

Recommender systems

# Content-based and Hybrid systems

Lecture 4  
Fall 2023

Anna Volodkevich

# Contents

Content-based recommender systems

Feature pre-processing

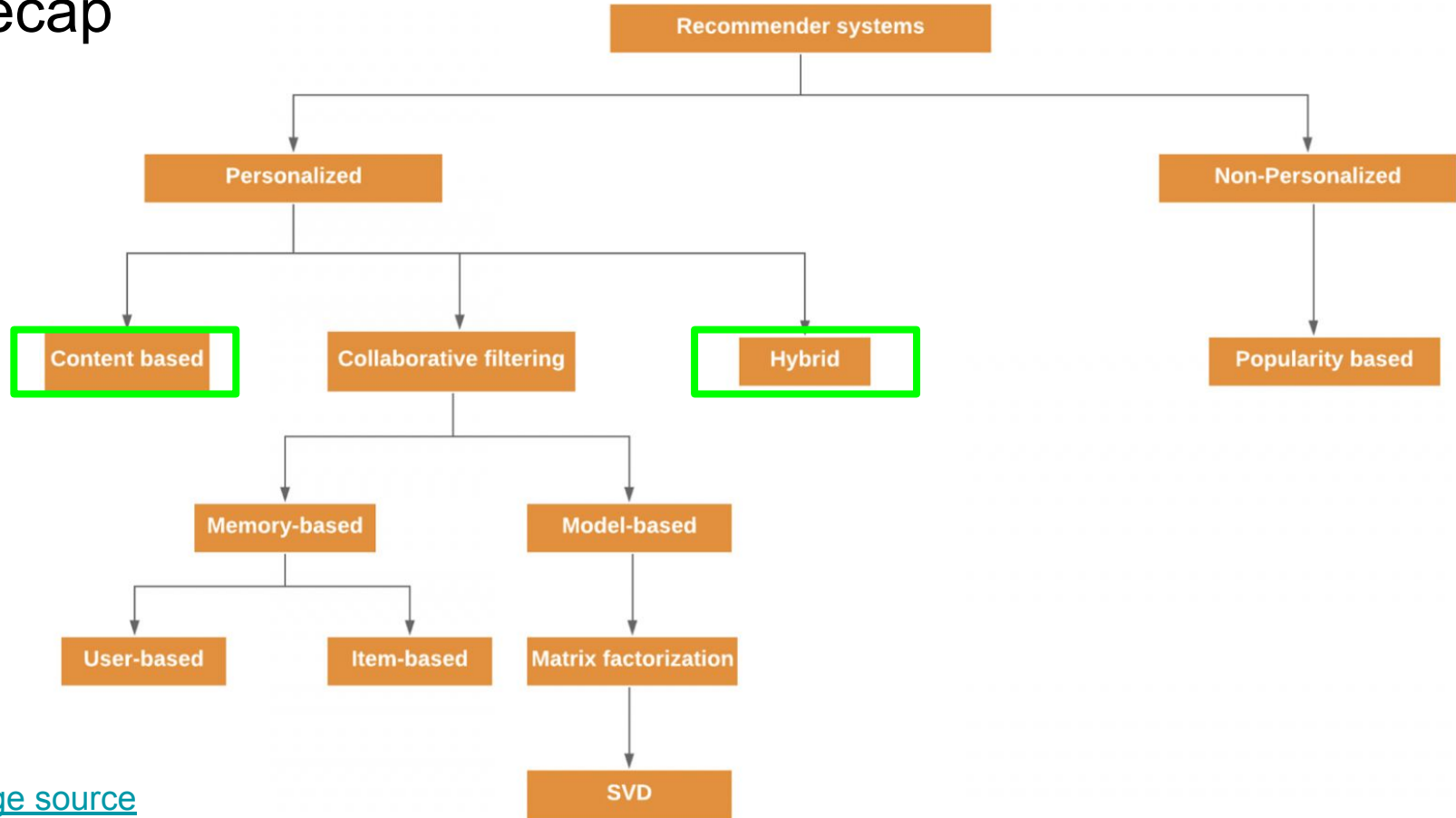
Hybrid recommender systems

Factorization machines

LightFM

Personalized ranking: BPR and WARP loss

# Recap



# Definitions

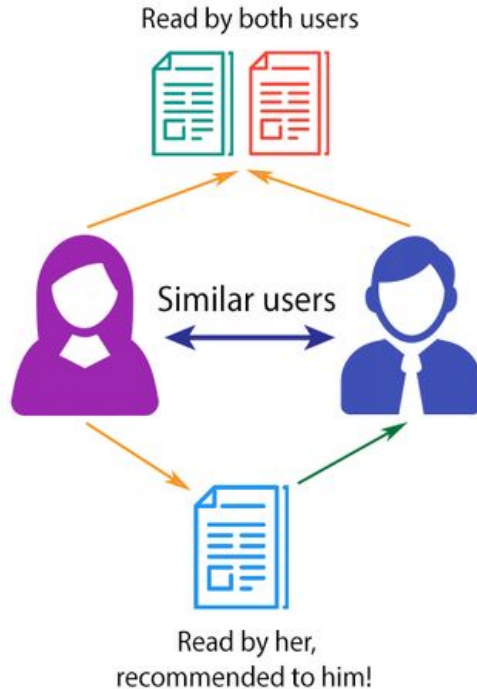
In **content-based** recommender systems, the **descriptive attributes of items** are used to make recommendations. The term “content” refers to these descriptions. In content-based methods, the **ratings and buying behavior of users are combined with the content information available in the items**.

- + RS nature: Demographic, Knowledge-based, Community-based, etc
- + Task specifics: Context-aware, time-aware, location-sensitive etc.

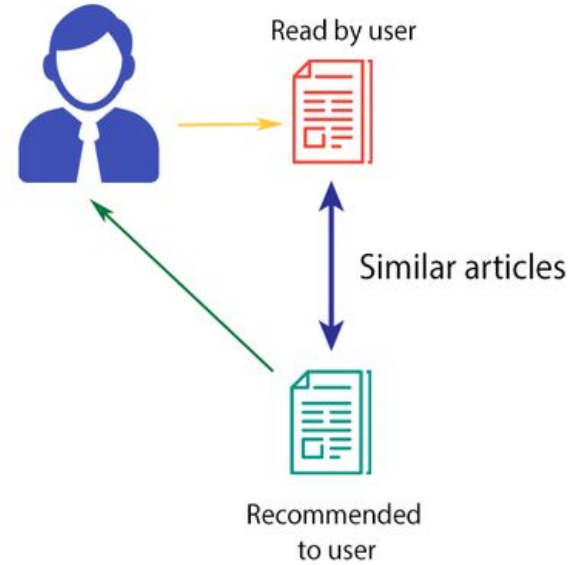
**Hybrid Recommender Systems.** These RSs are **based on the combination of the above mentioned techniques**. A hybrid system combining techniques A and B tries to use the advantages of A to fix the disadvantages of B.

# Content-based vs collaborative filtering

COLLABORATIVE FILTERING



CONTENT-BASED FILTERING

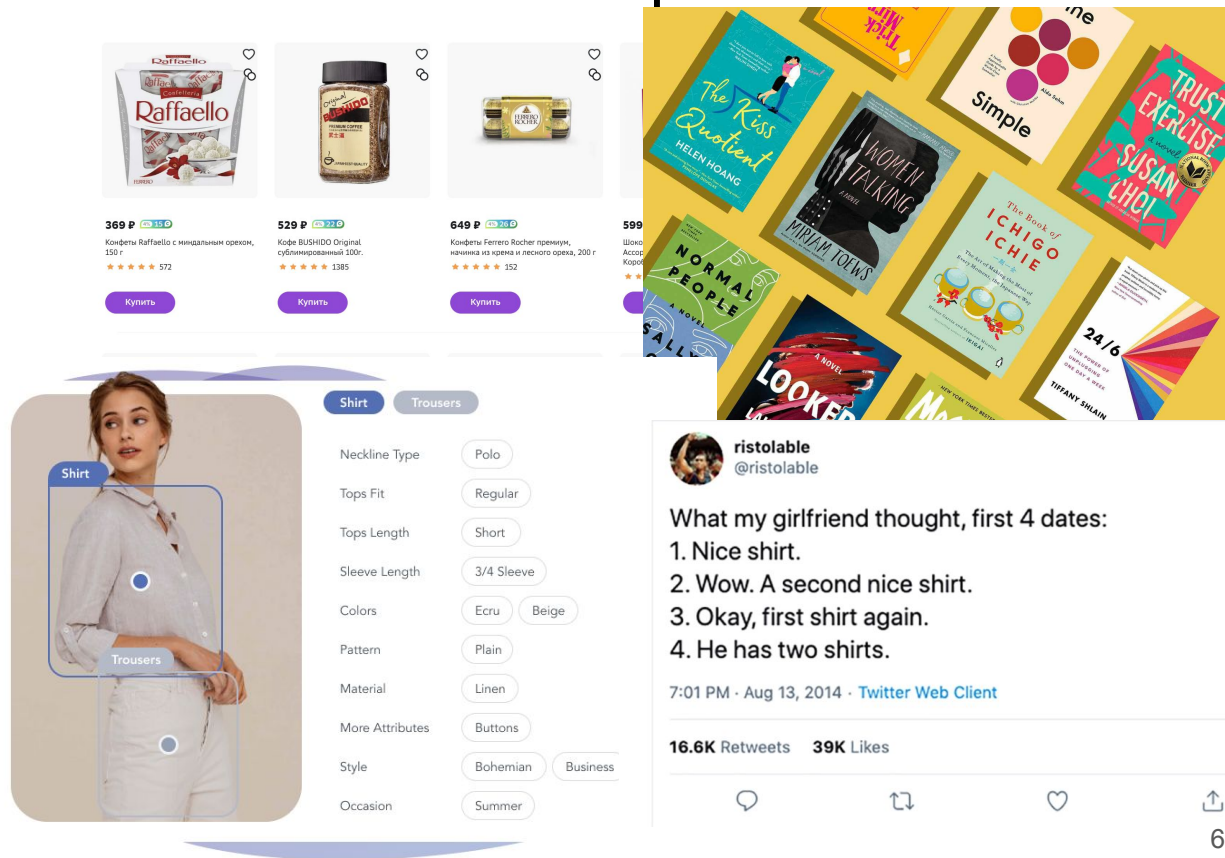


# Recap: available data and environment specifics

- interactions
- user, item features:  
numerical, categorical  
features, tags, images,  
text, etc.
- context: date and time,  
device, weather, etc.

Keep in mind:

- feature types
- aging speed for items
- speed of user preferences drift



# Feature preprocessing

Everything commonly used for a various machine learning tasks:

- numerical: binarization, scaling
- categorical/tags: one-hot encoding/ multilabel encoding
- text: BOW, TF-IDF, pre-trained word embeddings, etc
- images: autoencoders, etc
- time: one-hot encoding, cyclical encoding, etc

+ generated features (next lecture)

# Content-based models





$$r_{ui} = \frac{\sum_{j \in I_u} \text{sim}(i, j) r_{uj}}{\sum_{j \in I_u} |\text{sim}(i, j)|}$$

$$r_{ui} = \max_{j \in I_u, r_{uj} > \alpha} \text{sim}(i, j) r_{uj}$$

$\text{sim}(i, j)$  - item-item similarity, depends in features: cosine, dot, jaccard, etc.

Recap: Jaccard similarity

$$J(I, J) = \frac{|I \cap J|}{|I \cup J|}$$

|   | Education | Casual | Health |     | TimeWastr | Science R Us | Healthcare |
|---|-----------|--------|--------|-----|-----------|--------------|------------|
|  | ●         |        |        |     |           | ●            |            |
|  |           | ●      |        | ... | ●         |              |            |
|  |           |        | ●      |     |           |              | ●          |
|  | ●         |        |        | ... |           | ●            | ●          |

[Image source](#)

- solve cold-start problem for cold items
- applicable for the users with short history
- provide obvious recommendations
- do not use collaborative information



# Solving user cold-start problem. Demographic models

This type of system recommends items based on the demographic profile of the user. The assumption is that **different recommendations should be generated for different demographic niches**. Many websites adopt simple and effective personalization solutions based on demographics. For example, users are dispatched to particular websites based on their language or country. Or, suggestions may be customized according to the age of the user.

## Basic ideas:

- rule-based models
- user-based nearest neighbours
- users clustering and building separate models for each cluster
- users clustering and building a model for clusters instead of users
- interaction matrix factorization given user vectors in users' features space
- session-based models
- etc.

# Factorization machines (FM)

- Use user-item pairs, features, context
- Consider feature combinations/pairs, e.g. “film genre=Action, user age=18-35”
- Learn a vector for each feature to model weight for each feature pair as a dot product
- Customizable loss function based on the task

[source:FM](#)   [source:FFM](#)

## A. Factorization Machine Model

1) *Model Equation*: The model equation for a factorization machine of degree  $d = 2$  is defined as:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \quad (1)$$

where the model parameters that have to be estimated are:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^n, \quad \mathbf{V} \in \mathbb{R}^{n \times k} \quad (2)$$

| Feature vector $\mathbf{x}$ |      |   |   |     |       |    |    |    |     |                    |     |     |     |     |      |                  | Target $y$ |    |    |     |   |           |
|-----------------------------|------|---|---|-----|-------|----|----|----|-----|--------------------|-----|-----|-----|-----|------|------------------|------------|----|----|-----|---|-----------|
| $\mathbf{x}^{(1)}$          | 1    | 0 | 0 | ... | 1     | 0  | 0  | 0  | ... | 0.3                | 0.3 | 0.3 | 0   | ... | 13   | 0                | 0          | 0  | 0  | ... | 5 | $y^{(1)}$ |
| $\mathbf{x}^{(2)}$          | 1    | 0 | 0 | ... | 0     | 1  | 0  | 0  | ... | 0.3                | 0.3 | 0.3 | 0   | ... | 14   | 1                | 0          | 0  | 0  | ... | 3 | $y^{(2)}$ |
| $\mathbf{x}^{(3)}$          | 1    | 0 | 0 | ... | 0     | 0  | 1  | 0  | ... | 0.3                | 0.3 | 0.3 | 0   | ... | 16   | 0                | 1          | 0  | 0  | ... | 1 | $y^{(2)}$ |
| $\mathbf{x}^{(4)}$          | 0    | 1 | 0 | ... | 0     | 0  | 1  | 0  | ... | 0                  | 0   | 0.5 | 0.5 | ... | 5    | 0                | 0          | 0  | 0  | ... | 4 | $y^{(3)}$ |
| $\mathbf{x}^{(5)}$          | 0    | 1 | 0 | ... | 0     | 0  | 0  | 1  | ... | 0                  | 0   | 0.5 | 0.5 | ... | 8    | 0                | 0          | 1  | 0  | ... | 5 | $y^{(4)}$ |
| $\mathbf{x}^{(6)}$          | 0    | 0 | 1 | ... | 1     | 0  | 0  | 0  | ... | 0.5                | 0   | 0.5 | 0   | ... | 9    | 0                | 0          | 0  | 0  | ... | 1 | $y^{(5)}$ |
| $\mathbf{x}^{(7)}$          | 0    | 0 | 1 | ... | 0     | 0  | 1  | 0  | ... | 0.5                | 0   | 0.5 | 0   | ... | 12   | 1                | 0          | 0  | 0  | ... | 5 | $y^{(6)}$ |
|                             | A    | B | C | ... | TI    | NH | SW | ST | ... | TI                 | NH  | SW  | ST  | ... | Time | TI               | NH         | SW | ST | ... |   |           |
|                             | User |   |   |     | Movie |    |    |    |     | Other Movies rated |     |     |     |     |      | Last Movie rated |            |    |    |     |   |           |

# Factorization machines (FM)

- Advantages

- uses features and their combinations
- suits for cold items and users
- different loss functions available
- differs from [PolynomialFeatures](#), as learns vector for each feature (not weight for each combination)

- Disadvantages

- linear dependency on predicted score is assumed for numerical features
- there is no vector representation of user and item and no fast inference
- no sequential component
- may worse generalize collaborative data for some tasks

- Open source implementations

- [LibFM](#)
- [RankFM](#)

# LightFM

The latent representation of user  $u$  is given by the sum of its features' latent vectors:

$$\mathbf{q}_u = \sum_{j \in f_u} \mathbf{e}_j^U$$

The same holds for item  $i$ :

$$\mathbf{p}_i = \sum_{j \in f_i} \mathbf{e}_j^I$$

The bias term for user  $u$  is given by the sum of the features' biases:

$$b_u = \sum_{j \in f_u} b_j^U$$

The same holds for item  $i$ :

$$b_i = \sum_{j \in f_i} b_j^I$$

The model's prediction for user  $u$  and item  $i$  is then given by the dot product of user and item representations, adjusted by user and item feature biases:

$$\hat{r}_{ui} = f(\mathbf{q}_u \cdot \mathbf{p}_i + b_u + b_i) \quad (1)$$

[paper](#)

There is a number of functions suitable for  $f(\cdot)$ . An identity function would work well for predicting ratings; in this paper, I am interested in predicting binary data, and so after Rendle *et al.* [16] I choose the sigmoid function

$$f(x) = \frac{1}{1 + \exp(-x)}.$$

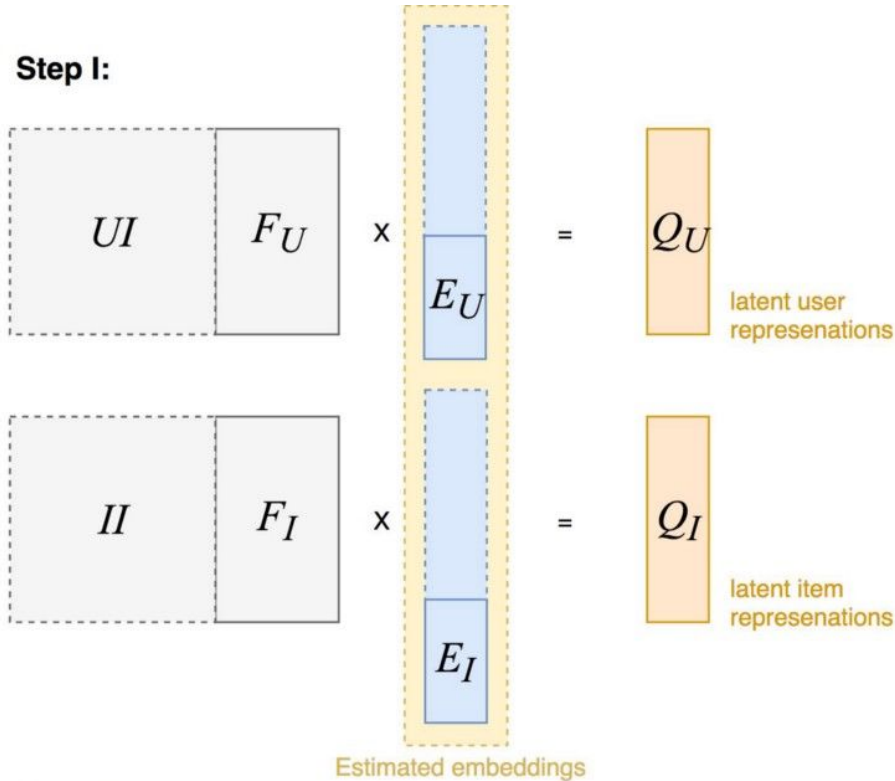
The optimisation objective for the model consists in maximising the likelihood of the data conditional on the parameters. The likelihood is given by

$$L(\mathbf{e}^U, \mathbf{e}^I, \mathbf{b}^U, \mathbf{b}^I) = \prod_{(u,i) \in S^+} \hat{r}_{ui} \times \prod_{(u,i) \in S^-} (1 - \hat{r}_{ui}) \quad (2)$$

original paper refers to the dataset with negative and positive feedback. negative feedback is often not available or insufficient.

# LightFM

Step I:



$$r_{ui} = \langle q_u, p_i \rangle + b_u + b_i$$

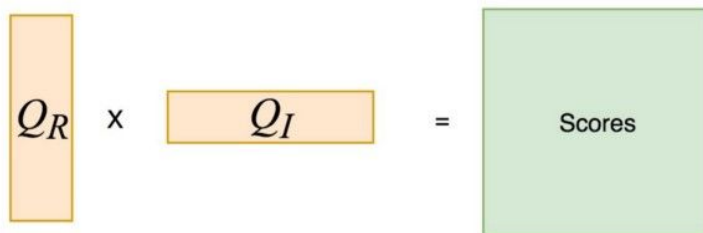
$$q_u = \sum_{j \in f_u} e_j^U$$

$$e_j^U = w_j^U e_j$$

$$b_u = \sum_{j \in f_u} b_j^U$$

$$b_j^U = w_j^U b_j$$

Step II:



# LightFM

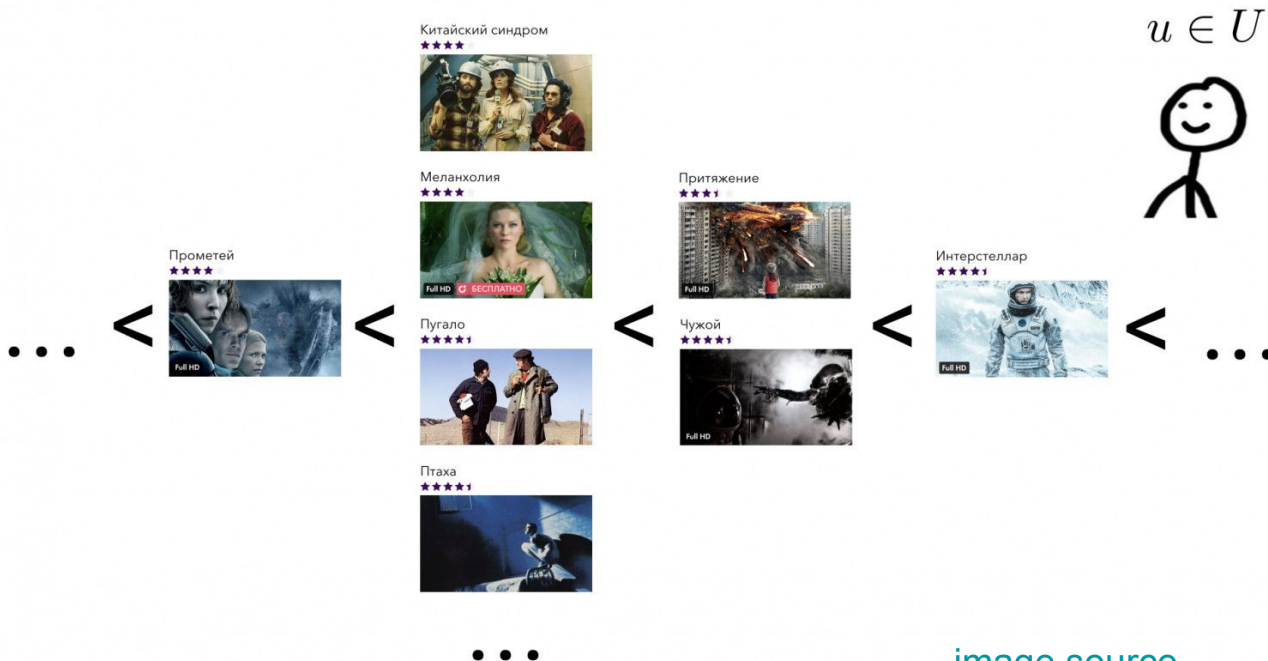
- gives user and item vectors for fast inference unlike general FM
- popular implementation with several loss function is open sourced
- may work with and without features

# Ranking task

Не понравившиеся фильмы  
(с отрицательным ранком)

Ещё не смотрел

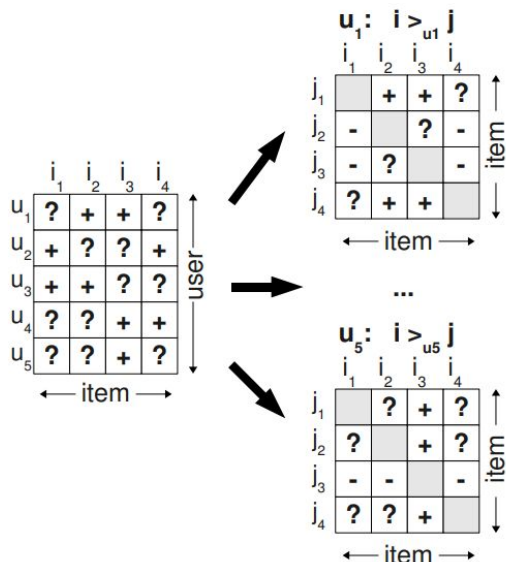
Понравившиеся фильмы  
(с положительным ранком)



[image source](#)

# Ranking task: Bayesian Personalized Ranking (BPR)

- We actually want to get the items, that the user would prefer to others, which leads to a ranking problem.
- Will consider triplets of user ( $u$ ), positive item ( $i$ ), negative/unknown item ( $j$ ). We want to score the positive item higher than the negative one.



The Bayesian formulation of finding the correct personalized ranking for all items  $i \in I$  is to maximize the following posterior probability where  $\Theta$  represents the parameter vector of an arbitrary model class (e.g. matrix factorization).

$$p(\Theta | >_u) \propto p(>_u | \Theta) p(\Theta)$$

Here,  $>_u$  is the desired but latent preference structure for user  $u$ . All users are presumed to act independently

$$\prod_{u \in U} p(>_u | \Theta) = \prod_{(u, i, j) \in D_S} p(i >_u j | \Theta)$$

$$D_S := \{(u, i, j) | i \in I_u^+ \wedge j \in I \setminus I_u^+\}$$

The semantics of  $(u, i, j) \in D_S$  is that user  $u$  is assumed to prefer  $i$  over  $j$ . As  $>_u$  is antisymmetric, the negative cases are regarded implicitly.

$$p(i >_u j | \Theta) := \sigma(\hat{x}_{uij}(\Theta))$$

where  $\sigma$  is the logistic sigmoid:

$$\sigma(x) := \frac{1}{1 + e^{-x}}$$

$$\hat{x}_{uij} := \hat{x}_{ui} - \hat{x}_{uj}$$



# Ranking task: Bayesian Personalized Ranking (BPR)

$$D_S := \{(u, i, j) | i \in I_u^+ \wedge j \in I \setminus I_u^+\}$$

The semantics of  $(u, i, j) \in D_S$  is that user  $u$  is assumed to prefer  $i$  over  $j$ . As  $>_u$  is antisymmetric, the negative cases are regarded implicitly.

$$p(\Theta) \sim N(0, \Sigma_\Theta)$$

$$\begin{aligned} \text{BPR-OPT} &:= \ln p(\Theta | >_u) \\ &= \ln p(>_u | \Theta) p(\Theta) \\ &= \ln \prod_{(u, i, j) \in D_S} \sigma(\hat{x}_{uij}) p(\Theta) \\ &= \sum_{(u, i, j) \in D_S} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta) \\ &= \sum_{(u, i, j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \|\Theta\|^2 \end{aligned}$$

where  $\lambda_\Theta$  are model specific regularization parameters.

$$\begin{aligned} \frac{\partial \text{BPR-OPT}}{\partial \Theta} &= \sum_{(u, i, j) \in D_S} \frac{\partial}{\partial \Theta} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \frac{\partial}{\partial \Theta} \|\Theta\|^2 \\ &\propto \sum_{(u, i, j) \in D_S} \frac{-e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} - \lambda_\Theta \Theta \end{aligned}$$

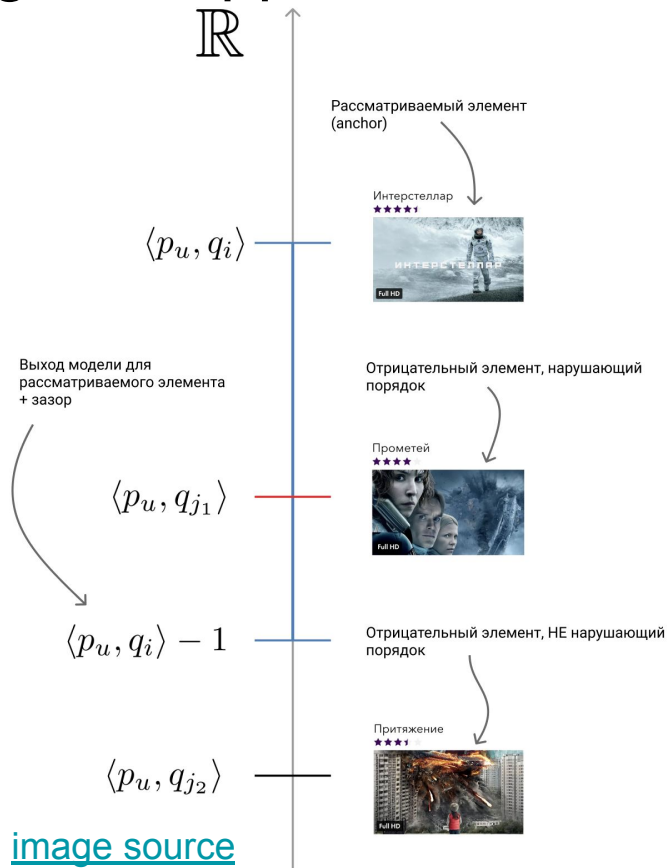
```

1: procedure LEARNBPR( $D_S, \Theta$ )
2:   initialize  $\Theta$ 
3:   repeat
4:     draw  $(u, i, j)$  from  $D_S$ 
5:      $\Theta \leftarrow \Theta + \alpha \left( \frac{e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \cdot \Theta \right)$ 
6:   until convergence
7:   return  $\hat{\Theta}$ 
8: end procedure
    
```

- related to AUC optimization
- may be applied to different backbone models (MF, KNN, FM)

# Ranking task: Weighted Approximate-Rank Pairwise (WARP)

- Will consider triplets of user ( $u$ ), positive item ( $i$ ), negative/unknown item ( $j$ ). We want to score the positive item higher than the negative one plus small border.
- Will sample negative items and find the rank of wrong ordering
- The more sampled items are ordered correctly, the higher the model quality and smaller update should be performed.



Repeat:

- 1) Sample positive pair  $u, i$
- 2)  $N = 0$  repeat:
- 3) Sample negative item  $j$
- 4)  $N += 1$
- 5) If  $f(u, i) < f(u, j) + 1$ : model update
- 6) elif  $N > \text{max\_sampled}$ : break

\*dot product on the picture may be replaced with another function  $f(u, i)$ , returning score for user-item pair

# Ranking task: Weighted Approximate-Rank Pairwise (WARP)

$$\text{rank}(u, i) \approx \lfloor \frac{|I| - 1}{N} \rfloor$$

$N$  - number of negatives sampled  
 $|I|$  - number of items

$$L(k) = \log(k)$$

$$L(k) = \sum_{i=1}^k \frac{1}{i}$$

$$\text{err}(u, i, j) = L(\text{rank}(u, i)) |1 - f(u, i) + f(u, j)|_+$$

hinge loss =  $\max(0, x)$

[paper](#)

[mts video](#)

[okko article](#)

- related to precision@k optimization
- may be applied to different backbone models (MF, KNN, FM)

# Additional sources

[People meet recommender systems. Factorization](#) (FM, BPR)

[OKKO post on Habr](#)

[Your first recsys MTS course](#)

“Recommender Systems. The Textbook”, 2016, Springer

“Recommender Systems Handbook”, 2011, Springer