UFMG UNIVERSIDADE FEDERAL DE MINAS GERAIS

Recommender Systems

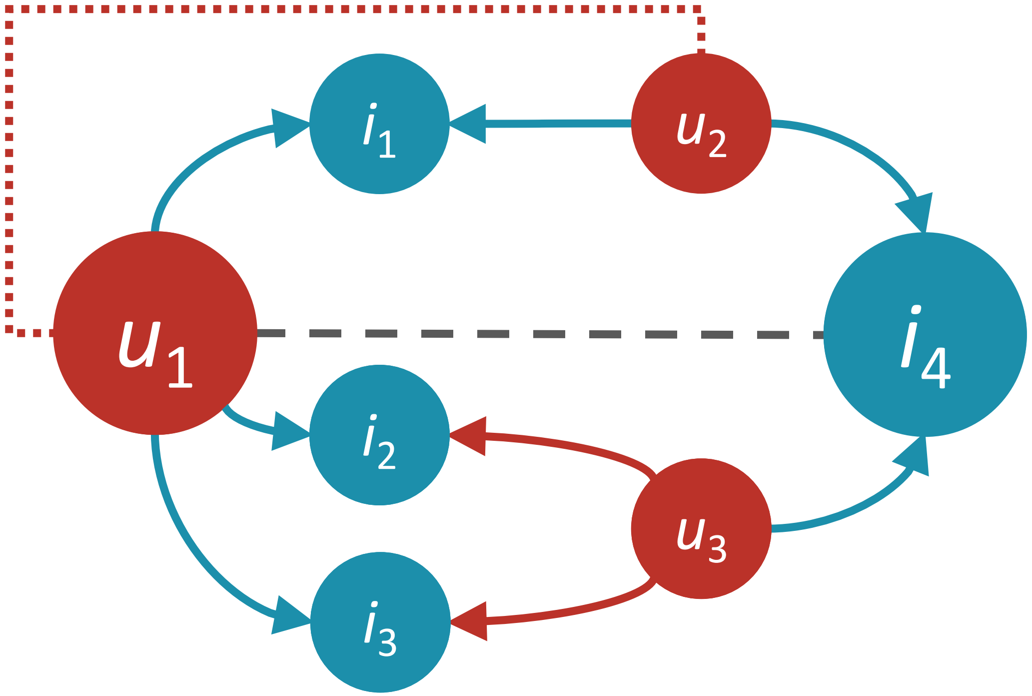
Matrix Factorization

Rodrygo L. T. Santos
rodrygo@dcc.ufmg.br

The recommendation problem

$$f(u,i)$$

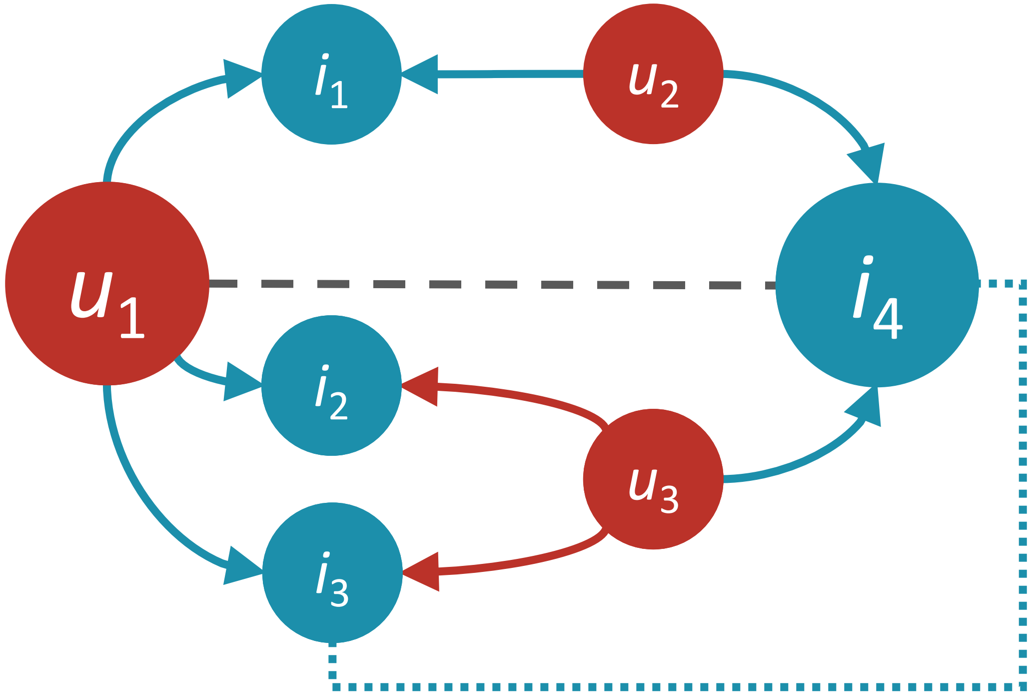
User-based collaborative filtering



similar users tend to
like the same items

recommend items liked
by users similar to u_1

Item-based collaborative filtering



users who like an item
tend to like similar items
recommend items similar
to those consumed by u_1

Memory-based collaborative filtering

How will user $\boldsymbol{u}$ like item $\boldsymbol{i}$?

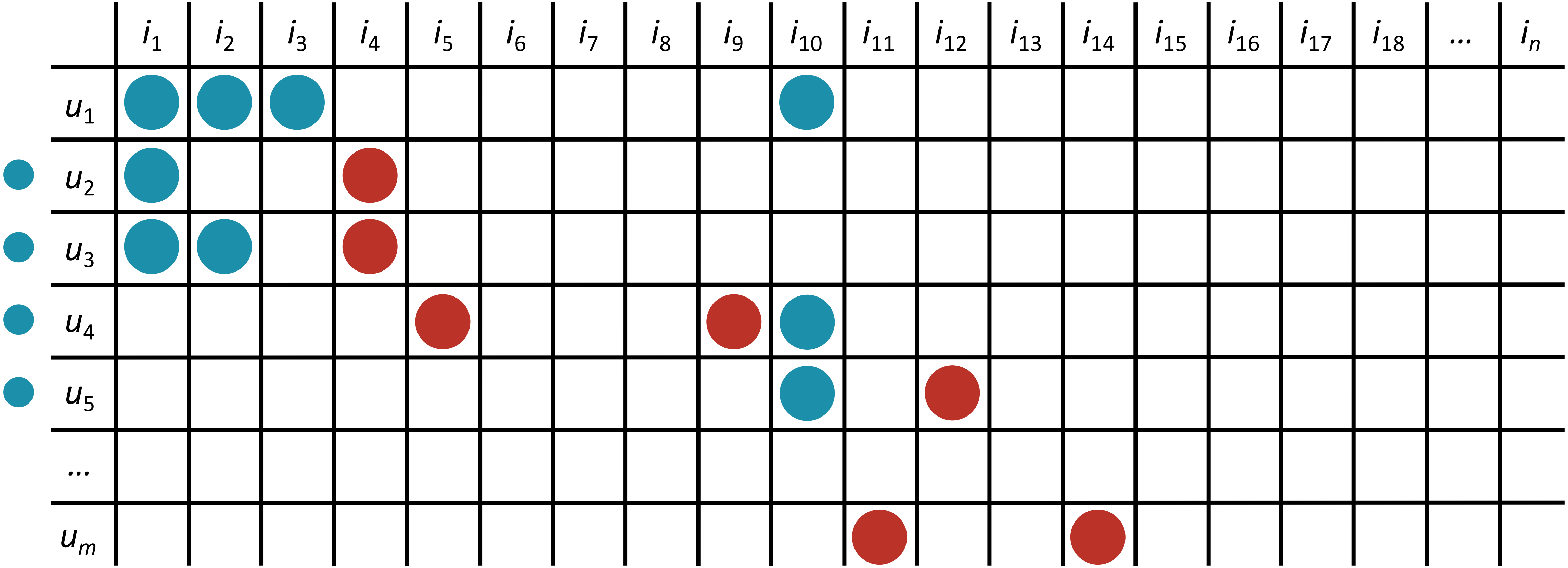
- User-based: *how do users similar to* $\boldsymbol{u}$ *like* $\boldsymbol{i}$*?*
- Item-based: *how do* $\boldsymbol{u}$ *like items similar to* $\boldsymbol{i}$*?*

Key difference: **neighborhoods**

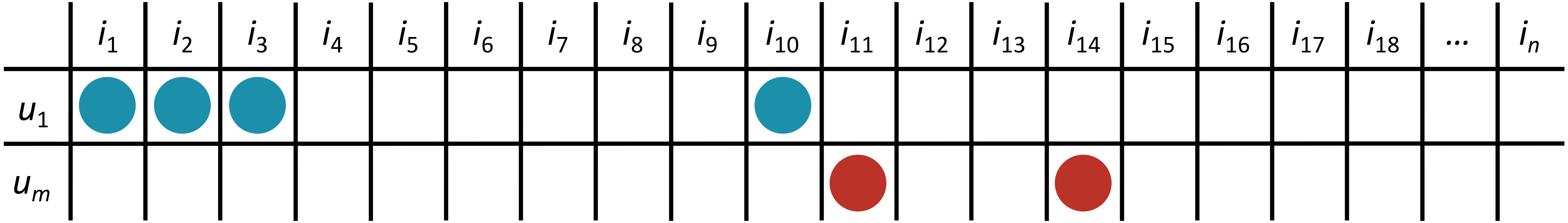
- User-based: **unstable**, hard to precompute
- Item-based: **stable**, easy to precompute

How to handle high dimensions?

User-based collaborative filtering



User-based collaborative filtering



are they neighbors?

User-based collaborative filtering

Memorize or model?

Memory-based

- Online "learning"
- Online prediction
- Lazy, instance-based learning

Costly recommendation

- $O(mn)$ worst case

Poor robustness

- Sparsity, perturbations

Overfit representation

- I like Star Wars, you like Star Trek
- Are we neighbors?

Memorize or model?

Memory-based

- Online “learning”
- Online prediction
- Lazy, instance-based learning

Model-based

- Offline learning
- Online prediction

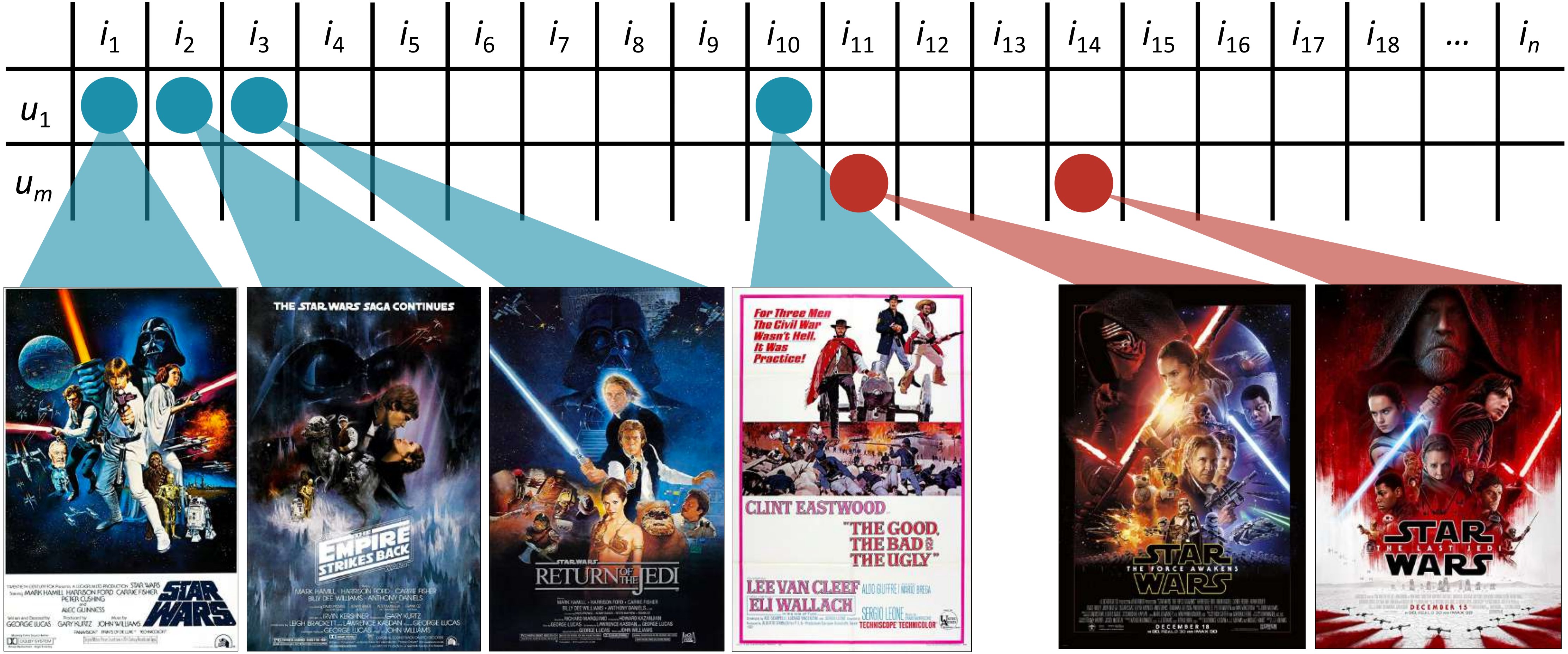
Dimensionality reduction

Distinct spaces in neighborhood models

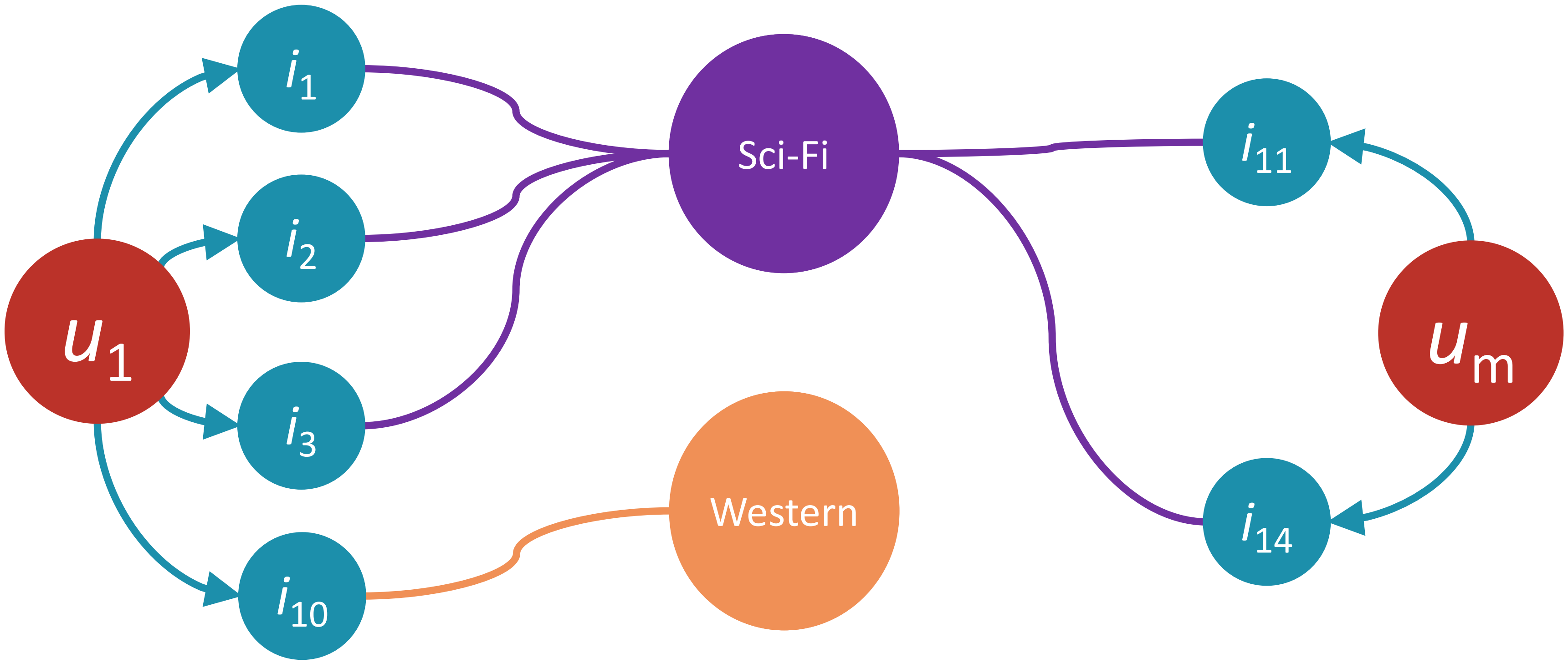
- Users as n-dimensional vectors over items
- Items as m-dimensional vectors over users
  - m and n could be in the hundreds of millions

Can we reduce the dimensions of the utility matrix while effectively retaining preference information?

Latent factor models



Latent factor models



Latent factor models

Exact solutions

- Matrix decomposition

Approximate solutions

- Regularized empirical risk minimization
- Pointwise, pairwise, listwise losses
- Negative sampling

Latent factor models

Distinct spaces in neighborhood models

- Users as n-dimensional vectors over items
- Items as m-dimensional vectors over users

Unified space in latent factor models

- Users and items as k-dimensional vectors
- Straightforward predictions via dot products

Singular value decomposition

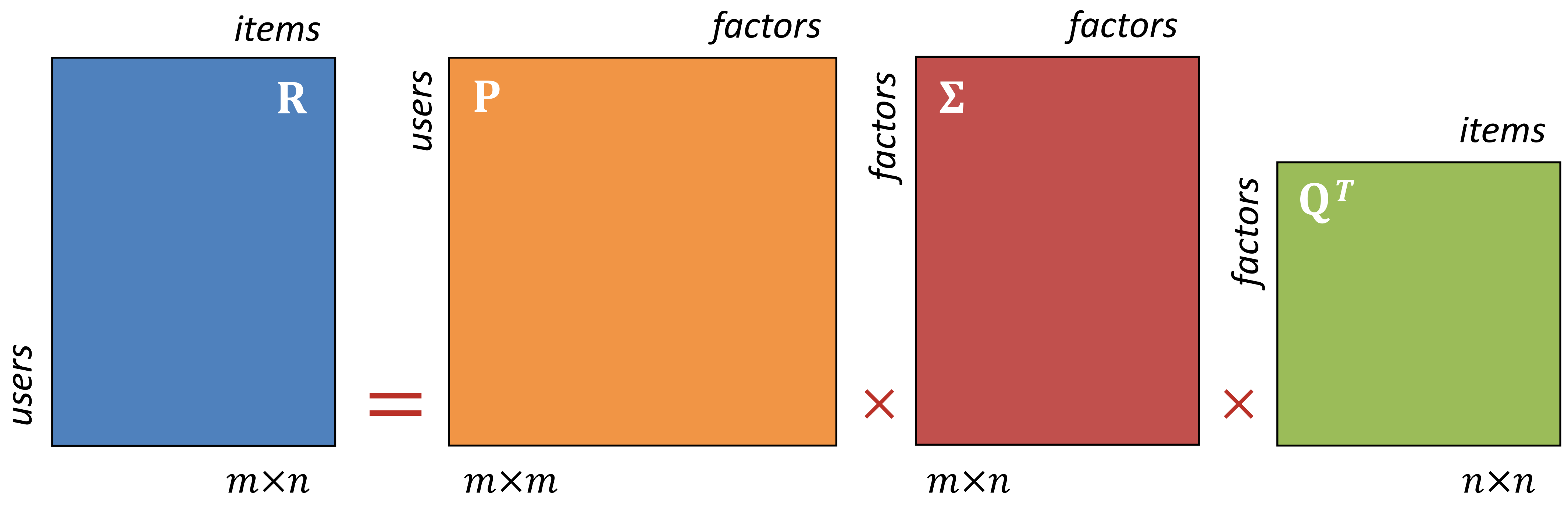
Reduce dimensionality of the problem

- Results in small, fast model
- Richer, denser neighbor network

One of various matrix factorization techniques

- Techniques based on eigenvalues (e.g., SVD)
- Techniques based on linear systems (e.g., LU)

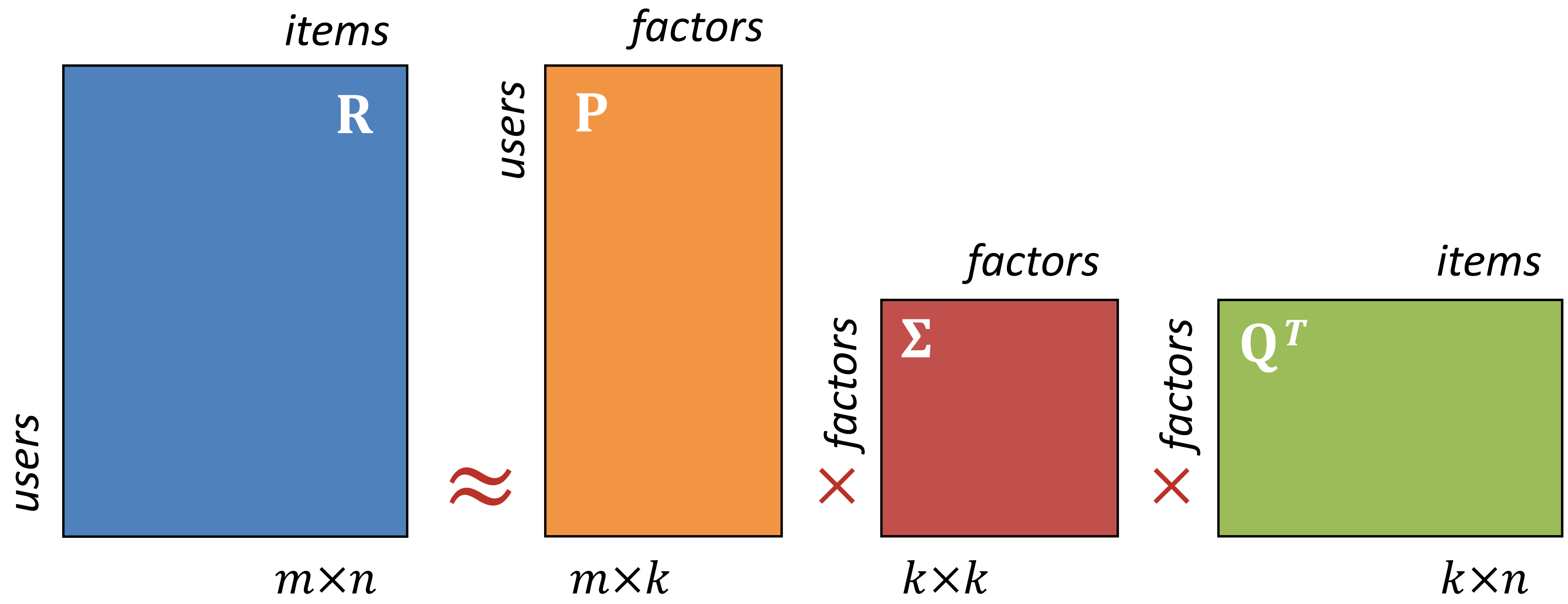
Factorization with SVD



Diagonal entries of **Σ** are singular values

(analogous of eigenvalues for non-square matrices)

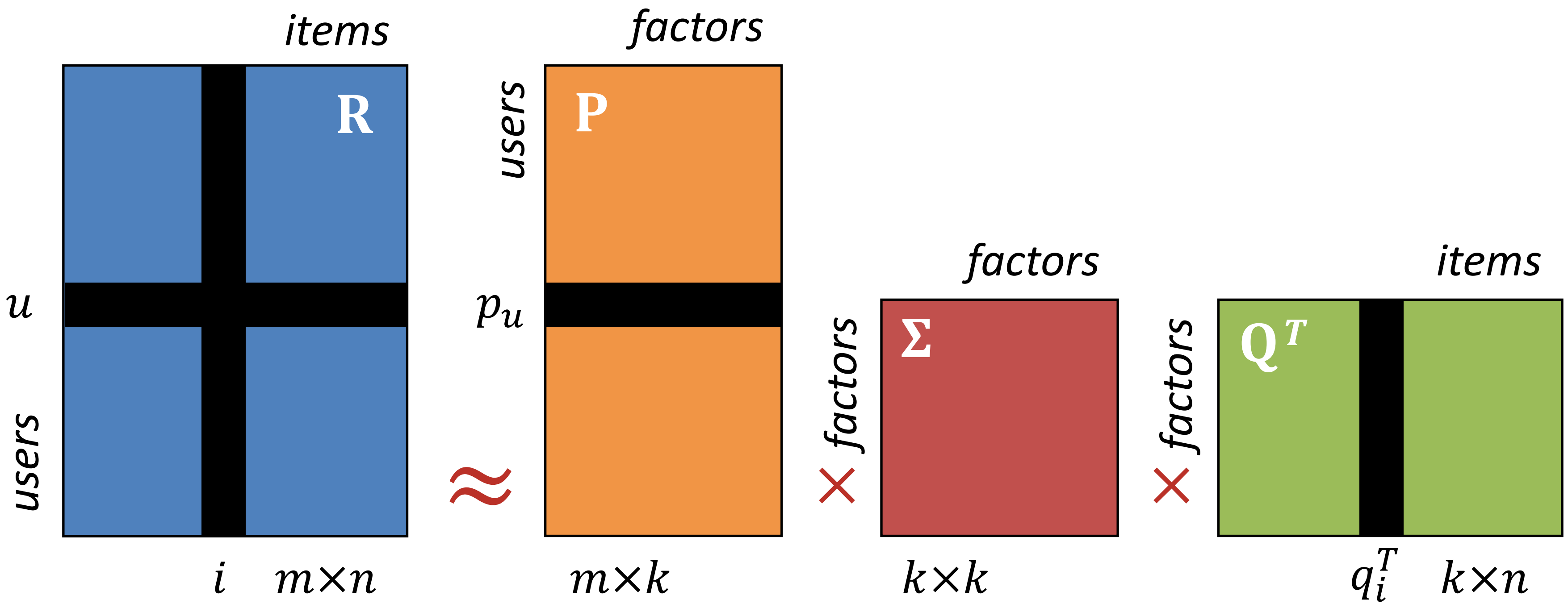
Factorization with SVD



“Truncated” SVD: keep k “most important” factors

(best rank-k approximation by Frobenius norm)

Predictions with SVD



$$\hat{r}_{ui} = p_u q_i^T = \sum_{f=1}^{k} p_{uf} q_{if}$$

SVD pros

Prediction quality generally increases…

- Noisy ratings filtered out
- Nontrivial correlations detected

Although it may also decrease

- Item-specific preferences no longer considered

SVD cons

Lack of transparency

- Latent factors hard to interpret

Missing values

- SVD is undefined for incomplete matrices

Computation complexity

- SVD computation is $O(m^2n + n^3)$

Missing values

SVD assumes a complete matrix

- If it’s complete, we don’t need a recommender!

What to do with missing values?

- Impute (assume they are a mean)
- Ignore!

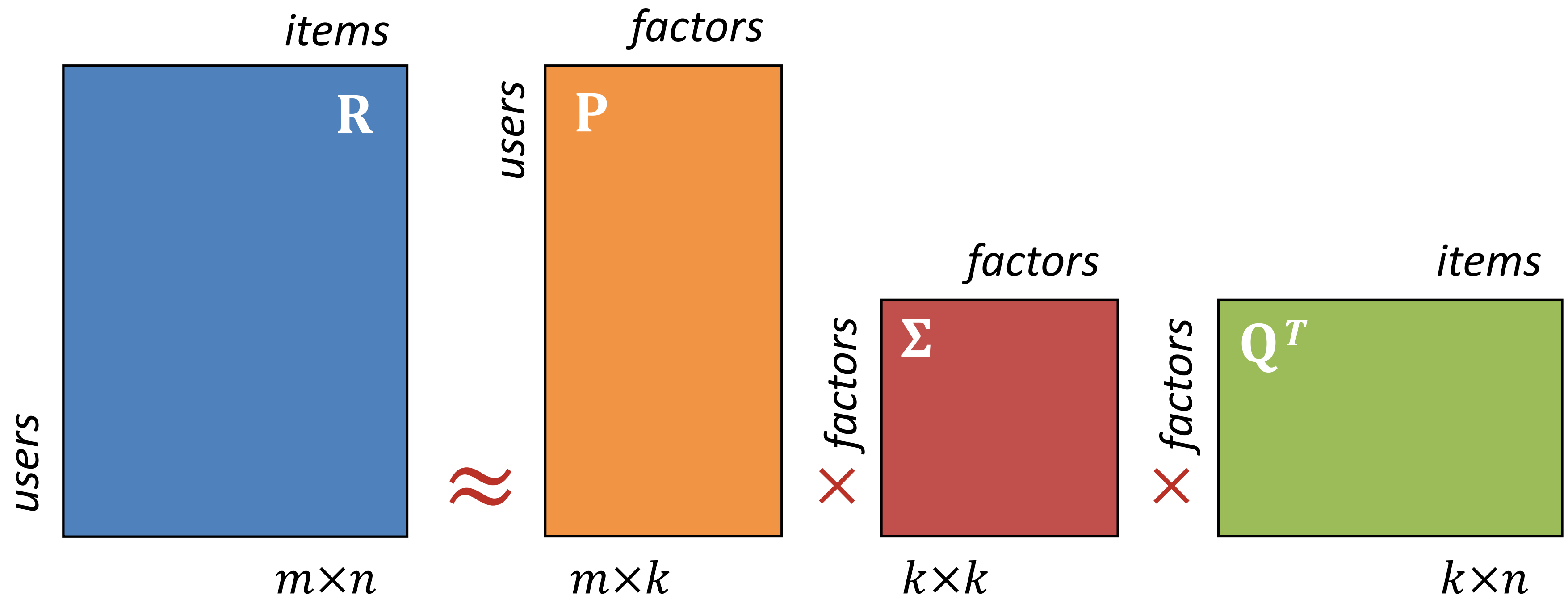
Computational complexity

Standard SVD is (very) slow

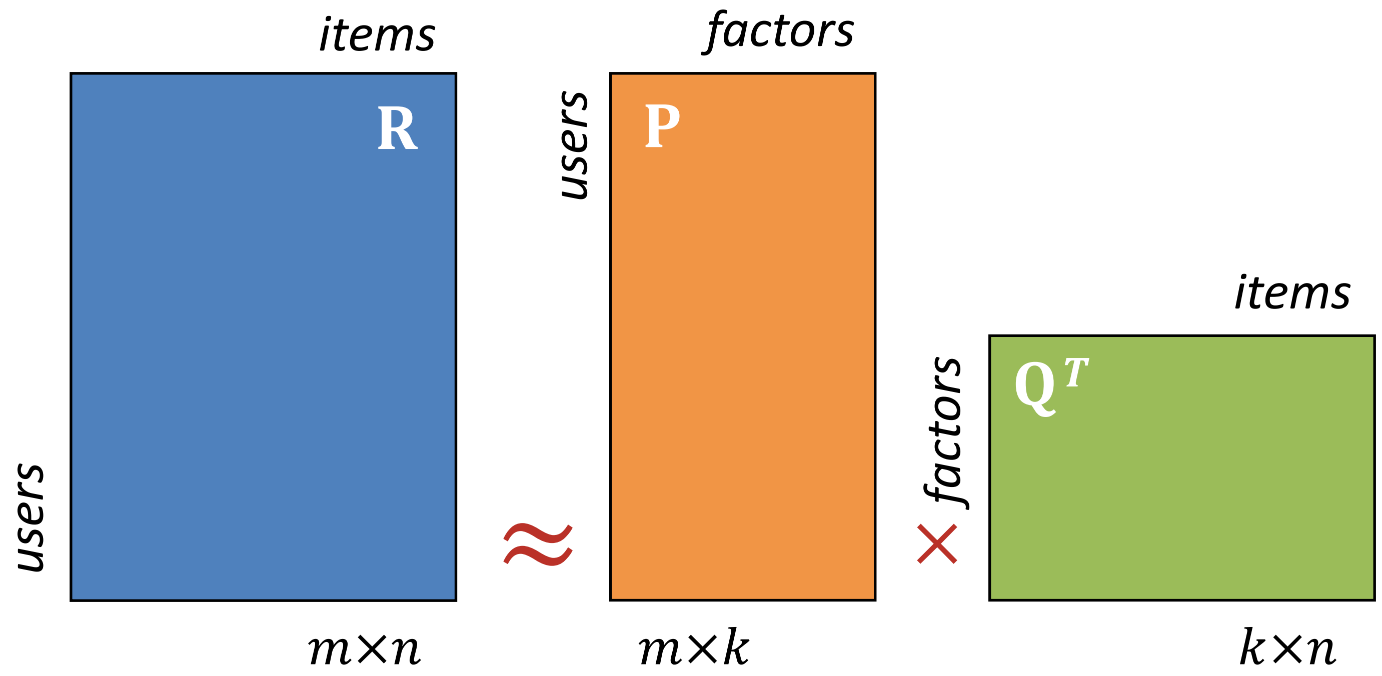
- All we need is a good approximation

Model it as an optimization problem!

Computational complexity



Computational complexity



Goal is to reconstruct the left matrix

- *Can we learn* $\mathbf{P}$ *and* $\mathbf{Q}$ *to minimize the reconstruction error?*

Credit: The Evolution of Cybernetics

Meanwhile during the Netflix prize…

❝ *Ok, so here's where I tell all about how I got to be tied for third place on the Netflix prize.*

- Simon Funk, 2006

Quantifying the error

Reconstruction error e_{ui} (aka residual)

- $e_{ui} = r_{ui} - \hat{r}_{ui}$

Given

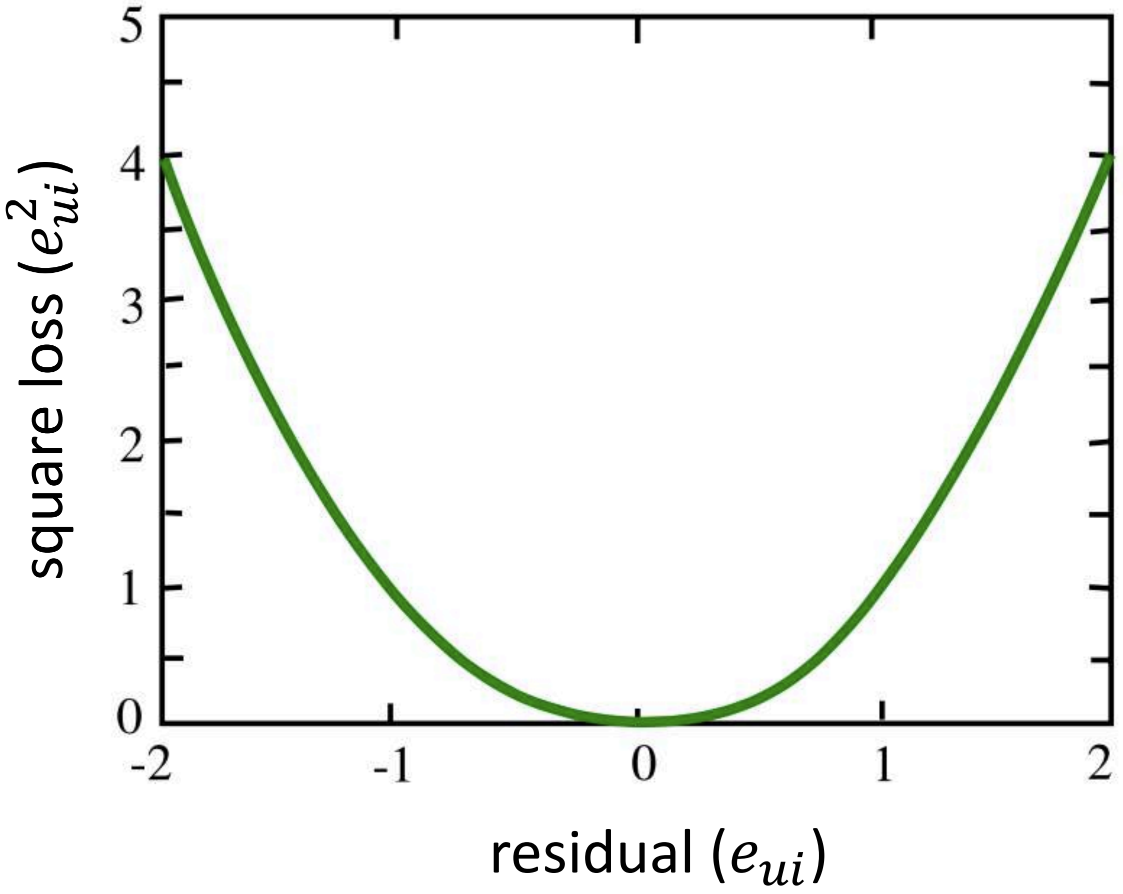
- Actual rating r_{ui}
- Predicted rating $\hat{r}_{ui} = p_u q_i^T = \sum_{f=1}^{k} p_{uf} q_{if}$

How to minimize the reconstruction error?

Example: square loss

- $e_{ui}^2 = (r_{ui} - \hat{r}_{ui})^2$
  $= \left(r_{ui} - \sum_{f=1}^{k} p_{uf} q_{if}\right)^2$

What configuration of P and Q give the minimum loss?



Gradient-based optimization

Reconstruction error

- $e_{ui}^2 = (r_{ui} - \hat{r}_{ui})^2 = \left(r_{ui} - \sum_{f=1}^{k} p_{uf} q_{if}\right)^2$

How to minimize the reconstruction error?

- ∇e_{ui}^2 points toward the greatest increase in e_{ui}^2

$$\nabla e_{ui}^2 = \left(\frac{\partial e_{ui}^2}{\partial p_{u1}}, \dots, \frac{\partial e_{ui}^2}{\partial p_{uk}}, \frac{\partial e_{ui}^2}{\partial q_{i1}}, \dots, \frac{\partial e_{ui}^2}{\partial q_{ik}}\right)$$

Gradient-based optimization

Reconstruction error

- $e_{ui}^2 = (r_{ui} - \hat{r}_{ui})^2 = \left(r_{ui} - \sum_{f=1}^{k} p_{uf} q_{if}\right)^2$

Taking partial derivatives w.r.t. each factor f

- $\frac{\partial e_{ui}^2}{\partial p_{uf}} = -2\left(r_{ui} - \sum_{f=1}^{k} p_{uf} q_{if}\right)(q_{if}) = -2e_{ui}q_{if}$
- $\frac{\partial e_{ui}^2}{\partial q_{if}} = -2\left(r_{ui} - \sum_{f=1}^{k} p_{uf} q_{if}\right)(p_{uf}) = -2e_{ui}p_{uf}$

Gradient-based optimization

Update rules (with learning rate α)

- $p_{uf} = p_{uf} - \alpha \frac{\partial e_{ui}^2}{\partial p_{uf}}$

  $= p_{uf} + 2\alpha e_{ui} q_{if}$

- $q_{if} = q_{if} - \alpha \frac{\partial e_{ui}^2}{\partial q_{if}}$

  $= q_{if} + 2\,\alpha e_{ui} p_{uf}$

Should we minimize the loss over all $\langle u, i \rangle$ pairs?

Gradient-based optimization

Minimize the loss over known entries only

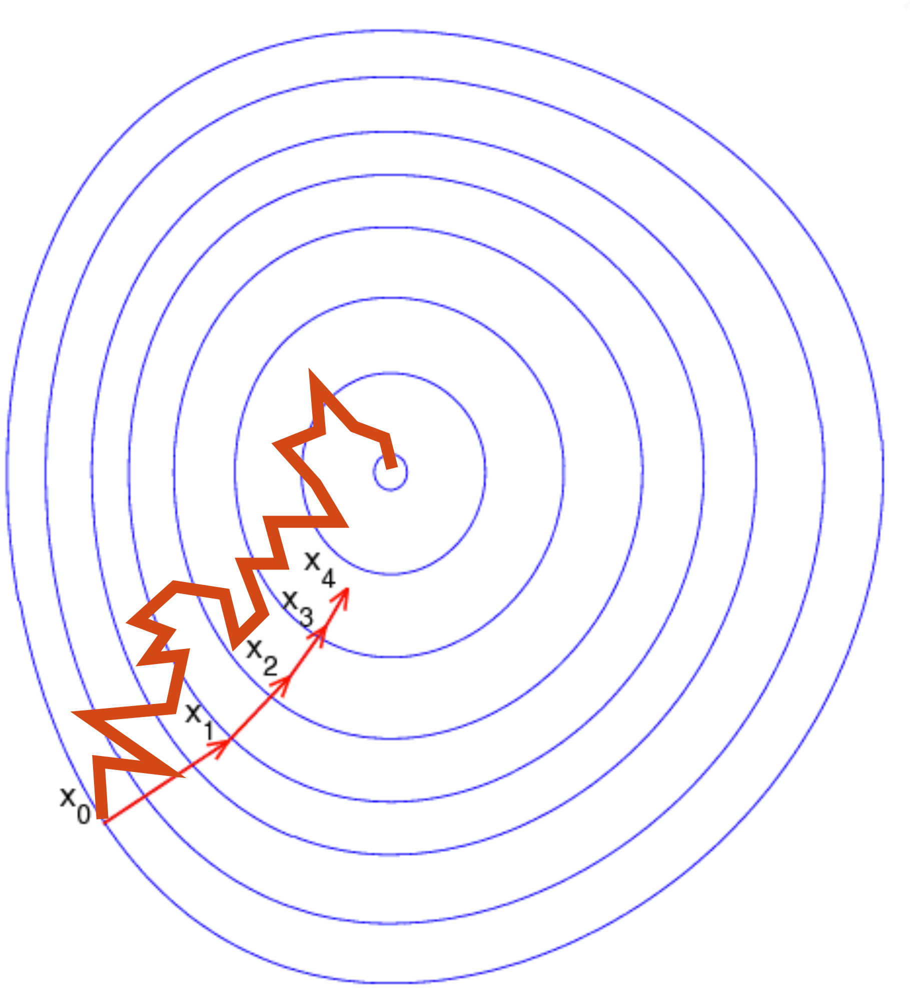
- $S = \{r_{ui} \in \mathbf{R} \mid r_{ui} > 0\}$

Take b entries from S at a time

- $b = |S|$: batch gradient descent
- $b = c \ll |S|$: mini-batch gradient descent
- $b = 1$: stochastic gradient descent

Stochastic gradient descent

SGD(S)
    init $\mathbf{P},\mathbf{Q}$
    repeat
        for each $r_{ui} \in S$
            $e_{ui} = r_{ui} - \sum_{f=1}^{k} p_{uf} q_{if}$
            $p_{uf} = p_{uf} + 2\alpha e_{ui} q_{if} \; \forall f = 1 \ldots k$
            $q_{if} = q_{if} + 2\alpha e_{ui} p_{uf} \; \forall f = 1 \ldots k$
    until converged
    return $\mathbf{P},\mathbf{Q}$



What could go wrong w/ minimizing over known ratings?

Overfitting

Matrix $\mathbf{R}$ is typically extremely sparse

◦ Learning from few examples may lead to overfitting

Solution: regularization (reg. parameter λ)

◦ $e_{ui}^2 = \left(r_{ui} - p_u q_i^T\right) + \lambda(\|p_u\|_2^2 + (\left\|q_i^T\right\|_2^2)$

A simplified example

Goal is to decompose the left-side matrix

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \\ u_{41} & u_{42} \\ u_{51} & u_{52} \end{bmatrix} \times \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \end{bmatrix}$$

A simplified example

An initial (random) guess

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

A simplified example

Not that impressive…

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

RMSE = 1.806

A simplified example

An improved guess

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} x & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

A simplified example

An improved guess

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} x+1 & x+1 & x+1 & x+1 & x+1 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Contribution of the first row

$$(5-(x+1))^2+(2-(x+1))^2+(4-(x+1))^2$$
$$+(4-(x+1))^2+(3-(x+1))^2$$

A simplified example

An improved guess

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} x+1 & x+1 & x+1 & x+1 & x+1 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

How to choose x to minimize the error?

- Take the gradient!

A simplified example

The error contribution

- $(5-(x+1))^2+(2-(x+1))^2+(4-(x+1))^2$
  $+(4-(x+1))^2+(3-(x+1))^2$

Simplifies to

- $(4-x)^2+(1-x)^2+(3-x)^2+(3-x)^2+(2-x)^2$

Taking the derivative and equating to 0

- $-2\times((4-x)+(1-x)+(3-x)+(3-x)+(2-x))=0$
  $\therefore x=2.6$

A simplified example

Replacing x = 2.6

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} x+1 & x+1 & x+1 & x+1 & x+1 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

A simplified example

Replacing x = 2.6

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} 3.6 & 3.6 & 3.6 & 3.6 & 3.6 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

RMSE ***reduced*** from 1.806 to 1.642

Summary

Matrix factorization is dominant

- Superior to classical nearest neighbor techniques

Matrix factorization is flexible (Koren et al., Computer 2009)

- Can integrate multiple forms of feedback, user and item biases, temporal dynamics, confidence levels

SVD is not the only factorization approach

References

Recommender Systems: An Introduction (Sec. 2.4)

Recommender Systems Handbook (Ch. 3)

Recommender Systems: The Textbook (Sec. 3.6)