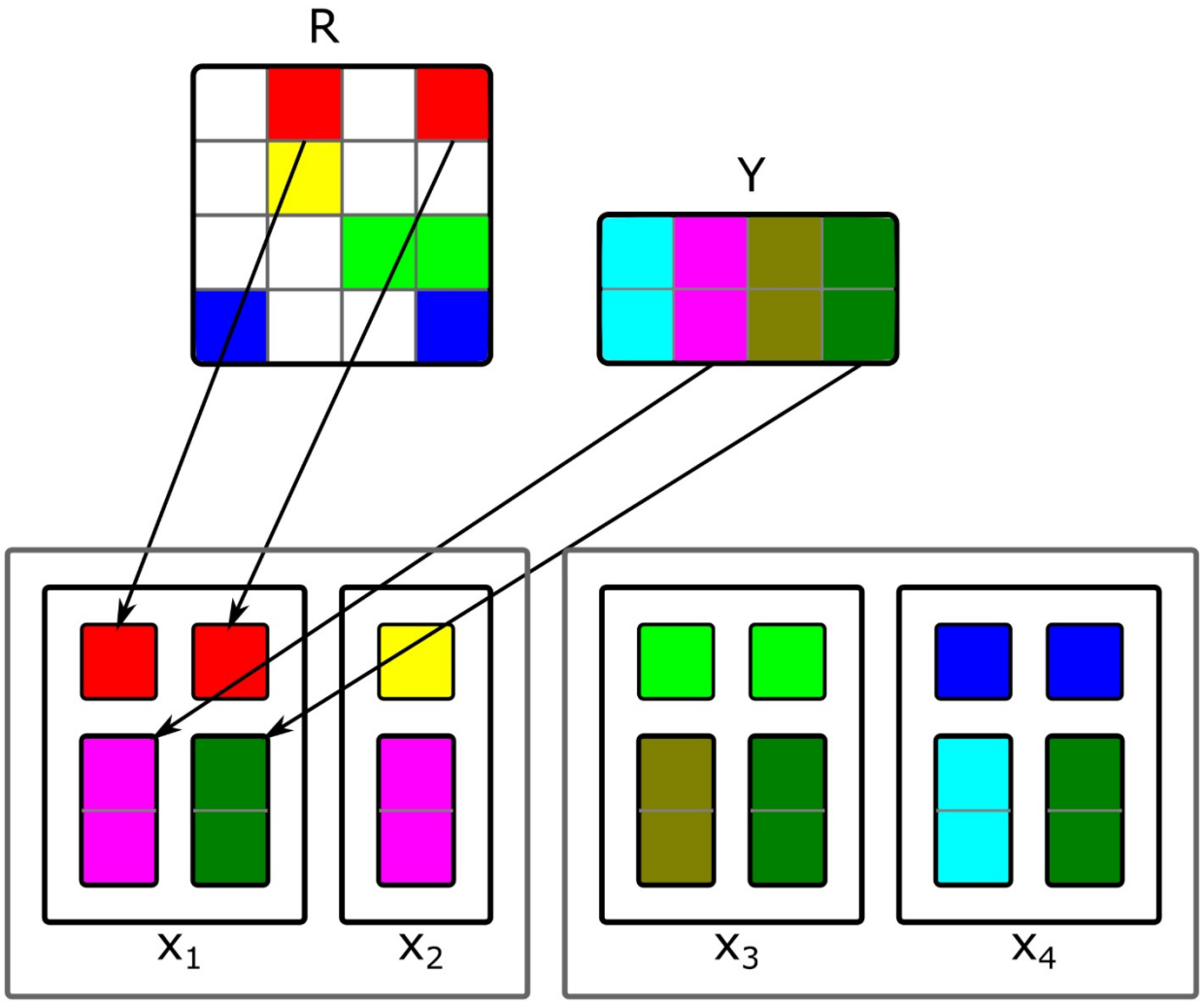
$$\min_{X,Y} \sum_{(u,i)\in R} (r_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2)$$

- Initialize X and Y with random values
- **Iterate in the loop (~10 iterations):**
  - Consider matrix X (for users) to be constant
  - Find optimal matrix Y by solving ridge regression for each item independently:



  - Consider Y to be constant and find optimal X

Scaling ALS: naïve join



R = XY, update X

Join R and Y by items → (u, r_{ui}, y_i)

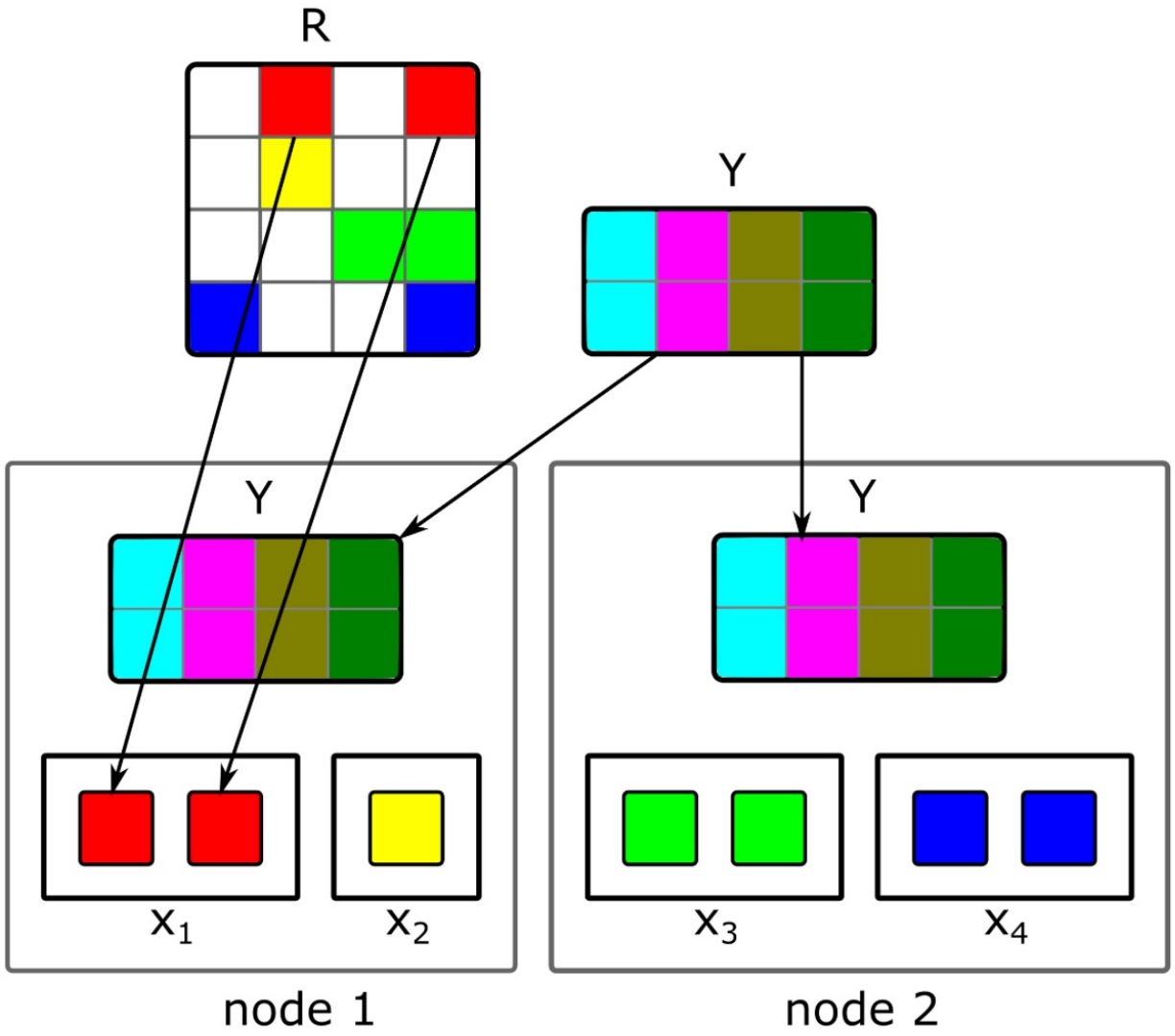
Group by u

On every node update x_u

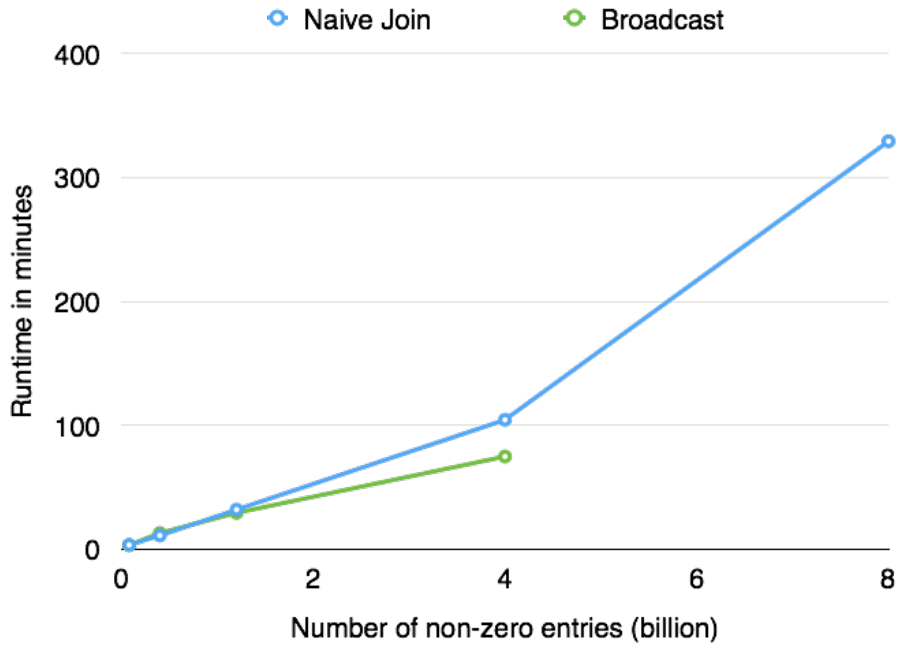
The same vector y_i will be copied many times to the node!

Embarrassingly parallel on every node (independent users)

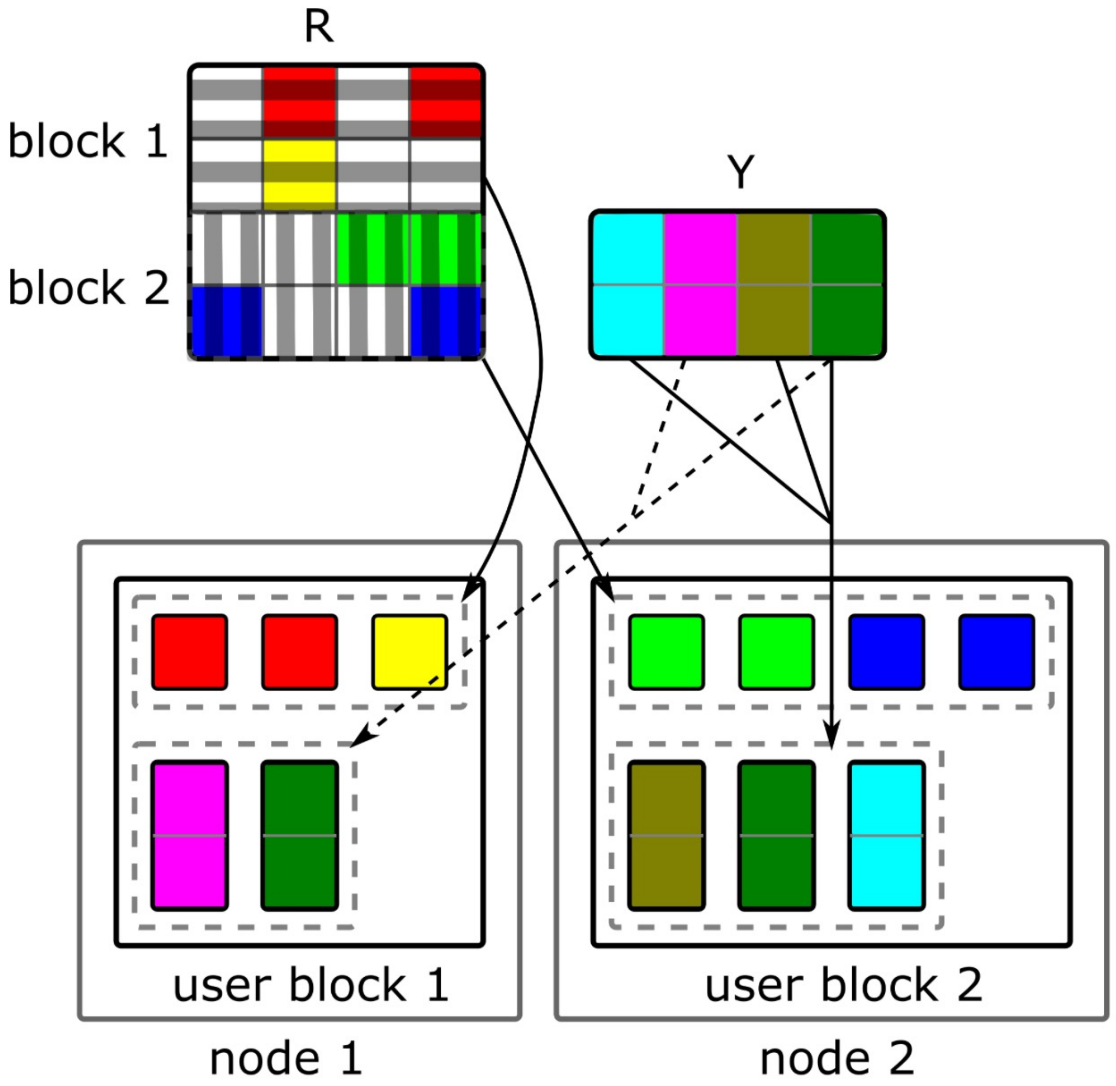
Scaling ALS: broadcast Y



Limited scalability
(Y must fit in memory)



Scaling ALS: blocked



Users and items are divided into blocks in advance

We know which items will be needed for every block (fixed mapping)

Only subset of Y columns is stored in memory of each node

Blocked ALS in Spark ML

- The more blocks you have (parameter numBlocks) the greater the chance that submatrix for a block will fit in RAM

```
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)
```

Ranking metrics

Which recommender is better: A or B?

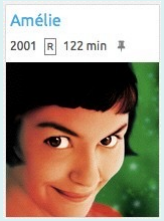
$$MAE = \frac{1}{|R|} \sum_{(u,i) \in R} |r_{ui} - \widehat{r_{ui}}|$$

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

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

Which recommender is better: A or B?

- We don’t care about exact ratings.
  We want a good ranking!

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

- Recommender A provides an ideal ranking

Ranking metrics

Point-wise:

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

Pair-wise:

$$\sum_{r_{ui} > r_{uj}} \widehat{r_{ui}} \leq \widehat{r_{uj}}$$

List-wise:

$$NDCG$$

You should pick the one that provides better business metrics like time spent, conversion rate, retention, etc.

Discounted Cumulative Gain (DCG)

$$DCG_u = \sum_{p=1} \frac{2^{r_{uip}} - 1}{\log_2(p+1)}$$

Ideal ranking (ground truth values of r_{ui} in the 1st row):

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**

Discounted Cumulative Gain (DCG)

$$DCG_u = \sum_{p=1} \frac{2^{r_{uip}} - 1}{\log_2(p+1)}$$

Ideal ranking (ground truth values of r_{ui} in the 1st row):

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**

Inversion on the top:

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**

Discounted Cumulative Gain (DCG)

$$DCG_u = \sum_{p=1} \frac{2^{r_{uip}} - 1}{\log_2(p+1)}$$

Ideal ranking (ground truth values of r_{ui} in the 1st row):

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**

Inversion on the top:

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**

Inversion on the bottom:

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 DCG (NDCG)

$$NDCG_u = \frac{DCG_u}{ideal\ DCG_u}$$

Ideal ranking (ground truth values of r_{ui} in the 1st row):

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**

Inversion on the top:

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**

Inversion on the bottom:

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**

Normalized DCG (NDCG)

$$NDCG = average\,[NDCG_u]$$

Ideal ranking (ground truth values of r_{ui} in the 1st row):

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**

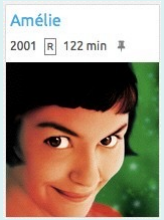
Inversion on the top:

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**

Inversion on the bottom:

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**

Example

					*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

Recommender A is better at ranking!