Recommendation using Matrix Decomposition

Debapriyo Majumdar

Indian Statistical Institute Kolkata

Singular Value Decomposition

- If A is an $m \times n$ matrix with rank r, then there exists a factorization of A as

$$\underbrace{A}_{m\times n} = \underbrace{U}_{m\times m} \underbrace{\Sigma}_{m\times n} \underbrace{V^T}_{n\times n}$$

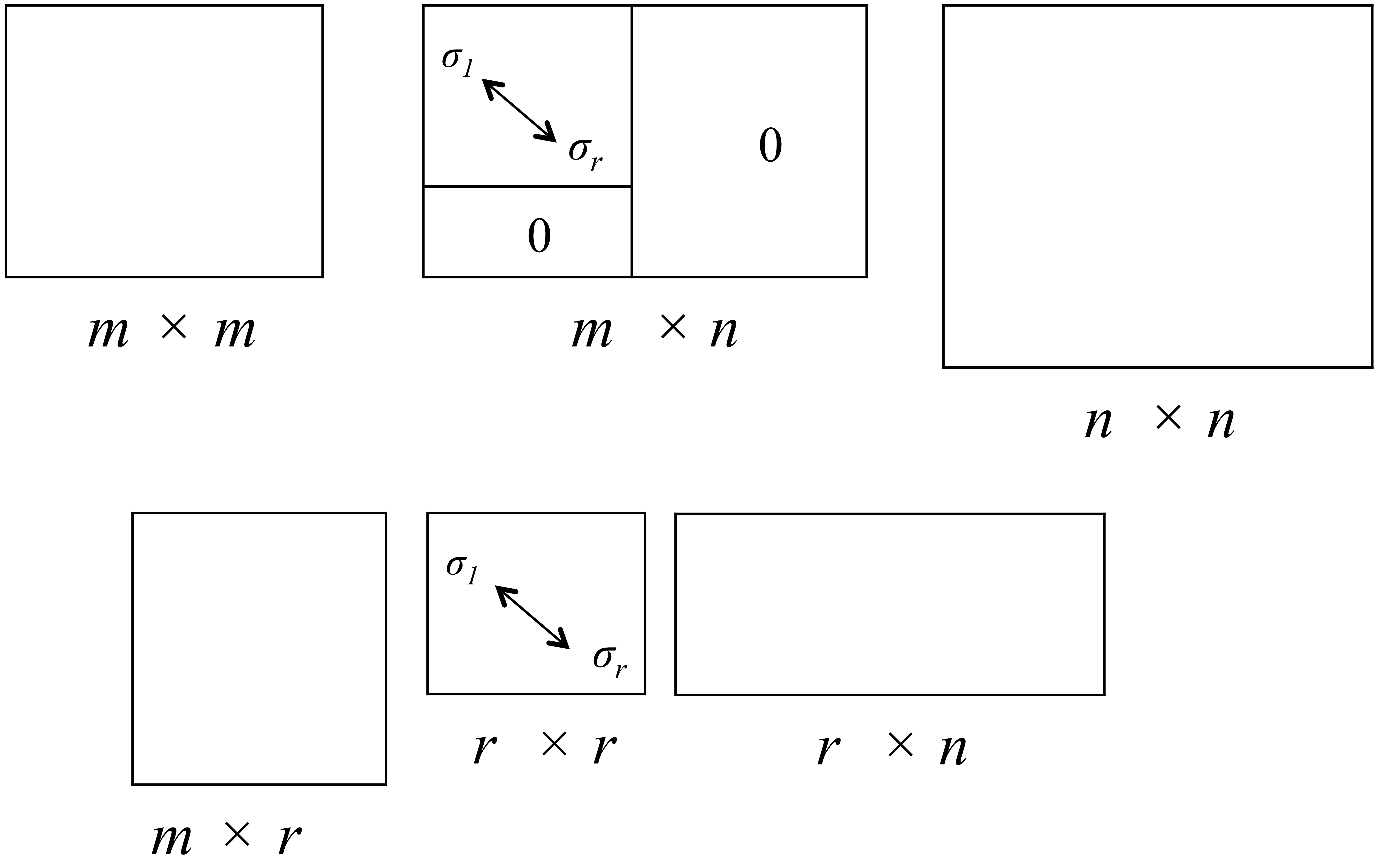
where U ($m \times m$) and V ($n \times n$) are orthogonal, and Σ ($m \times n$) is a diagonal-like matrix

$\Sigma = (\sigma_{ij})$, where $\sigma_{ii} = \sigma_i$, for $i = 1, \ldots, r$ are the singular values of A, all non-diagonal entries of Σ are zero $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$

Columns of U (V) are the left (right) singular vectors of A

Singular Value Decomposition

$$\underbrace{A}_{m\times n} = \underbrace{U}_{m\times m}\ \underbrace{\Sigma}_{m\times n}\ \underbrace{V^T}_{n\times n}$$



- It is equivalent to not consider the columns after the *r*-th column of *U* Since the singular values are zero after σ_r

Low Rank Approximation using SVD

- The largest singular values are most significant
- The corresponding singular vectors are the *principal* components
- The ones in the tail are not (think — noise)
- Discard the singular values after the *k*-th, and the corresponding columns of *U* and *V* and

$$\underbrace{A}_{m\times n} = \underbrace{U}_{m\times m}\ \underbrace{\Sigma}_{m\times n}\ \underbrace{V^T}_{n\times n}$$

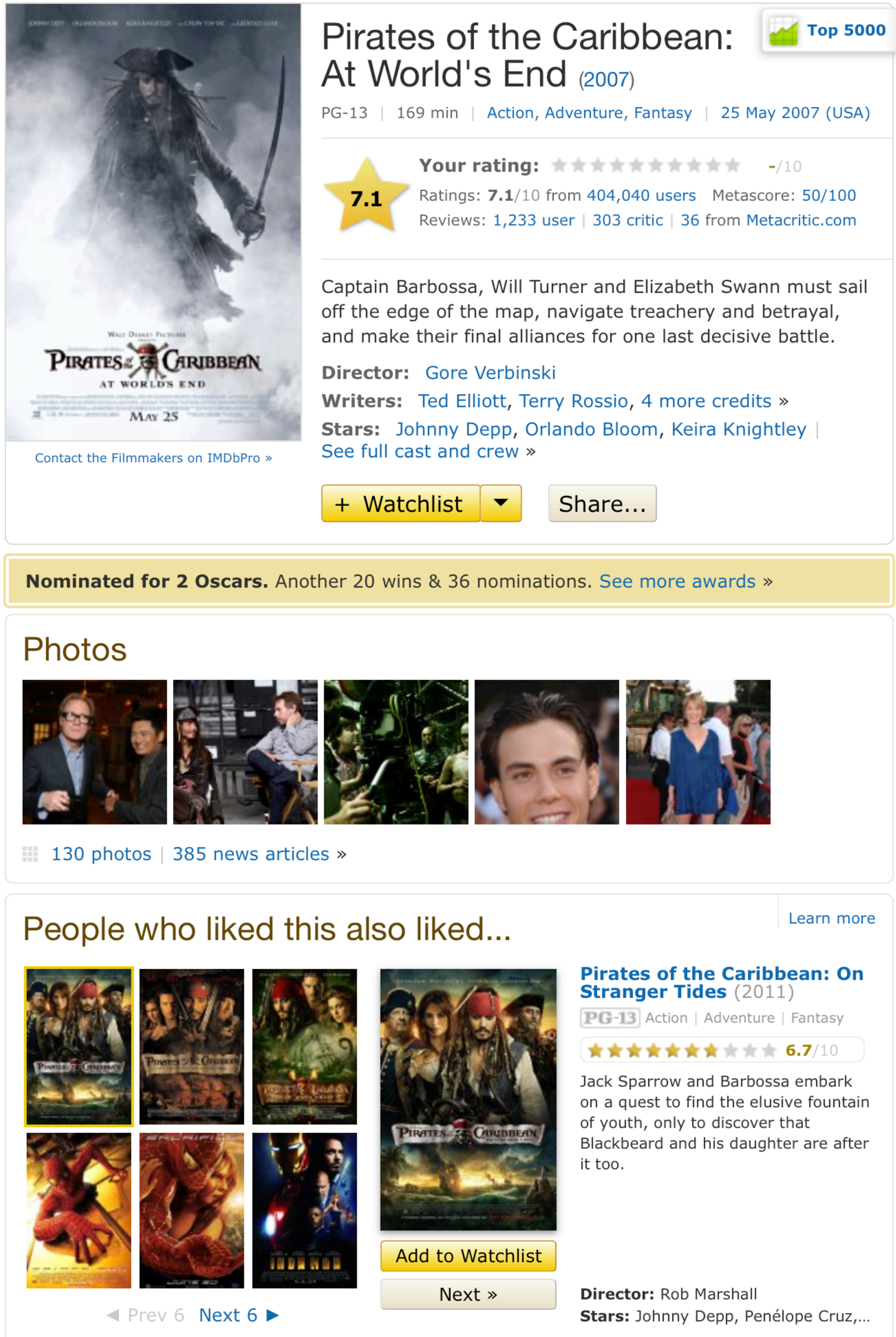
$$\Downarrow$$

$$\underbrace{A_k}_{m\times n} = \underbrace{U_k}_{m\times k}\ \underbrace{\Sigma_k}_{k\times k}\ \underbrace{V_k^T}_{k\times n}$$

Rank k

But this is still $m \times n$

Applications: Recommender Systems



- Finding **similar** movies
- Essential approach
  - A movie is represented by the viewers who watched / liked it
  - ***Similarity***: Viewers who liked this movie also liked some other movies
  - Recommendation: Since you are looking at this, you may as well look at those others…

Picture source: www.imdb.com

Application: recommender system

- Based on what factors do we like movies?
  - Actors, genre, plot, graphics, music, director, …
  - These choices are not strictly deterministic
  - May be arbitrary combinations of these
  - There may be similarities between two different actors, categories, etc
  - Too many dimensions

- Assumption: there are a few (much lesser than the number of all these different dimensions) factors which influence our choice

Approach: matrix factorization based collaborative filtering

- Assumption: there are a few factors which influence our choice

- Approach: we do not bother to know explicitly which factors. Let them remain hidden (latent)

- Setting: *m* users, *n* items, *k* latent factors essentially defining them
- Each item $\mathbf{y}_j$ is characterized by the *k* factors as
  $\mathbf{y}_j = [y_j^{(1)}, \ldots, y_j^{(k)}]$
- Each user $\mathbf{x}_i$ is characterized by his/her likings to these *k* factors as
  $\mathbf{x}_i = [x_i^{(1)}, \ldots, x_i^{(k)}]$

Latent factor model

- Items $\mathbf{y}_j = [y_j^{(1)}, \ldots, y_j^{(k)}]$, users $\mathbf{x}_i = [x_i^{(1)}, \ldots, x_i^{(k)}]$
- If this model is correct, then the rating user $\mathbf{x}_i$ should give to item $\mathbf{y}_j$ should correspond to the *inner-product* of the vectors $\mathbf{x}_i$ and $\mathbf{y}_j$

  $$r_{ij} = \mathbf{x}_i^T \mathbf{y}_j$$

- How do we use this model?
- We do not actually have the users and the items in our desired (latent) *k* dimensional space

Movie recommendation

- We have the user × items ($m \times n$) matrix
- With a lot of entries (most entries) missing
- According to our assumption, this matrix is

$$\underbrace{\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{x}_m \end{bmatrix}}_{m \times k} \underbrace{\begin{bmatrix} \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n \end{bmatrix}}_{k \times n}$$

Low rank (at most k)

Items

Users

-	5	-	5	-	-	3	-	-
1	-	2	-	2	-	-	1	-
-	-	-	-	-	4	3	-	5
-	5	-	5	-	-	-	4	-
4	-	5	-	-	4	3	-	-
-	-	-	-	4	-	-	5	-
-	1	2	-	-	-	3	-	-
2	-	-	4	-	-	-	2	-
-	-	1	-	-	2	-	-	1

Goal: estimate missing entries

Movie recommendation: approach

- Only a few users have rated an item. What is the average?
- **Step 1:** Compute the average of each column and fill the missing entries
- Do all users use the rating similarly?
  - Some users rate only when they are happy
  - Some users rate only when they are unhappy
  - Some users tend to give high ratings
  - Some users tend to give low ratings
- **Step 2:** Compute the average $\bar{a}_{ij}$ of each row and subtract that from the given ratings a_{ij}
  - Scale changes (centering)

Items

$$
\text{Users}\quad
\begin{matrix}
- & 5 & - & 5 & - & - & 3 & - & - \\
1 & - & 2 & - & 2 & - & - & 1 & - \\
- & - & - & - & - & 4 & 3 & - & 5 \\
- & 5 & - & 5 & - & - & - & 4 & - \\
4 & - & 5 & - & - & 4 & 3 & - & - \\
- & - & - & - & 4 & - & - & 5 & - \\
- & 1 & 2 & - & - & - & 3 & - & - \\
2 & - & - & 4 & - & - & - & 2 & - \\
- & - & 1 & - & - & 2 & - & - & 1
\end{matrix}
$$

matrix $A = (a_{ij})$

Movie recommendation

- Compute SVD of the ratings matrix A

  $A = U\Sigma V^T$

- Assume A is of rank k (<< m, n)

- Low rank approximation of A:

  $A_k = U_k\Sigma_k V_k^T = ((a_k)_{ij})_{m\times n}$

- Should capture the latent factors

- Now scale back the entries
- For each user, add back the row averages

  $\hat{a}_{ij} = (a_k)_{ij} + \bar{a}_{ij}$

- Many other variants exist

Random Projection vs SVD

- Different things to measure
- How much information is retained?
  - Retained variance: sum of the eigenvalues before and after mapping
- Performance in some task
  - Classification, IR, similarity measurement
- Efficiency (how fast)
- Feasibility

References and Acknowledgements

- General reference: “Mining of Massive Datasets” by Leskovec, Rajaraman and Ullman
- Other citations are given in the respective slides