Matrix Factorization and Collaborative Filtering

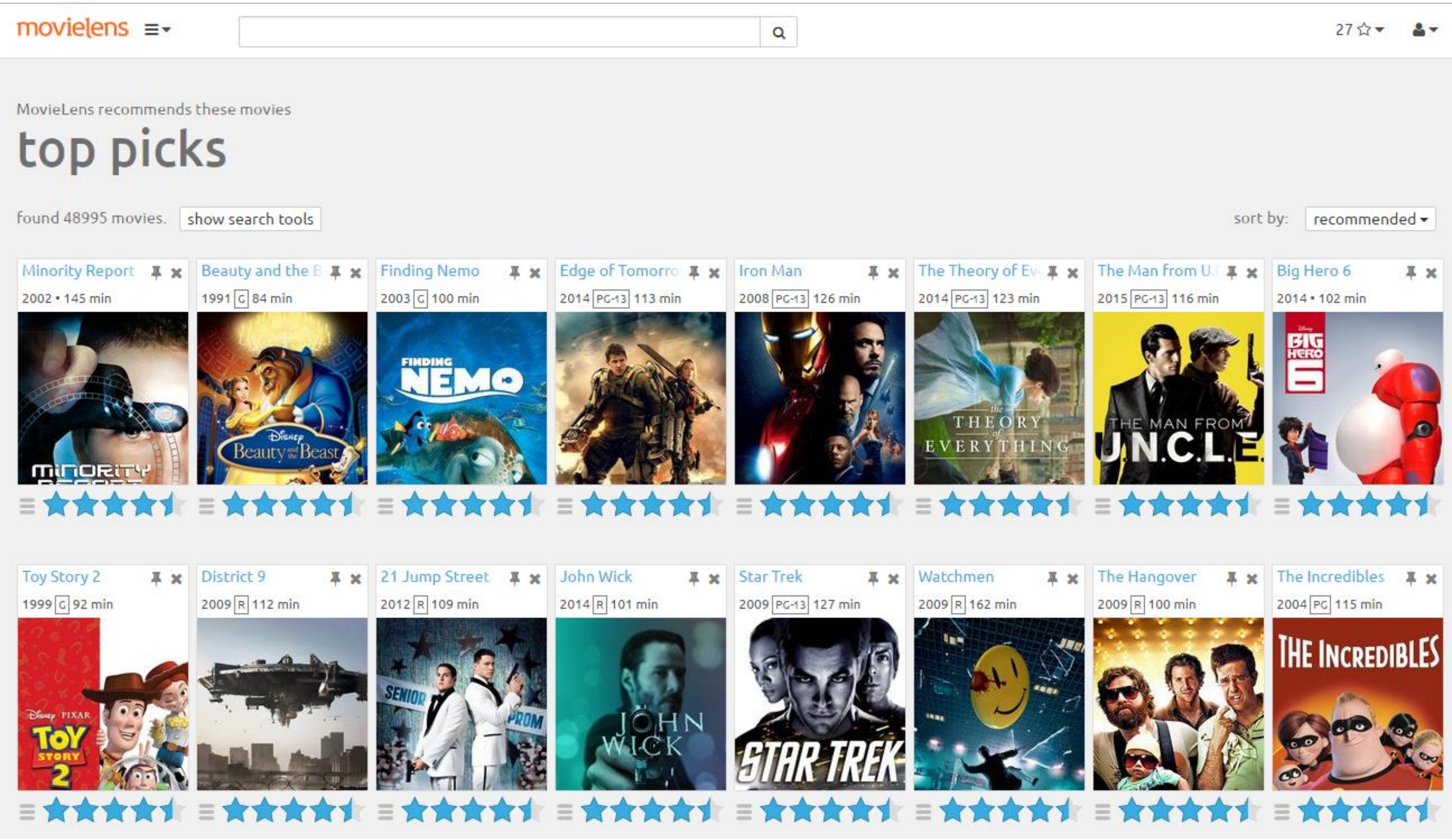
Hyewon Lim

4 Jan 2017

Outline

- **Recommender System**
- Matrix Factorization
- Reference

Recommender System



movielens

27

MovieLens recommends these movies

top picks

found 48995 movies. show search tools

sort by: recommended

Minority Report
2002 • 145 min

Beauty and the B
1991 G 84 min

Finding Nemo
2003 G 100 min

Edge of Tomorro
2014 PG-13 113 min

Iron Man
2008 PG-13 126 min

The Theory of Ev
2014 PG-13 123 min

The Man From U.
2015 PG-13 116 min

Big Hero 6
2014 • 102 min

Toy Story 2
1999 G 92 min

District 9
2009 R 112 min

21 Jump Street
2012 R 109 min

John Wick
2014 R 101 min

Star Trek
2009 PG-13 127 min

Watchmen
2009 R 162 min

The Hangover
2009 R 100 min

The Incredibles
2004 PG 115 min

Recommender System Strategies

1. Content Filtering
   – Create a profile for each user or product to characterize its nature

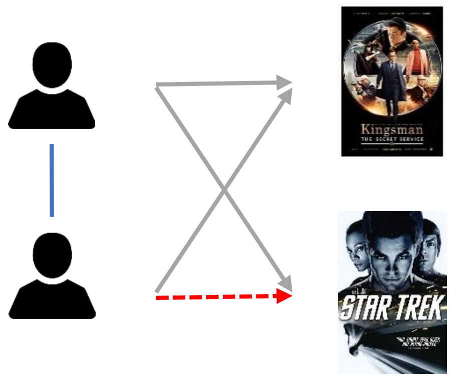
Genres: Crime, Comedy, Action, Adventure
Directors: Matthew Vaughn
Cast: Taron Egerton, Colin Firth, Samuel L. Jackson, …
Distributor: Fox
Box Office Popularity: …

Gender
Region
…
Answers provided on a suitable questionnaire

Recommender System Strategies

2. Collaborative filtering
   - Rely only on past user behavior
   - Everyday examples
     - Bestseller lists
     - Top 40 music lists
     - Unmarked but well-used paths thru the woods
     - The "recent returns" shelf at the library

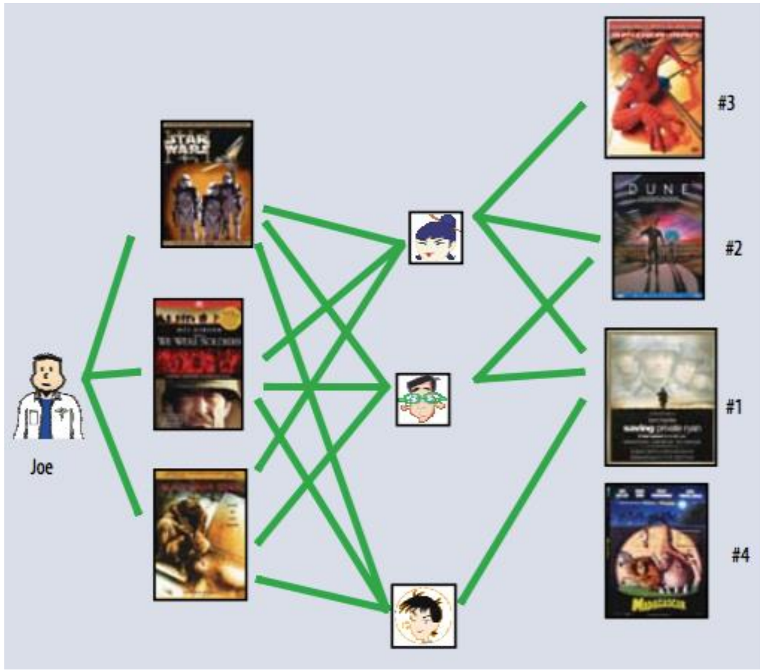
   - Common insight: **personal tastes are correlated**

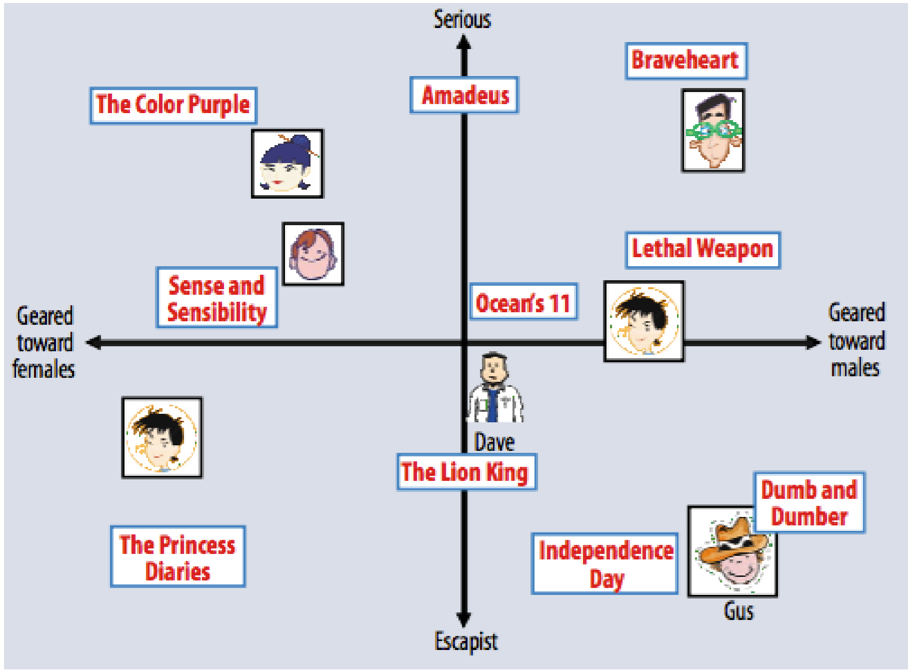


Types of Collaborative Filtering

a. Neighborhood Methods
   - Find neighbors based on similarity of movie preferences
   - Recommend movies that those neighbors watched

b. Latent Factor Methods
   - Characterize both items and users
   - Recommend a movie based on its proximity to the user in the latent space



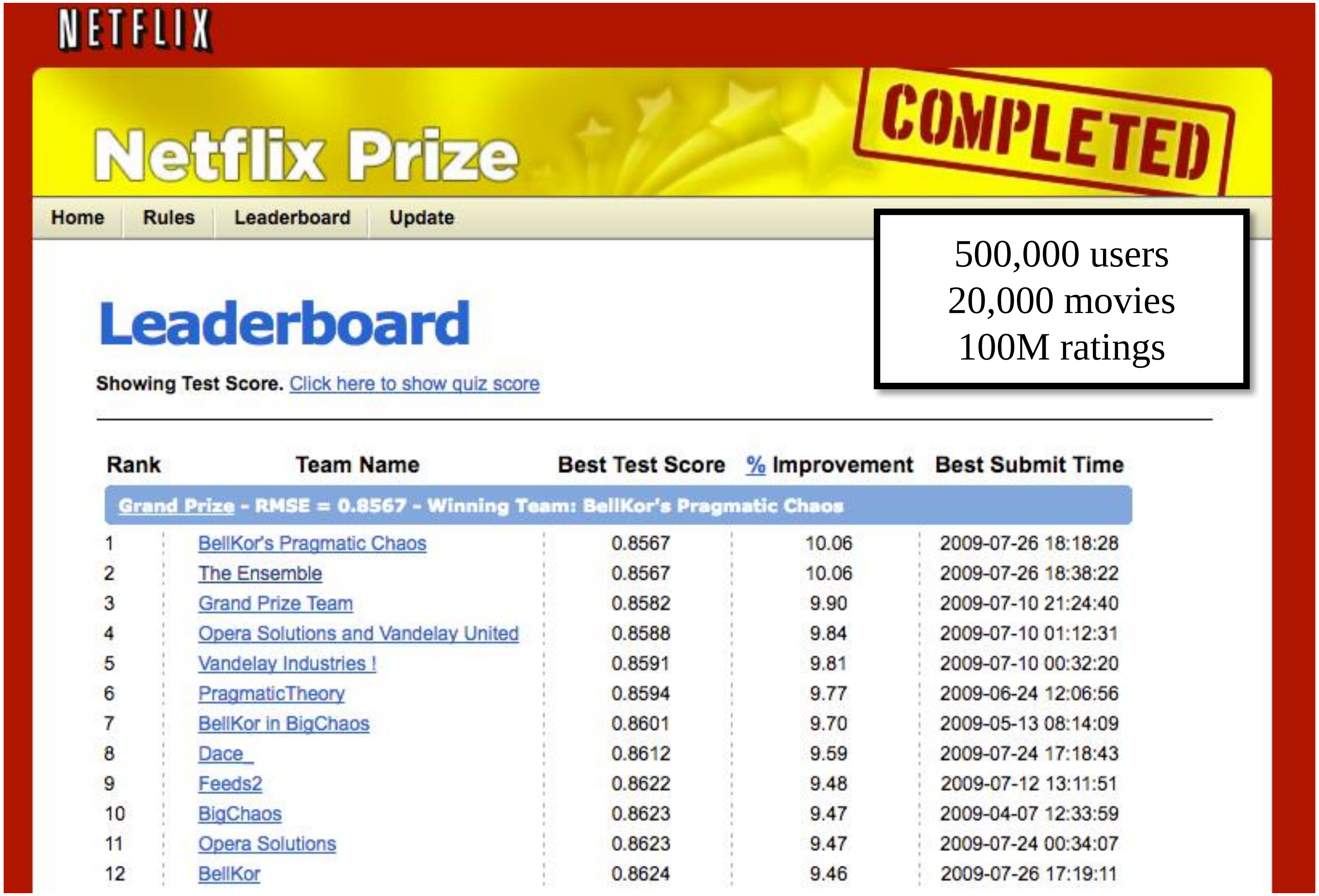


* Figures in [3]

Outline

- Recommender System
- **Matrix Factorization**
- Reference

Netflix Prize



500,000 users
20,000 movies
100M ratings

Leaderboard

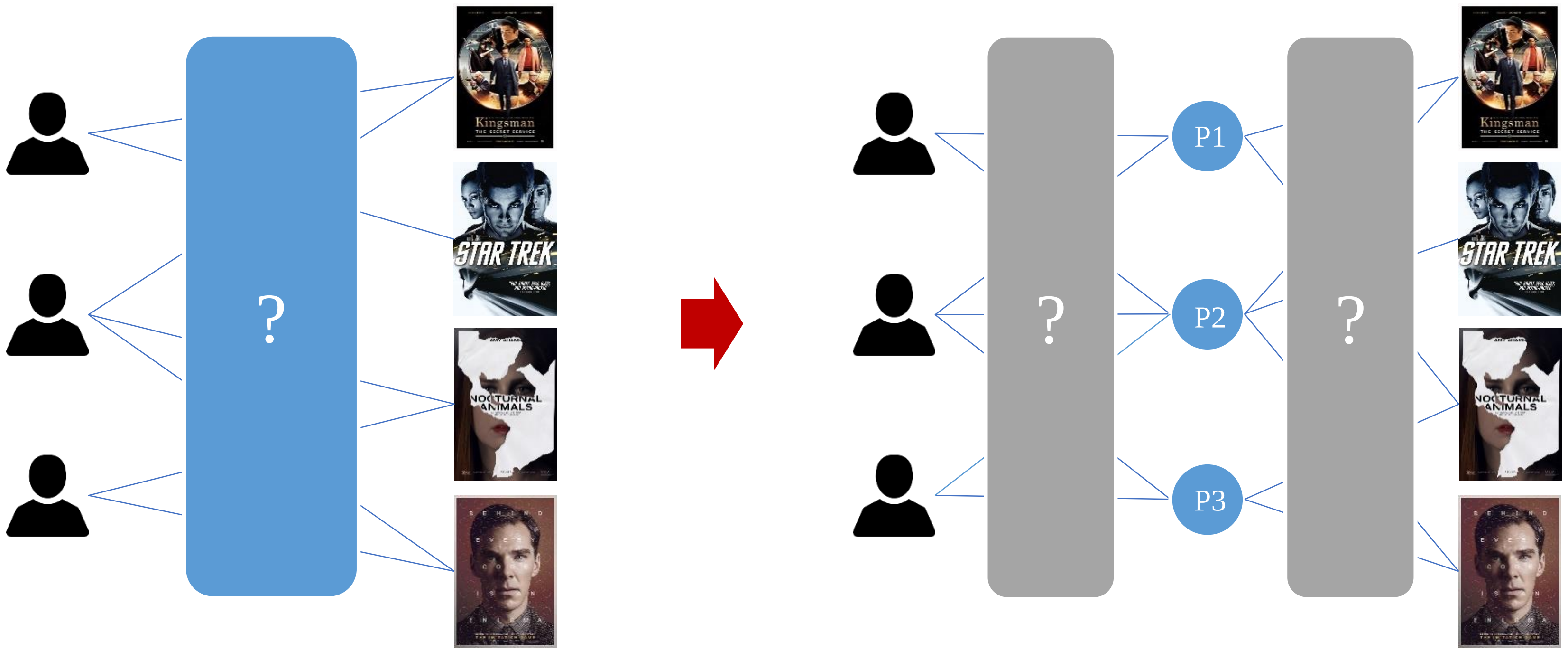
Showing Test Score. Click here to show quiz score

Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
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
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11

Matrix Factorization

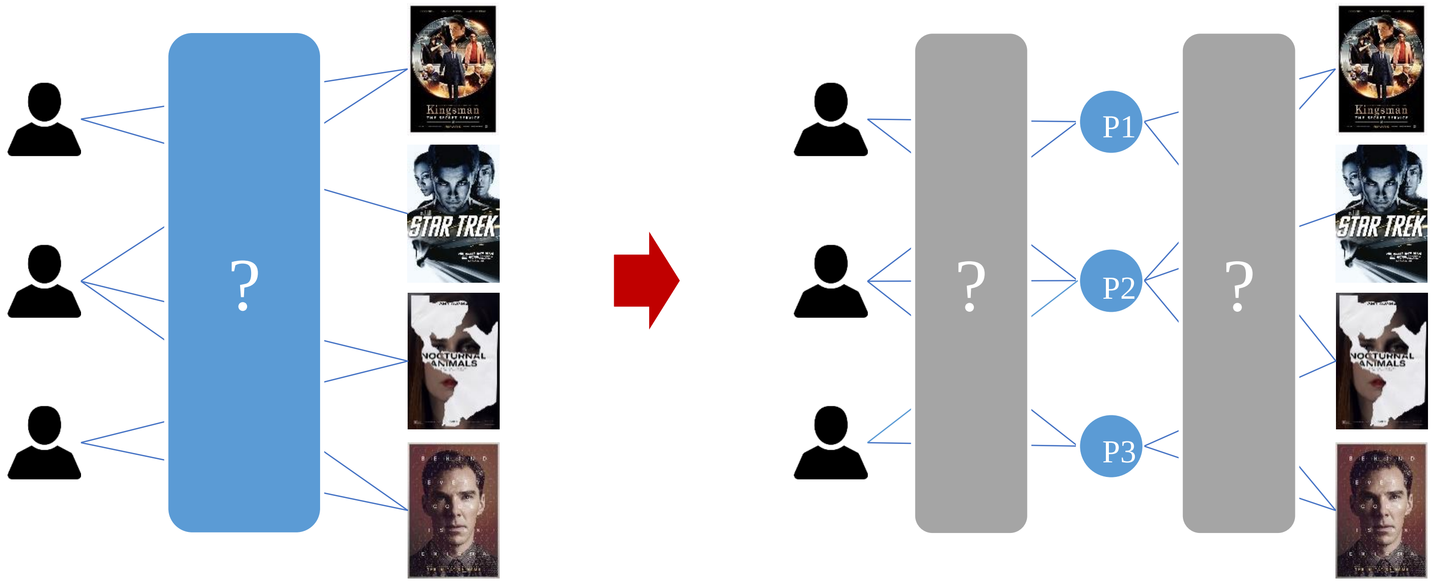
- Assume latent factors in user preference



* Redraw figures in [2]

Matrix Factorization

- Assume latent factors in user preference



	I_1	I_2	I_3	I_4
U_1				
U_2				
U_3				

	P_1	P_2	P_3
U_1			
U_2			
U_3			

	I_1	I_2	I_3	I_4
P_1				
P_2				
P_3				

* Redraw figures in [2]

Singular Value Decomposition

$$\boxed{A} = \boxed{U} \times \boxed{s} \times \boxed{V^T}$$

```
[ 4.   2.   3.   5.   1.]
[ 0.   3.   0.   4.   2.]
[ 5.   4.   3.   3.   0.]
[ 0.   0.   5.   5.   2.]
[ 5.   0.   0.   5.   0.]
```

$=$

```
[-0.54   0.03  -0.021  0.099 -0.835]
[-0.29  -0.225  0.393 -0.84   0.07 ]
[-0.506  0.372  0.574  0.374  0.371]
[-0.417 -0.79  -0.217  0.277  0.279]
[-0.441  0.432 -0.685 -0.26   0.287]
```

$\times$

```
[ 13.707   0.       0.       0.       0.   ]
[  0.      5.607    0.       0.       0.   ]
[  0.      0.       3.791    0.       0.   ]
[  0.      0.       0.       3.645    0.   ]
[  0.      0.       0.       0.       0.155]
```

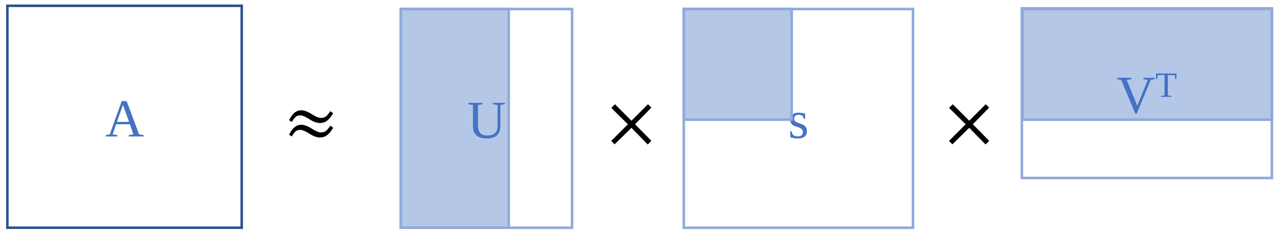
$\times$

```
[-0.503 -0.29  -0.381 -0.705 -0.143]
[ 0.738  0.156 -0.489 -0.254 -0.357]
[-0.169  0.905  0.15  -0.35   0.087]
[ 0.265 -0.227  0.769 -0.455 -0.282]
[-0.321  0.147  0.029  0.33  -0.875]
```

$\Downarrow$

We can drop less important information

Singular Value Decomposition



```
[-0.54    0.03   -0.021  0.099 -0.835]
[-0.29   -0.225   0.393 -0.84    0.07 ]
[-0.506   0.372   0.574  0.374   0.371]
[-0.417 -0.79   -0.217   0.277   0.279]
[-0.441   0.432 -0.685 -0.26    0.287]
```

×

```
[ 13.707   0.      0.      0.      0.   ]
[  0.      5.607   0.      0.      0.   ]
[  0.      0.      3.791   0.      0.   ]
[  0.      0.      0.      3.645   0.   ]
[  0.      0.      0.      0.      0.155]
```

×

```
[-0.503 -0.29   -0.381 -0.705 -0.143]
[ 0.738   0.156 -0.489 -0.254 -0.357]
[-0.169   0.905   0.15   -0.35    0.087]
[ 0.265 -0.227   0.769 -0.455 -0.282]
[-0.321   0.147   0.029   0.33   -0.875]
```

```
[-0.54    0.03   -0.021  0.099 -0.835]
[-0.29   -0.225   0.393 -0.84    0.07 ]
[-0.506   0.372   0.574  0.374   0.371]
[-0.417 -0.79   -0.217   0.277   0.279]
[-0.441   0.432 -0.685 -0.26    0.287]
```

×

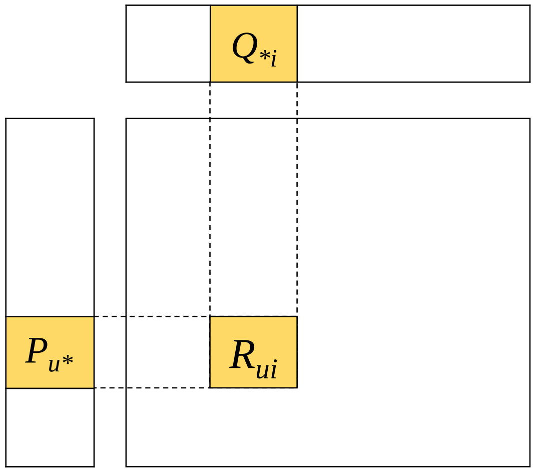
```
[ 13.707   0.      0.      0.      0.   ]
[  0.      5.607   0.      0.      0.   ]
[  0.      0.      3.791   0.      0.   ]
[  0.      0.      0.      0.      0.   ]
[  0.      0.      0.      0.      0.   ]
```

×

```
[-0.503 -0.29   -0.381 -0.705 -0.143]
[ 0.738   0.156 -0.489 -0.254 -0.357]
[-0.169   0.905   0.15   -0.35    0.087]
[ 0.265 -0.227   0.769 -0.455 -0.282]
[-0.321   0.147   0.029   0.33   -0.875]
```

Matrix Factorization

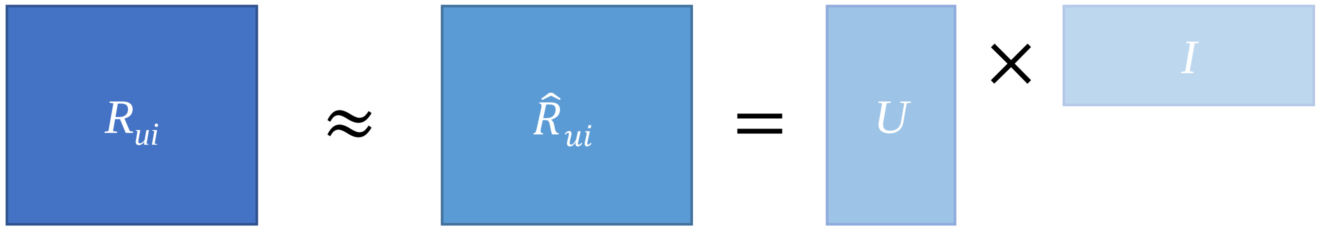
- Matrices
  - User vector
    - $(P_{u*})^T \in \mathbb{R}^f$
  - Item vectors:
    - $(Q_{*i}) \in \mathbb{R}^f$
  - Rating prediction
    - $R_{ui} = P_{u*}Q_{*i} = [PQ]_{ui}$



- Vectors
  - User vector
    - $p_u \in \mathbb{R}^r$
  - Item vectors:
    - $q_i \in \mathbb{R}^r$
  - Rating prediction
    - $\hat{r}_{ui} = q_i^T p_u$
  - Set of non-zero entries
    - $\kappa = \{(u, i)\colon r_{ui} \neq 0\}$
  - Objective
    - $\min_{q*,p*} \sum_{(u,i)\in\kappa}(r_{ui} - q_i^T p_u)^2$

* Figures in [1]

Matrix Factorization



- Minimize the error between R and R̂

$$\min_{q*,p*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2$$

$$\min_{q*,p*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \boxed{\lambda(\|q_i\|^2 + \|p_u\|^2)}$$

Regularization factor
- avoid overfitting
- make simple model

* Figures in [2]

Matrix Factorization

- How to deal with empty cells in matrix
  - With 0
  - With the average of the whole users
  - With the average of each user

	I_1	I_2	I_3	I_4
U_1		3	4	2
U_2	5			
U_3	3		2	

- Consider user bias and item bias

$$\boldsymbol{b_{ui} = \mu + b_i + b_u}$$

μ: average of the whole users

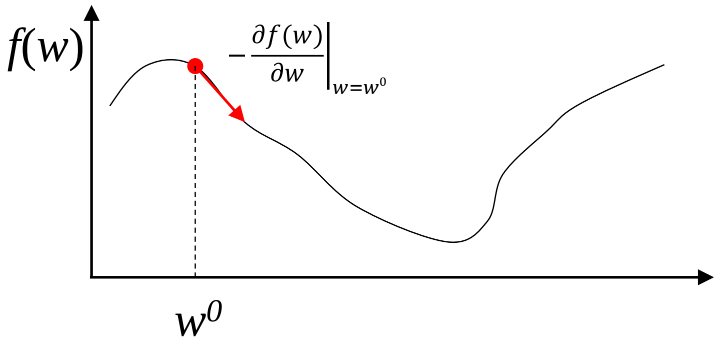
b_i, b_u: the observed deviations of u and i

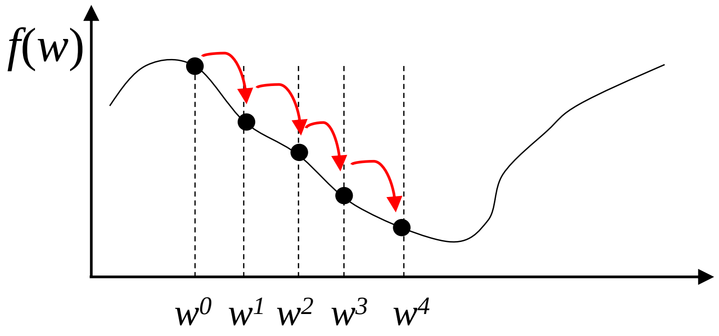
$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

$$\min_{q*,p*,b*} \sum_{(u,i)\in\kappa} (r_{ui} - \mu - b_i - b_u - p_u^T q_i)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + b_u^2 + b_i^2)$$

Matrix Factorization

- Approaches to minimizing $\min_{q*,p*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$
  1. Stochastic gradient descent





- Associated prediction error e_{ui}
  - $q_i \leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda \cdot q_i)$
  - $p_u \leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda \cdot p_u)$

* Figures in [2]

Matrix Factorization

- Approaches to minimizing $\min_{q*,p*} \sum_{(u,i)\in\kappa}(r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$

  2. Alternating least squares
     - Rotate between fixing the q_i's and fixing the p_u's
       - When all p_u's are fixed, the system recomputes the q_i's by solving a least-squares problems, and vice versa

     - Stochastic gradient descent is easier and faster than ALS in general, ALS is favorable in at least two cases
       - When the system can use parallelization
       - For systems centered on implicit data

HOSVD [4]

- SVD on each matrix

  $A_1 = U^{(1)} \cdot S_1 \cdot V_1^T$

  $A_2 = U^{(2)} \cdot S_2 \cdot V_2^T$

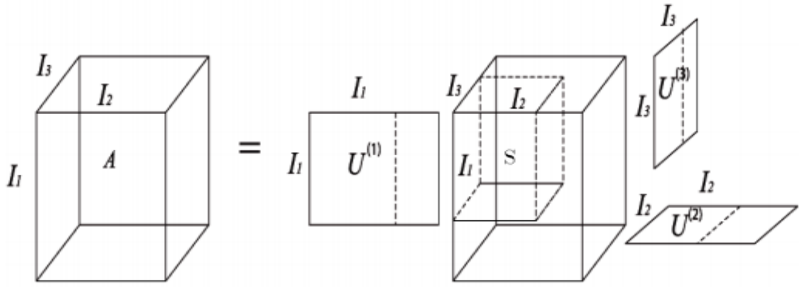
  $A_3 = U^{(3)} \cdot S_3 \cdot V_3^T$

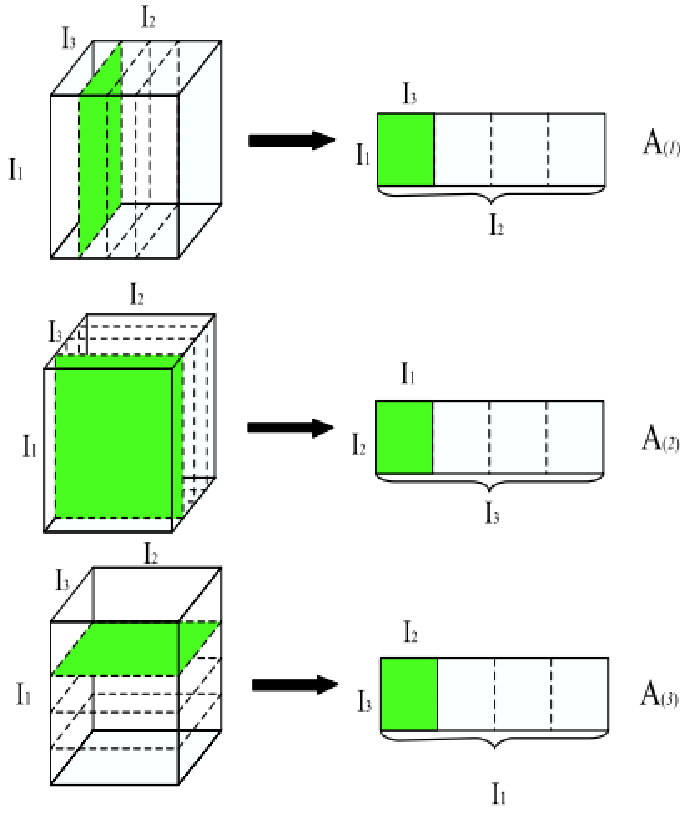
- Construction of core tensor

$$\mathcal{S} = \mathcal{A} \times_1 {U_{c_1}^{(1)}}^T \times_2 {U_{c_2}^{(2)}}^T \times_3 {U_{c_3}^{(3)}}^T$$

- Construction of tensor $\hat{\mathcal{A}}$

$$\hat{\mathcal{A}} = \mathcal{S} \times_1 U_{c_1}^{(1)} \times_2 U_{c_2}^{(2)} \times_3 U_{c_3}^{(3)}$$





Example

- MF in Python

```python
#!/usr/bin/python

import numpy as np

np.set_printoptions(precision = 3) # set decimal display

# Matrix
A = np.zeros((5, 5))

A[0, 4] = 1
A[0, 1] = A[1, 4] = A[3, 4] = 2
A[0, 2] = A[1, 1] = A[2, 2] = A[2, 3] = 3
A[0, 0] = A[1, 3] = A[2, 1] = 4
A[0, 3] = A[2, 0] = A[3, 2] = A[3, 3] = A[4, 0] = A[4, 3] = 5

# SVD
U, s, V = np.linalg.svd(A, full_matrices = True)

# Reconstruction
S = np.diag(s)

P = np.dot(U, np.dot(S, V))

```

```
[ 4.   2.   3.   5.   1.]
[ 0.   3.   0.   4.   2.]
[ 5.   4.   3.   3.   0.]
[ 0.   0.   5.   5.   2.]
[ 5.   0.   0.   5.   0.]
```

```
[   4e+00     2e+00     3e+00     5e+00     1e+00]
[  -4e-16     3e+00     3e-15     4e+00     2e+00]
[   5e+00     4e+00     3e+00     3e+00     7e-16]
[  -4e-15     4e-15     5e+00     5e+00     2e+00]
[   5e+00     1e-15    -2e-16     5e+00    -5e-16]
```

Example

- MF in Python with r

```python
#!/usr/bin/python

import rpy2.robjects as robjects

r = robjects.r

r('''
    rsvd <- function() {
        # MATRIX
        A <- matrix(c(4, 2, 3, 5, 1, 0, 3, 0, 4, 2, 5, 4, 3, 3, 0, 0, 0, 5, 5, 2, 5, 0, 0, 5, 0), nrow = 5, ncol = 5, byrow = TRUE)

        # SVD
        result <- svd(A)

        # RECONSTRUCTION
        U <- result$u
        s <- result$d
        V <- result$v

        ApproxA <- U %*% diag(s) %*% t(V)
    }
    ''')

svd = r['rsvd']

result = svd()

print result

```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    4    2    3    5    1
[2,]    0    3    0    4    2
[3,]    5    4    3    3    0
[4,]    0    0    5    5    2
[5,]    5    0    0    5    0
```

```
              [,1]         [,2]          [,3] [,4]          [,5]
[1,]  4.000000e+00 2.000000e+00  3.000000e+00    5  1.000000e+00
[2,] -1.955901e-16 3.000000e+00  5.999975e-16    4  2.000000e+00
[3,]  5.000000e+00 4.000000e+00  3.000000e+00    3 -1.110223e-16
[4,] -6.366435e-16 3.387048e-15  5.000000e+00    5  2.000000e+00
[5,]  5.000000e+00 1.949829e-15 -6.570265e-16    5  1.408595e-15
```

Reference

1. Slides in "Matrix Factorization and Collaborative Filtering"
   - By Matt Gormley (Carnegie Mellon Univ.)
2. Slides in "Recommender Systems"
   - By Jee-Hyong Lee (Sungkyunkwan Univ.)
3. Y. Koren *et al*., "Matrix Factorization Techniques for Recommender Systems," Journal Computer, 42(8), 2009
4. P. Symeonidis *et.al*, "Tag Recommendations based on Tensor Dimensionality Reduction," Recsys'08