Recommender systems

Matrix Factorization models

Lecture 3
2025

Anna Volodkevich

Contents

Recap

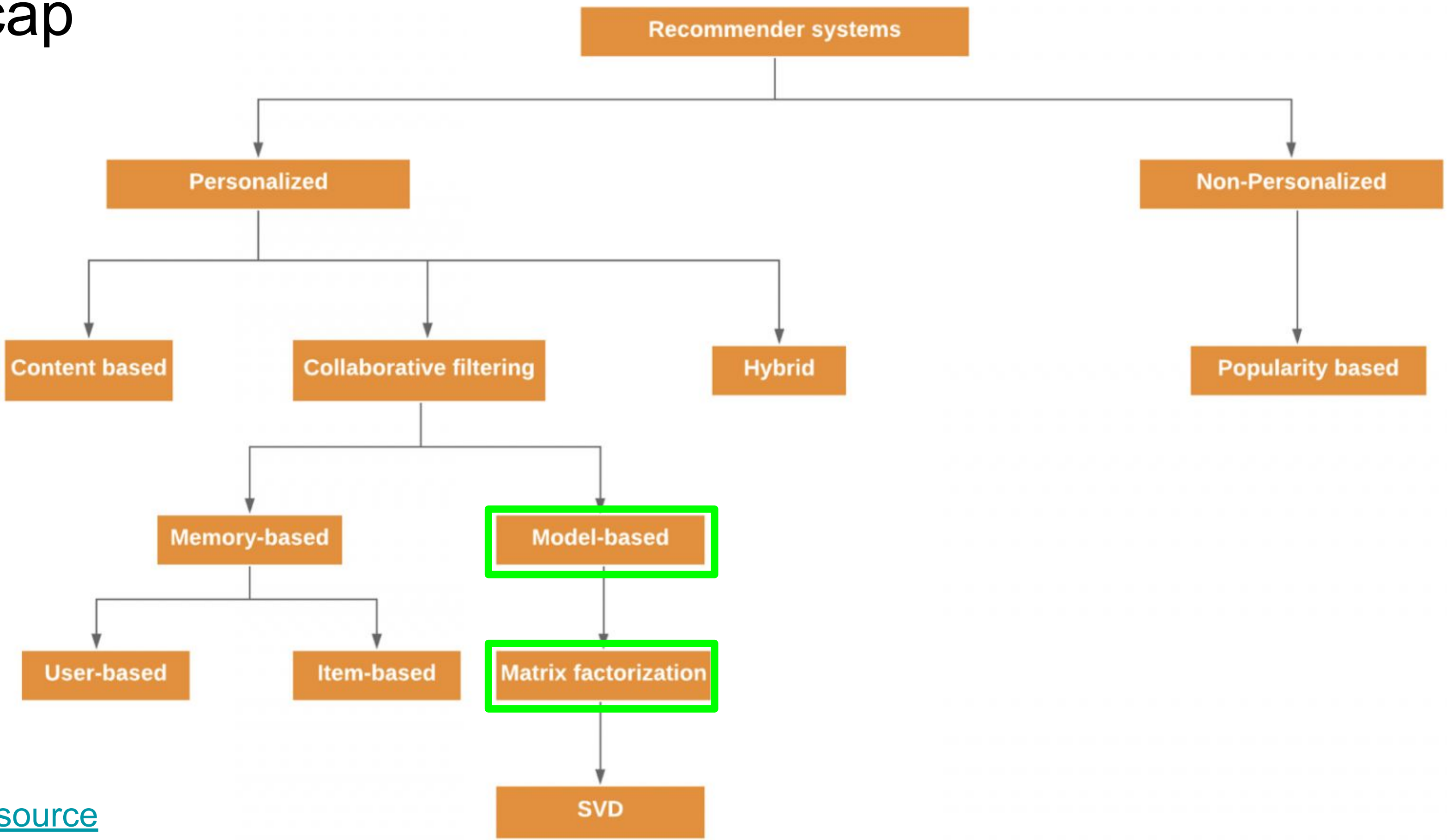


Image source

Recap

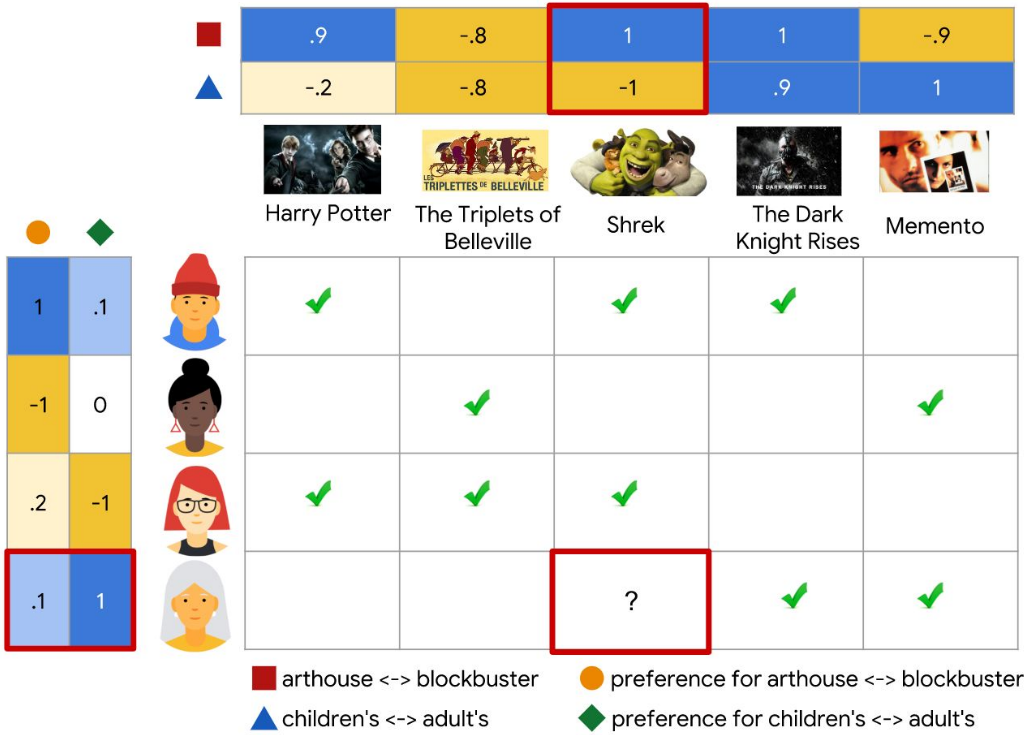
Model-based methods

In model-based methods, machine learning and data mining methods are used in the context of predictive models. In cases where the model is parameterized, the parameters of this model are learned within the context of an optimization framework. Some examples of such model-based methods include decision trees, rule-based models, Bayesian methods and **latent factor models**. Many of these methods, such as latent factor models, have a high level of coverage even for sparse ratings matrices.

Latent factor models, such as matrix factorization, transform both items and users to the same latent factor space. The latent space tries to explain ratings by characterizing both products and users on factors automatically inferred from user feedback.

Recommendations as a matrix completion

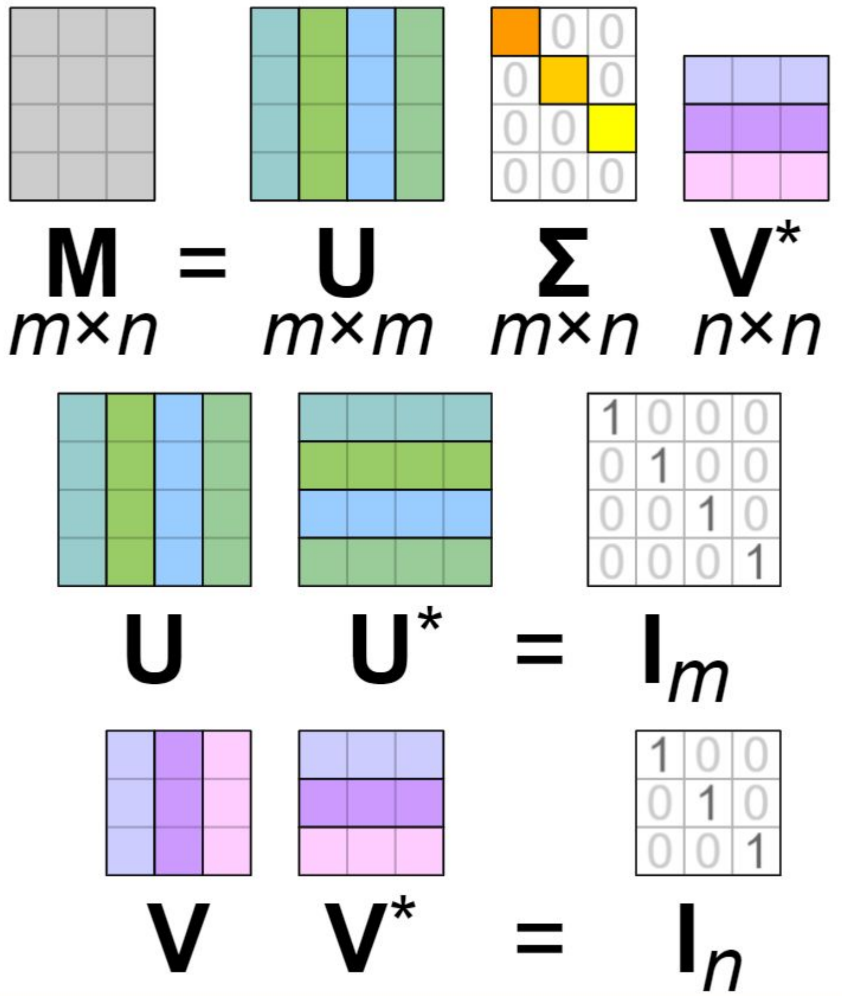
- User-item interactions are represented in a matrix form
- We assume that the users and items could be represented in a low-dimensional latent feature space
- MNAR problem
- Need to consider different approaches for different feedback types



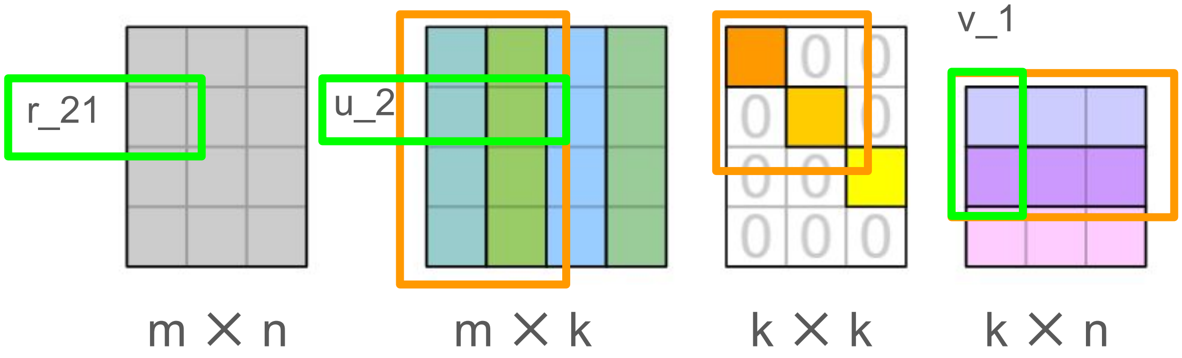
[Image source](#)

SVD: recap

- Σ - diagonal m×n matrix of non-increasing singular values $\sigma_1, \ldots, \sigma_n$
- U и V – orthogonal matrices left-singular vectors and right-singular vectors



Truncated SVD



k << n, k << m

$$||R - U_k \Sigma_k V_k^t||_F^2 \to \min$$

$$R \approx U_k \Sigma_k V_k^T$$

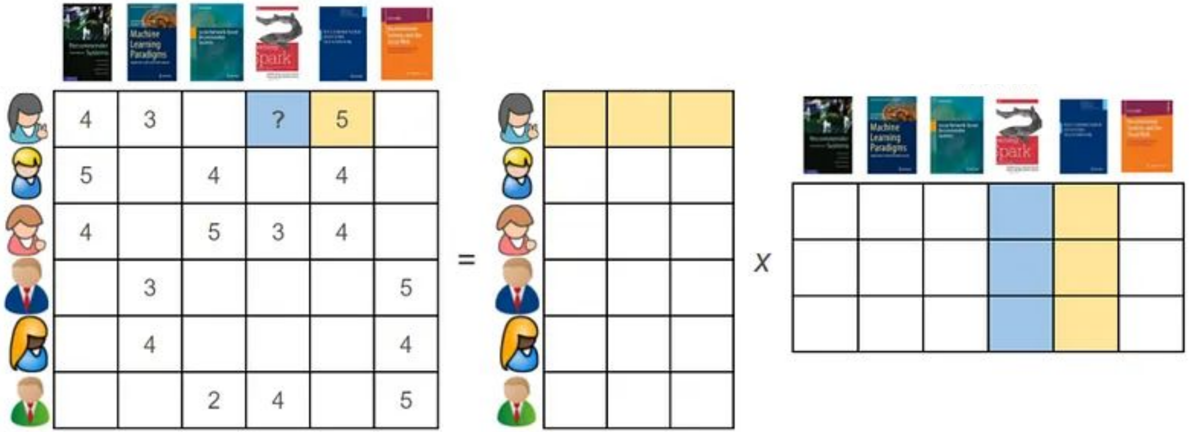
$$R \approx M V_k V_k^T$$

[SVD visualization]

- Could be interpreted as calculating user-item similarity in latent features space considering top-k most important topics
- The best top-k approximation in terms of Frobenius norm
- Folding-in and User/item vector update without retraining

Funk SVD

$$R \approx PQ^T$$



$$\sum_{\boxed{(i,j)\in R}} (r_{ij} - p_i^T q_j)^2 + \sum_{i=1}^{m} \lambda_1 ||p_i||^2 + \sum_{j=1}^{n} \lambda_2 ||q_j||^2 \to \min$$

Image source

Similarity in latent space

- MF provides a solution for top-k personalized recommendations and item2item recommendations
- approximate nearest neighbours search could be applied for fast recommendation building

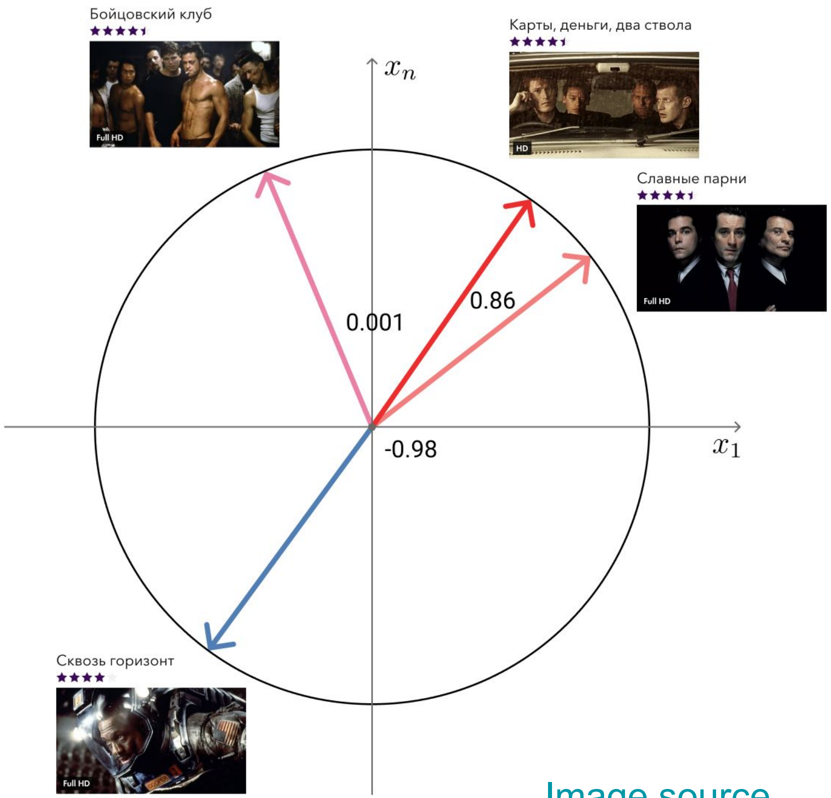


Image source

Recup: cosine similarity, dot product

Dot product

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \ \|\mathbf{b}\| \cos\theta,$$

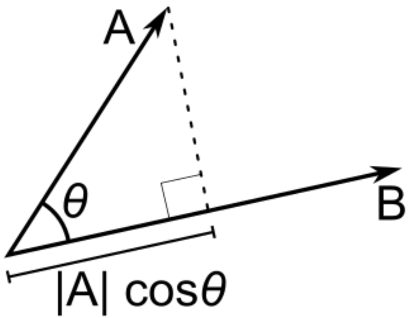
where θ is the angle between $\mathbf{a}$ and $\mathbf{b}$.

a is projected to b:

$$a_b = \|\mathbf{a}\| \cos\theta,$$

.

$$\mathbf{a} \cdot \mathbf{b} = a_b \ \|\mathbf{b}\| = b_a \ \|\mathbf{a}\| \ .$$



Cosine similarity

$$S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

SVD and Funk SVD: pros and cons

Disadvantages

- do not use features and context (could be extended, see Frolov, E., & Oseledets, I. (2019, September). HybridSVD: when collaborative information is not enough)
- Truncated SVD: consider empty values as zeros
- Funk SVD: do not consider empty values
- popularity bias

Advantages

- strong baseline out of the box (scipy, sklearn)
- fast inference with ANN
- Truncated SVD: new user/item vectors inference with closed-form solution (folding in)
- Funk SVD learning objective could be customized for a particular task

Funk SVD customization

- Adding biases

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

$$\min_{b_*, q_*, p_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_i - b_u - q_i^T p_u)^2 + \lambda_4 (b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2) \,.$$

- SVD++: adding implicit feedback

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

y_j - implicit feedback item vectors,

q_i - explicit feedback item vector

- timeSVD, timeSVD++: adding time dependencies

- …

Recommender Systems Handbook, Second Edition, Springer, 2011

Alternating least squares

$$\min_{X,Y} \sum_{\boxed{r_{ui}\ observed}} (r_{ui} - x_u^\intercal y_i)^2 + \lambda\left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2\right)$$

Our approach will be to fix Y and optimize X, then fix X and optimize Y , and repeat until convergence.

Alternating least squares

Algorithm 1 ALS for Matrix Completion

Initialize X, Y
repeat
  for $u = 1 \dots n$ **do**

$$x_u = \Big(\sum_{r_{ui} \in r_{u*}} y_i y_i^{\intercal} + \lambda I_k \Big)^{-1} \sum_{r_{ui} \in r_{u*}} r_{ui} y_i \tag{2}$$

  end for
  for $i = 1 \dots m$ **do**

$$y_i = \Big(\sum_{r_{ui} \in r_{*i}} x_u x_u^{\intercal} + \lambda I_k \Big)^{-1} \sum_{r_{ui} \in r_{*i}} r_{ui} x_u \tag{3}$$

  end for
until convergence

source   Users vectors and item vectors are independent and could be found in parallel

Alternating least squares for implicit feedback (iALS)

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases} \qquad c_{ui} = 1 + \alpha r_{ui}$$

p_{ui} - feedback presence (bool)

c_{ui} - level of certainty

$$\min_{x_\star, y_\star} \sum_{\boxed{u,i}} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

$f \times f$ matrix $Y^T Y$ in time $O(f^2 n)$. For each user u, let us define the diagonal $n \times n$ matrix C^u where $C^u_{ii} = c_{ui}$, and also the vector $p(u) \in \mathbb{R}^n$ that contains all the preferences by u (the p_{ui} values). By differentiation we find an analytic expression for x_u that minimizes the cost function (3):

$$\underbrace{x_u}_{f \times 1} = (\underbrace{Y^T C^u Y}_{f \times f} + \lambda I)^{-1} \underbrace{Y^T C^u}_{f \times n} p(u) \;\blacktriangleright\; x_u = (\underbrace{\boxed{(Y^T Y}}_{f \times f} + \lambda I + \underbrace{\sum_{\boxed{\forall i | p_{ui} \neq 0}} (c_{ui} - 1) y_i y_i^T}_{f \times f})^{-1} (\sum_{\boxed{\forall i | p_{ui} \neq 0}} c_{ui} p_{ui} y_i)$$

n - number of items
f - user/item vector dimension

source

ALS/iALS: pros and cons

Disadvantages

- interactions data only (do not use features and context)
- **quadratic (MSE) loss only**

Advantages

- strong baseline out of the box (implicit, pyspark mllib)
- fast inference with ANN
- new user/item vectors inference with closed-form solution (folding in)
- **any feedback type**
- **good parallelisation**

Additional resources

Articles (RU)

Habr: Рекомендательные системы: идеи, подходы, задачи

Habr: Как работают рекомендательные системы. Лекция в Яндексе

Лекция по рекомендательным системам из курса ВШЭ по машинному обучению

Books

“Recommender Systems. The Textbook”, 2016, Springer

“Recommender Systems Handbook”, 2011, Springer

Frameworks

implicit: iALS and item-based KNN