

## Master on Foundations of Data Science



# Recommender Systems

Collaborative Recommender Systems: Factorization Models

Santi Seguí | 2023-2024

# **Model-Based CF methods: Factorization Models**

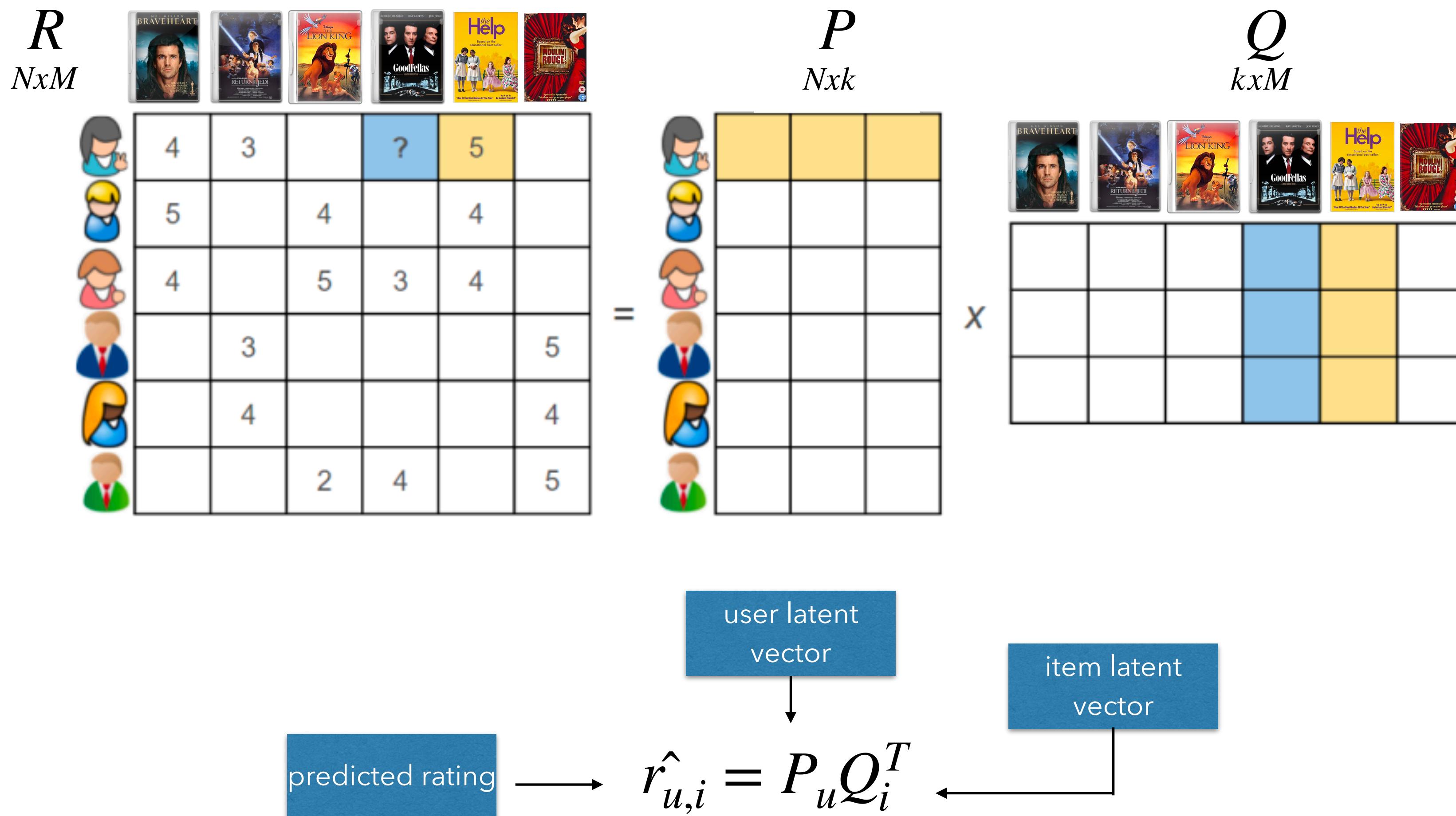
# Model-Based methods

Although **Neighborhood-based** models are very popular because its **simplicity**, today are not necessarily the most accurate ones.

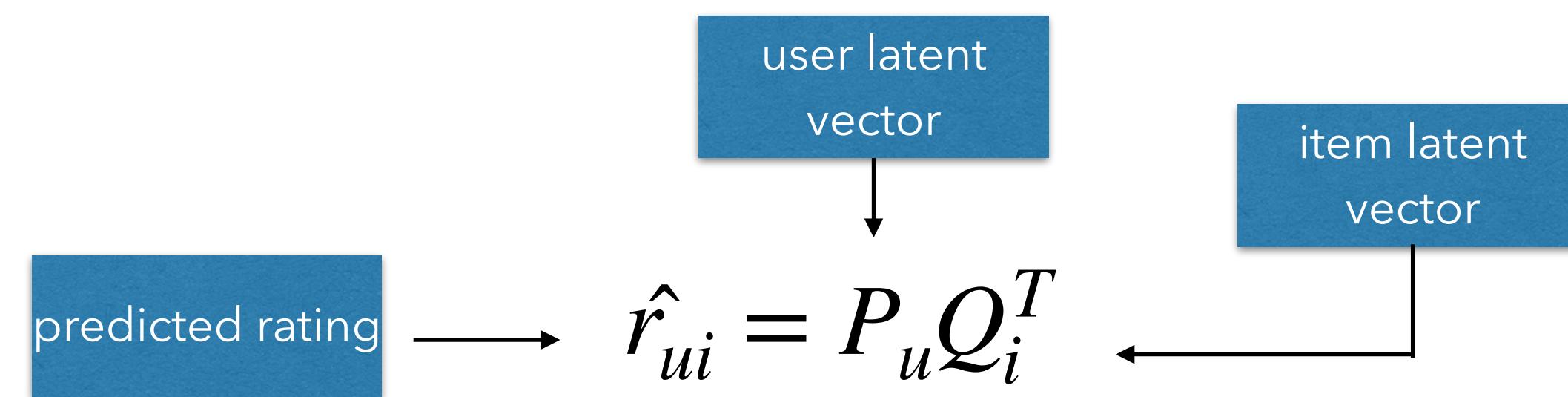
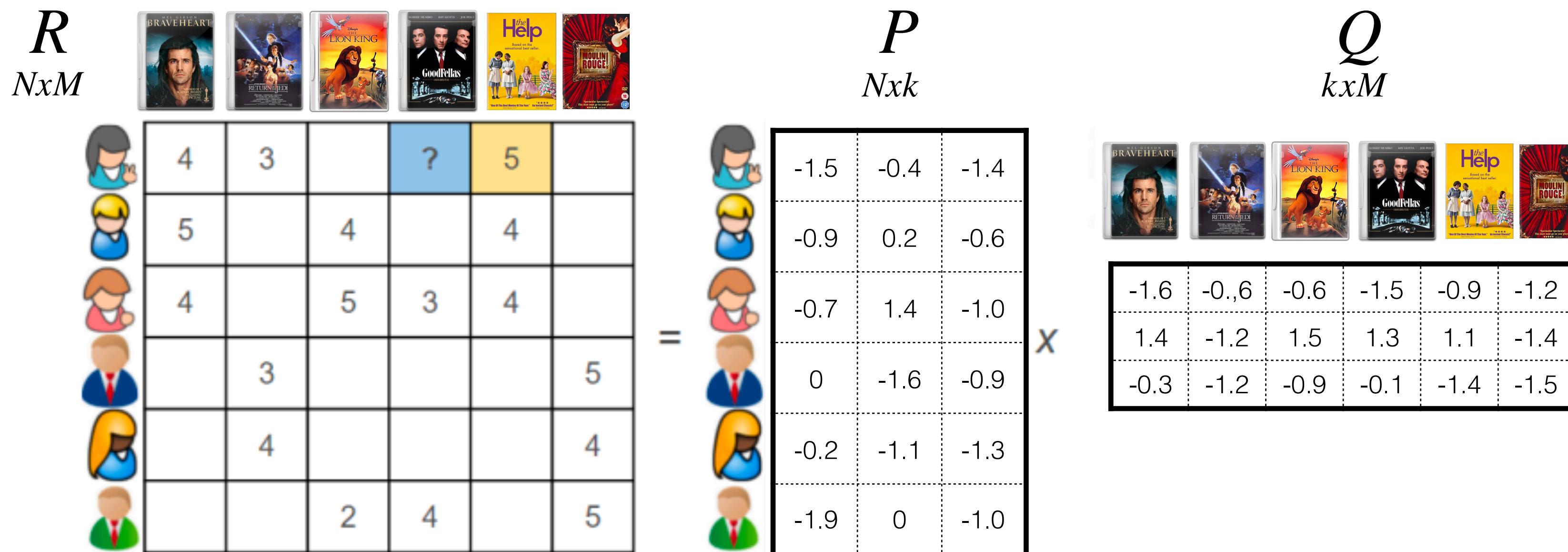
**Latent Factor Models** pose the recommendation problem as an **optimization problem**

- **GOAL:** Make good Recommendations
  - Quantify **goodness** using the **RMSE**.
  - **Lower the RMSE better** is our **recommender** system.
- **THE MOST COMMON BASELINE METHOD**

# Factorization Models



# Factorization Models



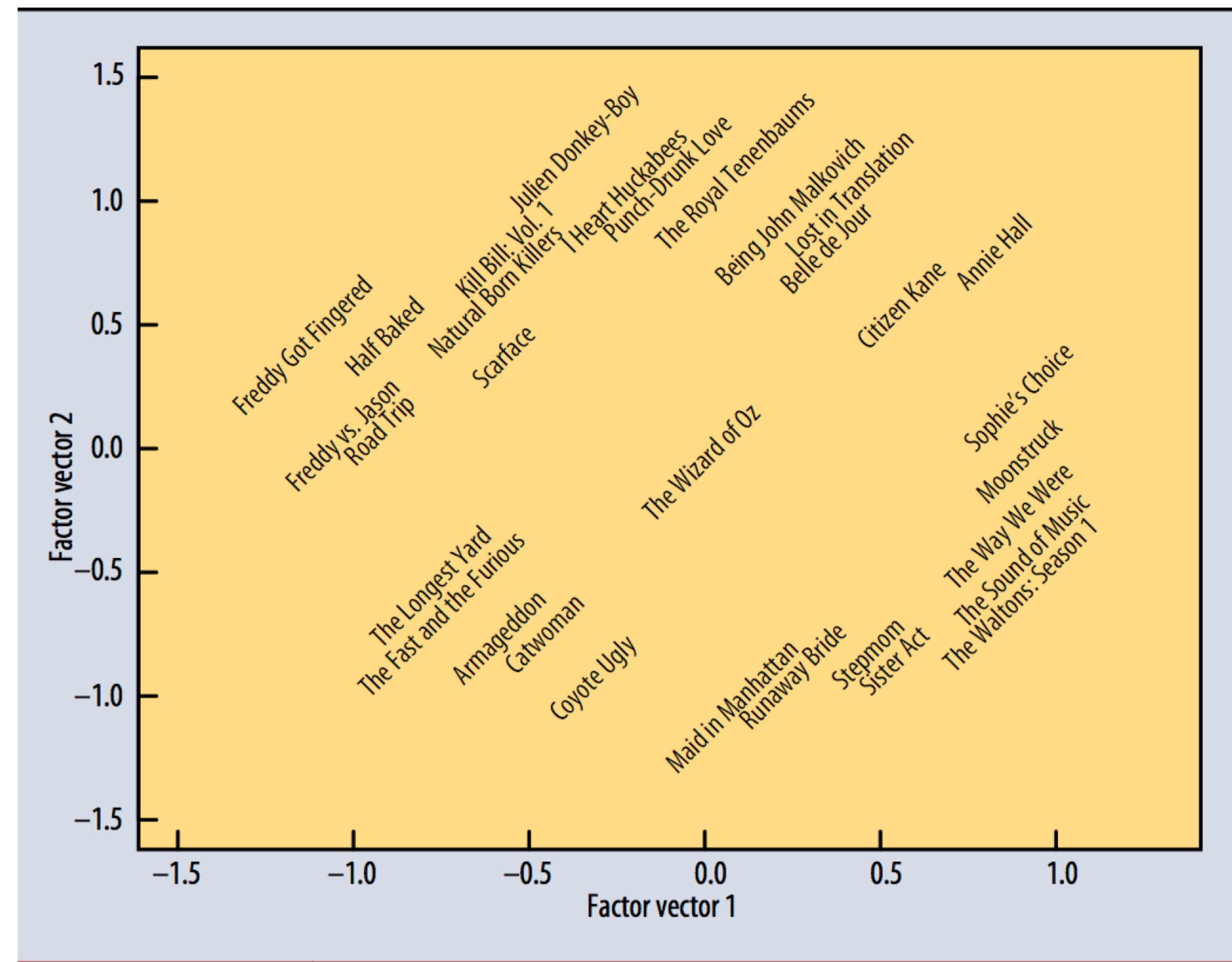


Figure 1: A mapping of movies based on two latent factors



# Factorization Models

- Latent factor models approach tries to explain the ratings by characterizing both items and users to a small number of factors inferred from the rating patterns.
- **PROBLEM:** Hard to optimize due to the huge amount of missing values

# Factorization Models

- Several variants of matrix factorization:
  - Singular Value Decomposition (SVD)
  - Vanilla Matrix Factorization
  - Matrix Factorization with biases
  - Non-Negative Matrix Factorization
  - SVD++
  - ...

# Singular Value Decomposition (SVD)

- Singular Value Decomposition (SVD) is a well established technique for identifying latent semantic factors. Done by factorizing the user-item rating matrix.
- Columns of U and V are constrained to be mutually orthogonal.
- Mutual orthogonality has the advantage that the concepts can be completely independent of one another. Can be interpreted in scatterplots

$$\hat{X} \approx U S V^T$$
$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots \\ \vdots & \vdots & \ddots \\ x_{m1} & & & x_{mn} \end{pmatrix}_{m \times n} \approx \begin{pmatrix} u_{11} & \dots & u_{1r} \\ \vdots & \ddots \\ u_{m1} & & u_{mr} \end{pmatrix}_{m \times r} \begin{pmatrix} s_{11} & 0 & \dots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix}_{r \times r} \begin{pmatrix} v_{11} & \dots & v_{1n} \\ \vdots & \ddots \\ v_{r1} & & v_{rn} \end{pmatrix}_{r \times n}$$

**PROBLEM:** Hard to  
optimize due to the huge  
amount of missing values

# Solution

- Fill missing values with 0
- Fill missing values with some statistics (e.g. user mean rating)
- Recursive methods

# Example of Singular Value Decomposition

$$R_f = \begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & ? & -1 & -1 & -1 \\ ? & 1 & 1 & -1 & -1 & ? \\ 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & ? & -1 & 1 & 1 & 1 \end{pmatrix}$$

The original Matrix

# Example of Singular Value Decomposition

$$R_f = \begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -0.2 & -1 & -1 & -1 \\ 0 & 1 & 1 & -1 & -1 & 0 \\ 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & 0.2 & -1 & 1 & 1 & 1 \end{pmatrix}$$

Step 1: Fill missing values with the mean value of the row

# Example of Singular Value Decomposition

$$R_f = \begin{pmatrix} 1.0592 & -1.1604 & 0.9716 & -0.8515 & 0.8040 & -1.0592 \\ 0.6636 & 0.9039 & 0.5881 & -0.9242 & -1.1244 & -0.6636 \\ 0.4300 & 0.9623 & 0.3746 & -0.6891 & -1.1045 & -0.4300 \\ -0.9425 & -0.8181 & -0.8412 & 1.2010 & 1.1320 & 0.9425 \\ -1.0290 & -0.2095 & -0.9270 & 1.1475 & 0.5535 & 1.0290 \end{pmatrix}$$

Step 2: Apply SVD to the matrix

# Example of Singular Value Decomposition

$$R_f = \begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & 0.5881 & -1 & -1 & -1 \\ 0.4300 & 1 & 1 & -1 & -1 & -0.43 \\ 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & -0.2095 & -1 & 1 & 1 & 1 \end{pmatrix}$$

Step 3. Modify the target matrix setting in the missing values the learn values and iterate until convergence

# Movie Recommender System using SVD



# Factorization Models

**Fill the missing values with some values**

- Hard to do
- Inaccurate filling can distort the data

**Modeling directly the observed rating only??**

# Modeling directly the observed rating only??

$$\min_{P \in \mathbb{R}^{m \times k}, Q \in \mathbb{R}^{n \times k}} \sum_{(i,j) \in obs} (r_{u,i} - p_u q_i^T)^2$$

# Modeling directly the observed rating only?

- We can minimize the objective function with **Stochastic gradient descent**:
  - For each given training set, the system predicts  $r_{u,i}$ , and computes the associated prediction error

$$e_{ui} \stackrel{\text{def}}{=} r_{ui} - q_i^T p_u.$$

- Then it modifies the parameters by a magnitude proportional to  $\gamma$  in the opposite direction of the gradient, yielding:

$$p_u = p_u - \gamma \frac{\partial f}{\partial p_u} = p_u + \gamma(e_{ui} q_i)$$

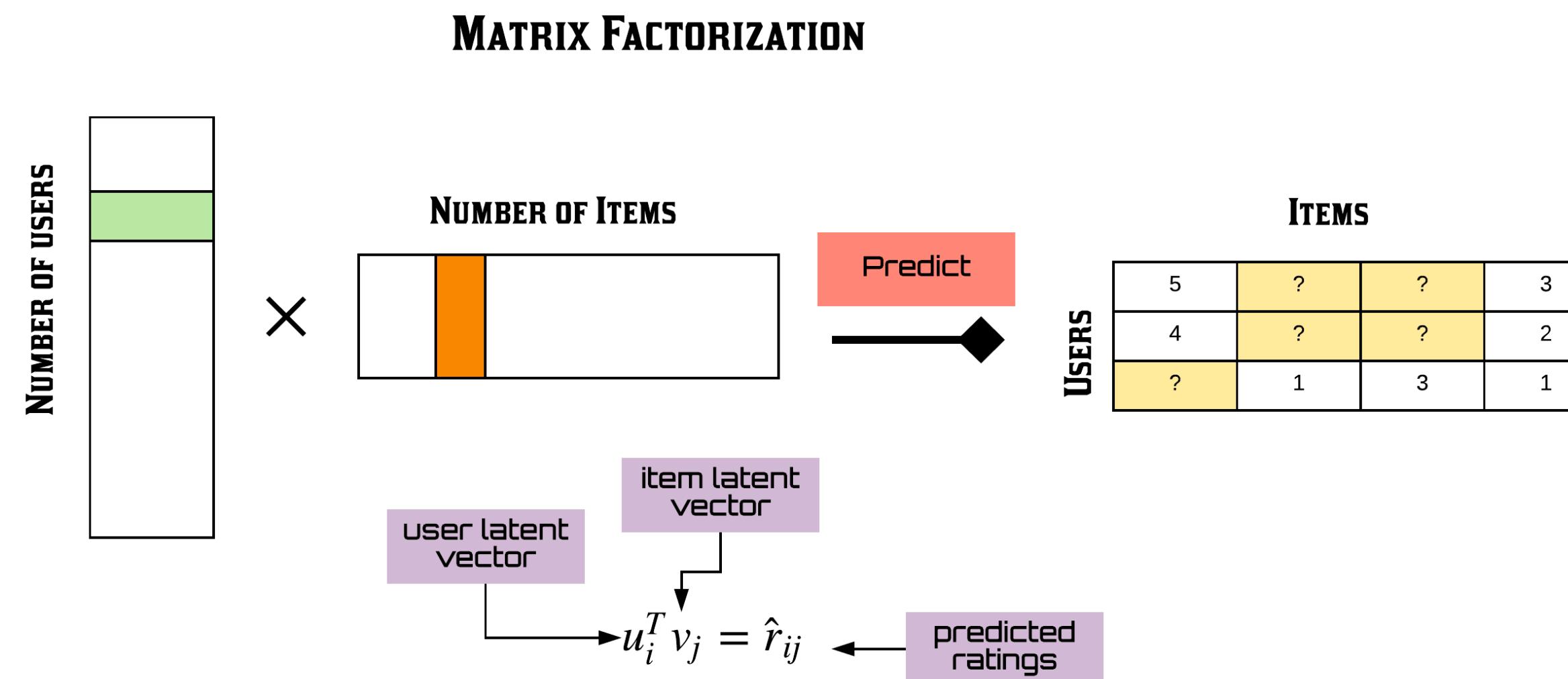
$$q_i = q_i - \gamma \frac{\partial f}{\partial q_i} = q_i + \gamma(e_{ui} p_u)$$

# 1) Vanilla Matrix Factorization

## SVD funk

A straightforward matrix factorization model maps both users and items to a joint latent factor space of dimensionality D — such that user-item interactions are modeled as inner products in that space.

- Each item  $i$  is associated with a vector  $q_i$ , and each user  $u$  is associated with a vector  $p_u$
- For a given item  $i$ ,  $q_i$  measures the interest that a user has for the latent factors.
- The resulting dot product ( $q_i \cdot p_u$ ) captures the interaction between user  $u$  and item  $i$ , which is the user's overall interest in the item's characteristics.
  - $\hat{r}_{ij} = q_i \cdot p_u$



# Movie Recommender System using Vanilla MF



# 2) Regularized Vanilla Matrix Factorization

$$\sum_{(i,j) \in obs} (r_{u,i} - q_i \cdot p_u)^2 + \text{Regularize}(q_i + p_i)$$

$$\sum_{(i,j) \in obs} (r_{u,i} - q_i \cdot p_u)^2 + \lambda(\|q_i\|^2 + \|p_i\|^2)$$

where  $\lambda$  controls the extent of regularization

- now, update rules are:

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

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

# 3) Regularized Matrix Factorization with biases

Adding **Bias**

$$P'_{ui} = b_u + b_i + (p_u^T q_i)$$

$b_u$  is observed deviation of user u;  
 $b_i$  is observed deviation of item i.

Improving regularized singular value decomposition for collaborative filtering  
A Paterek  
Proceedings of KDD Cup and Workshop 2007, 5-8

713

2007

# 4) SVD++

- An extension of previous method which also uses **implicit information**
  - **Explicit** (e.g. numerical rantings) + **Implicit** information (e.g. likes, purchases, skipped, bookmarked,...)

$$\hat{r}_{ui} = b_{ui} + q_i^T \left( p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right)$$

- $N(u)$  is the **set of items for which the user  $u$  has implicit information**
  - The user  $u$  is modeled as  $p_u + |N(u)|^{-1/2} \sum_{j \in N(u)} y_j$
  - $y_j$  is a **new item factor** term that captures the implicit ratings (e.g. if a user  $u$  has rated a item  $j$ , regardless the rating value)

Factorization meets the neighborhood: a multifaceted collaborative filtering model  
Y Koren  
Proceedings of the 14th ACM SIGKDD international conference on Knowledge ...

2098 2008



# Movie Recommender System

# 5) Non-Negative Matrix Factorization

- Can be used for ratings matrices that are non negative

$$\begin{aligned} \text{Minimize} \quad J &= \frac{1}{2} \|R - UV^T\| \\ \text{subject to:} \quad U &\geq 0, V \geq 0 \end{aligned}$$

- The major advantage of this approach is not the accuracy, but that of the **high level of interpretability** it provides in understanding the user-item interactions.

# 6) Sparse Linear Models (SLIM)

- Computes the item-item relations, by estimating an **item x item** sparse aggregation coefficient **matrix  $S$** .
- The recommendation score of an unrated item  $i$  for a user  $u$  is:

$$\hat{r}_{u,i} = r_u^T S_i$$

$$\begin{aligned} & \underset{S}{\text{minimize}} && \frac{1}{2} \sum_{u,i} (r_{ui} - \hat{r}_{ui})^2 + \frac{\beta}{2} \|S\|_F^2 + \lambda \|S\|_1, \\ & \text{subject to} && S \geq 0, \text{ and} \\ & && \text{diag}(S) = 0. \end{aligned}$$

SLIM: Sparse linear methods for top-n recommender systems

X Ning, G Karypis

Data Mining (ICDM), 2011 IEEE 11th International Conference on, 497-506

115

2011

# 6) Sparse Linear Models (SLIM)

$$\hat{R} = \begin{array}{c} i_1 & i_2 & i_3 & \dots & i_N \\ \hline u_1 & 0 & 2 & 0 & \dots & 2 \\ u_2 & ? & 3 & 2 & \dots & 0 \\ \hline \dots & \dots & \dots & \dots & \dots & 2 \\ \hline u_M & 0 & 4 & 2 & \dots & 2 \end{array} \times \begin{array}{c} i_1 & i_2 & i_3 & \dots & i_N \\ \hline i_1 & 0 & & \dots & \\ i_2 & & 0 & & \dots \\ i_3 & & & 0 & \dots \\ \hline \dots & \dots & \dots & 0 & \dots \\ \hline i_N & & & \dots & 0 \end{array}$$

$$\hat{r}_{u,i} = r_u^T S_i$$

$$\begin{aligned} & \underset{S}{\text{minimize}} && \frac{1}{2} \sum_{u,i} (r_{ui} - \hat{r}_{ui})^2 + \frac{\beta}{2} \|S\|_F^2 + \lambda \|S\|_1, \\ & \text{subject to} && S \geq 0, \text{ and} \\ & && \text{diag}(S) = 0. \end{aligned}$$

# 6) Sparse Linear Models (SLIM)

- Although the SLIM method proposes a prediction model for the rating, the final use of the ratings is for **ranking** the items
- Since, the weights are restricted to be positive, the impact of each weight on the score are **highly interpretable**
- Generally used for data sets with unary ratings (clicks, buy,... ) from **implicit feedback**
  - **Outperforms MF**

# Model-Based methods

- Advantages over neighborhood methods:
  - **Space-efficiency.** Usually, the learned model is much smaller than the original rating matrix
  - **Training speed and prediction speed.** Neighborhood-based models takes quadratic complexity in either number of users or number of items
  - **Avoiding overfitting.** The summarized model use to help in avoiding overfitting. Regularization methods can be used to make these models robust

# Minimizing the Objective Function

- Common algorithms to minimize the objective function include:
  - **Stochastic gradient descent (SGD)** is a generic method to minimize loss functions.
  - **Weighted Alternating Least Squares (WALS)** is specialized method to this particular objective.
    - WALS works by initializing the embeddings randomly, then alternating between:
      - Fixing  $P$  and solving for  $Q$
      - Fixing  $Q$  and solving for  $P$
    - This technique is guaranteed to converge because each step is guaranteed to decrease the loss.

# Minimizing the Objective Function

SGD and WALS have advantages and disadvantages

## SGD

- 👍 Very flexible—can use other loss functions.
- 👍 Can be parallelized.
- 👎 Slower—does not converge as quickly.
- 👎 Harder to handle the unobserved entries (need to use negative sampling or gravity).

## WALS

- 👎 Reliant on Loss Squares only.
- 👍 Can be parallelized.
- 👍 Converges faster than SGD.
- 👍 Easier to handle unobserved entries.