ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Lesson 8: Recommender System

Agenda

1. Summary on Recommender System
2. Evaluation methods
3. Colaborative filtering using k-Nearest Neightbors
4. Colaborative filtering using Maxtrix Factorization
5. Neural Colaborative Filtering
6. Session-based Recommender System

1. Summary

Why we need recommender system?

- Users are overloaded with information on the web
- Sellers need to offer the right product to
  - Increase sale revenue
  - Improve serving quality
- The trend of personalization and digitization is inevitable

Recommender System and Query System

- Query System: Users express their wishes through a query
- Recommender System : Users do not know what they want

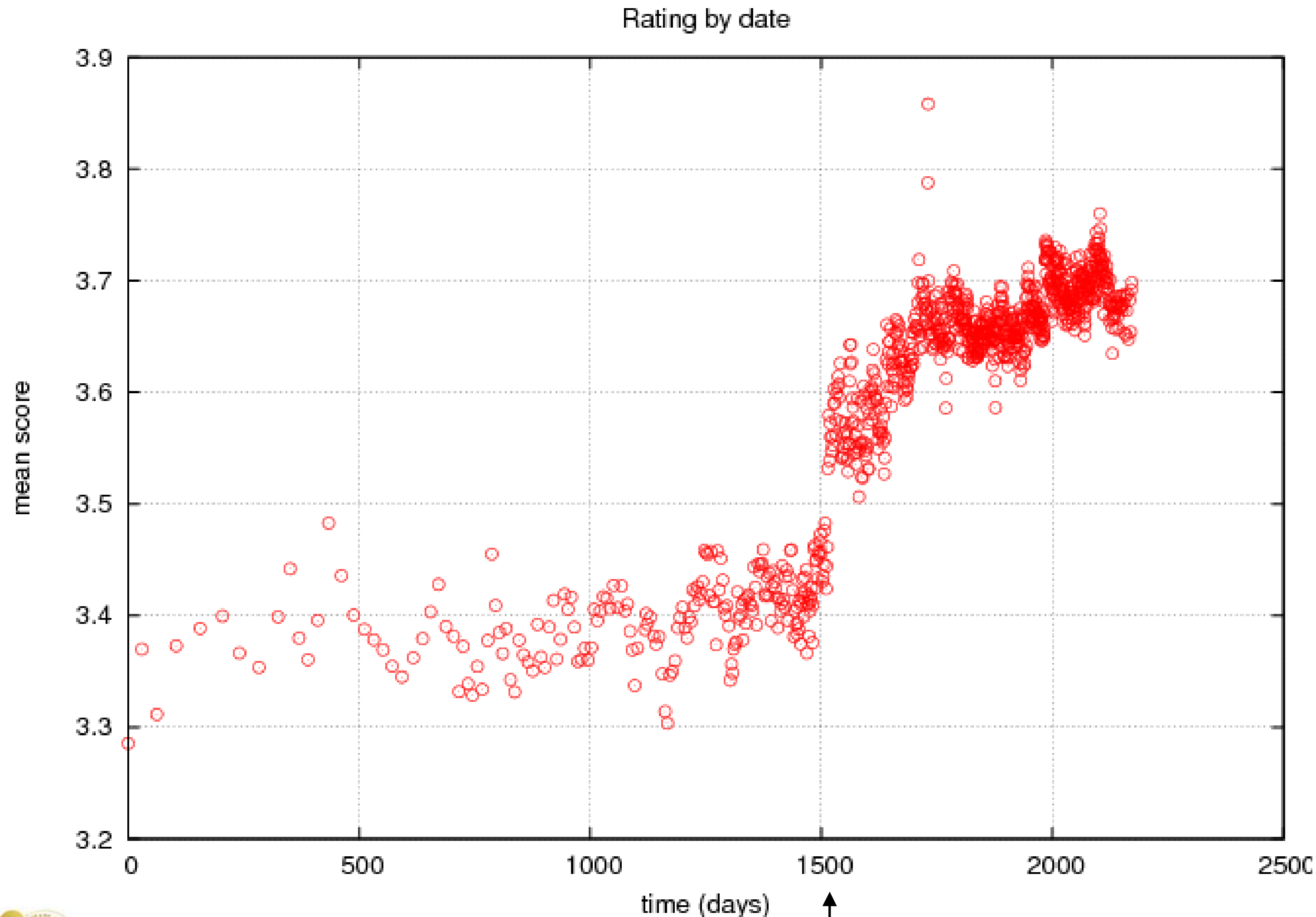
Application fields

- Ecommerce
- Online entertainment
- Online News
- Forums, social networks
- Scientific research
- Online Dating

Application fields (cont.)

- *Amazon*:
  - Recommend products
  - More than 30% increase in revenue
- *Netflix*:
  - Recommend movies, TV shows
  - Bring in $1B per year
- *Google News*:
  - News Suggestions
  - Nearly 40% increase in traffic

Application fields (cont.)



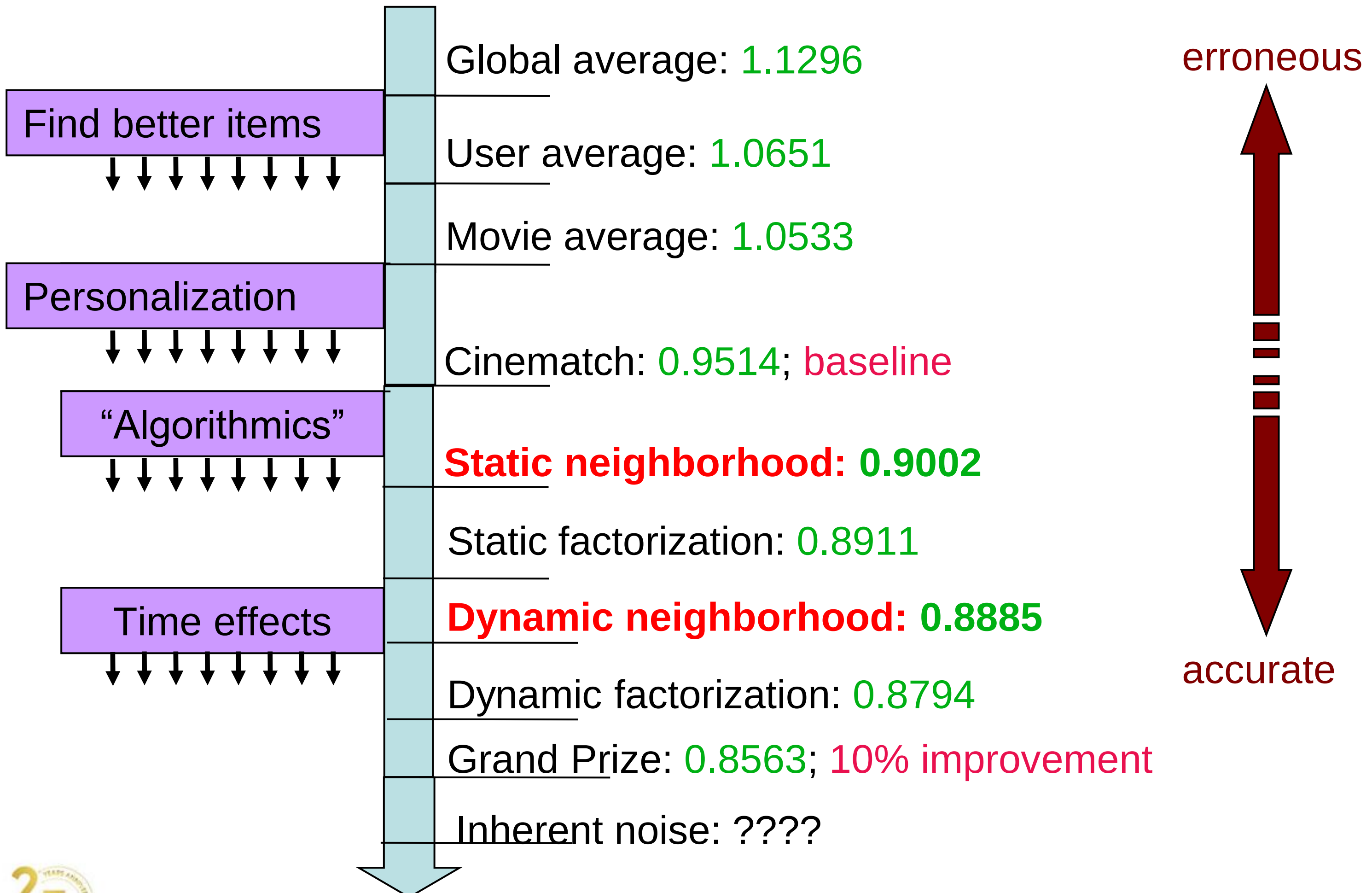
2004

From Y. Koren (2009)

Recommendation methods

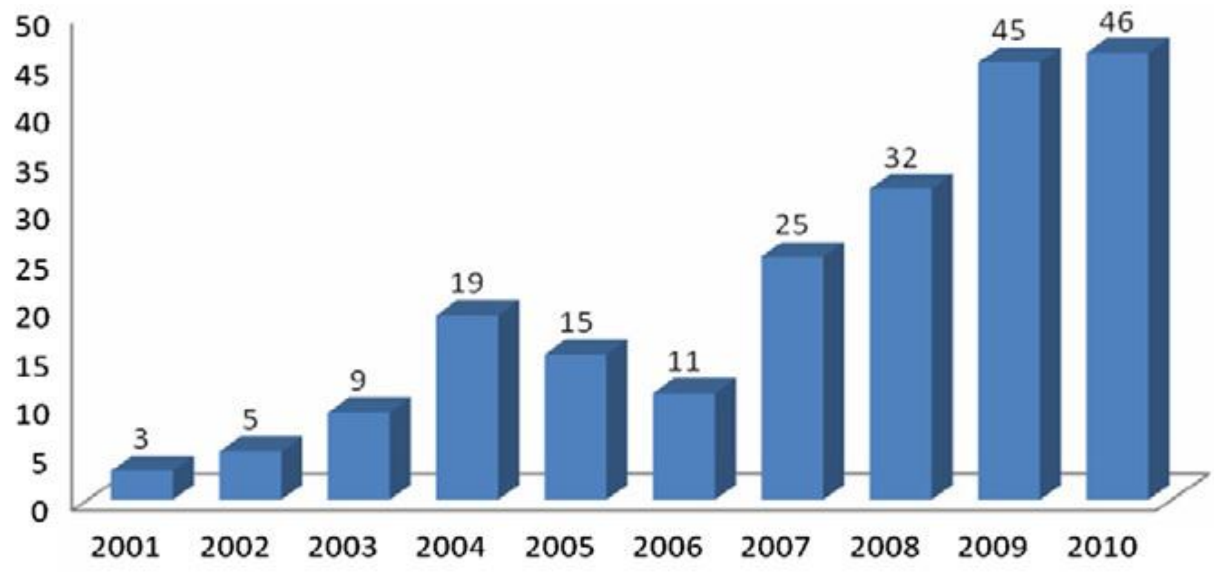
- Content-based:
  - Suggestions based on user's transaction history
- Colaborative filtering:
  - Suggestions based on users with similar interests
- Session-based:
  - Suggestions based on transaction chains
- Hybrid methods

Netflix challenge



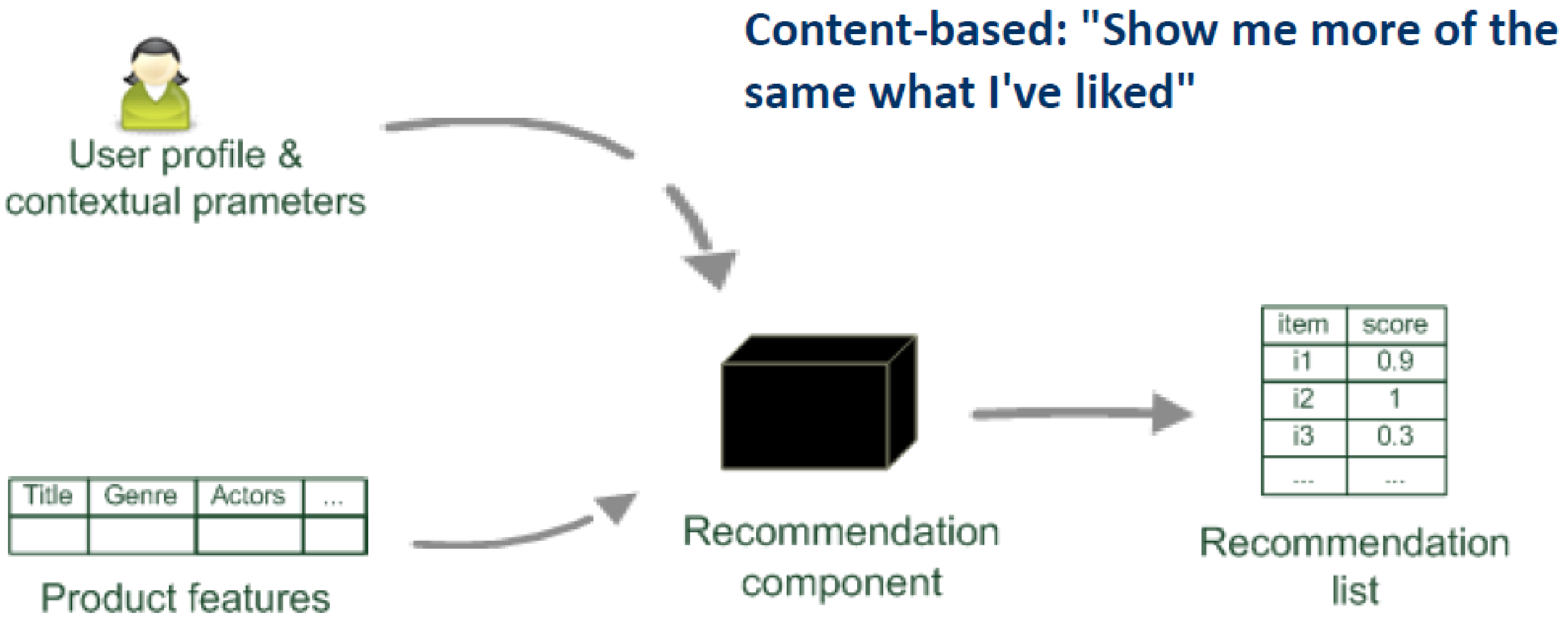
From Y. Koren

Netflix challenge (cont.)



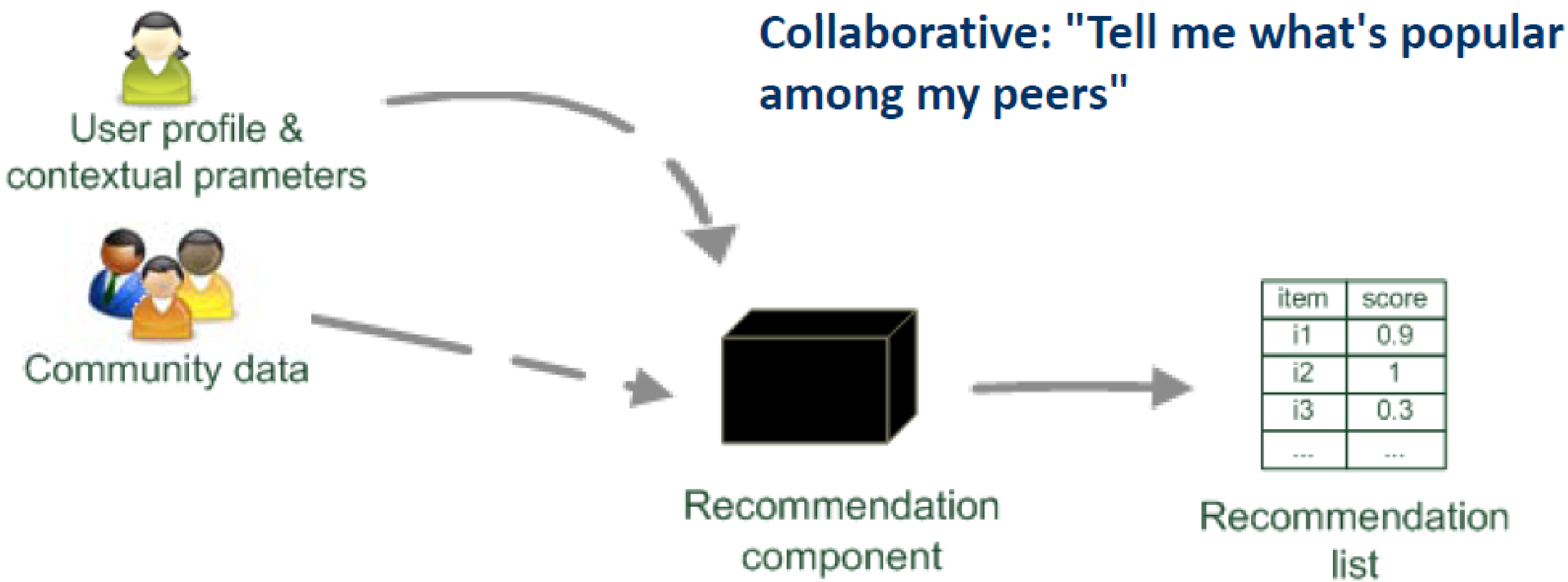
Number of papers on recsys by years

Content-based



from Recommender Systems: An Introduction

Colaborative Filtering



from Recommender Systems: An Introduction

Challenge of recommender system

- The number of transactions is very small compared to the actual number of users and products
- Not enough information about new users and products
- Users and products change over time, seasonally
- Consumption habits change over time, seasonally
- Real-time recommendation

2. Evaluation methods

- Give
  - User set U
  - Item set I
- Dataset of transactions (u, i, r_{ui}, t)
  - u: user $u \in U$
  - i: item I $\in I$
  - r_{ui}: ratings of user u to item i
  - t: rating time

- r_{ui}
  - 5-stage scale (1, 2, 3, ,4 ,5)
  - Binary scale (0, 1)
- Dataset separated into train set and test set
- RS is trained on train set
- On test set, RS predict ratings p_{ui} of user *u* to item *i*

train/test

	i_1	i_2	i_3	i_4
u_1	-	5	3	-
u_2	4	-	2	3
u_3	4	1	-	5

	i_1	i_2	i_3	i_4
u_1	-	5		-
u_2		-	2	3
u_3	4	1	-	

train

	i_1	i_2	i_3	i_4
u_1	-		3	-
u_2	4	-		
u_3			-	5

test

Metrics

- (N)MAE
- RMSE
- Ranking:
  - Precision/Recall/F-score

MAE

$$MAE = \frac{\sum_{ui} |p_{ui} - r_{ui}|}{n}$$

- p_{ui}: prediction of model on ratings of user *u* to item *i*
- r_{ui}: ratings of user *u* to item *i*
- *n*: number of examples in test set

NMAE

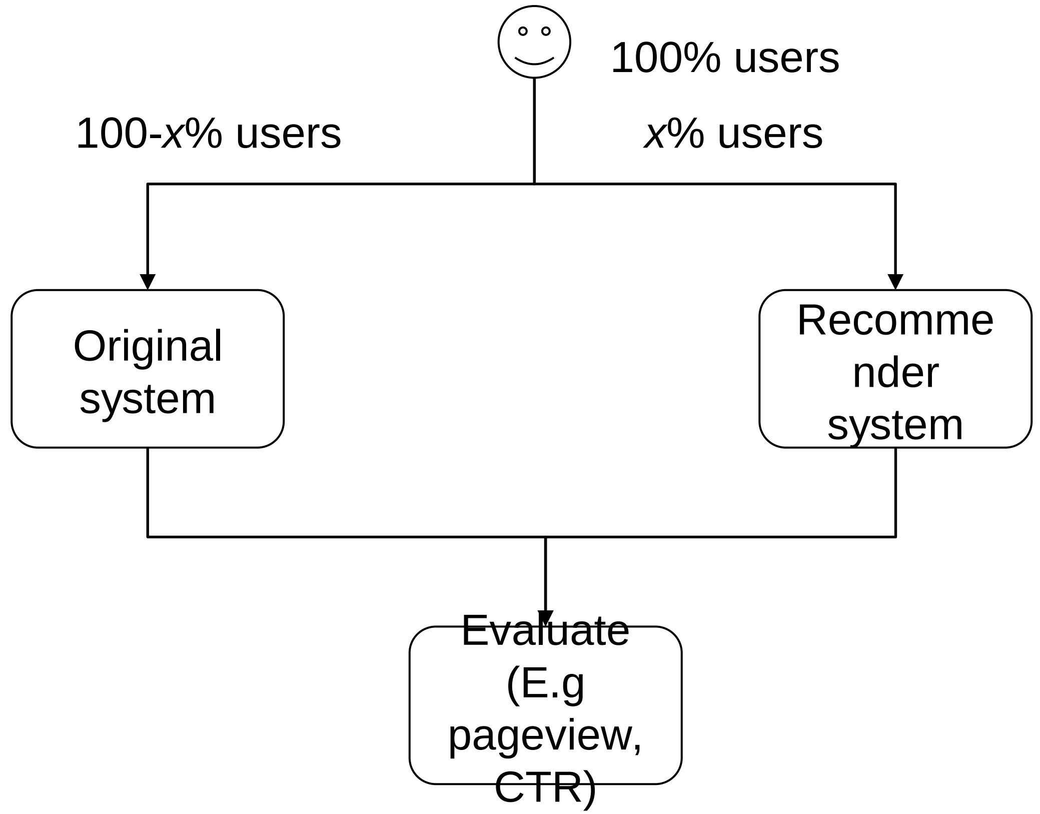
$$NMAE = \frac{MAE}{r_{max} - r_{min}}$$

- r_{max}: User's maximum ratings value
- r_{min}: User’s minimum ratings value

RMSE

$$RMSE = \sqrt{\frac{1}{n}\sum_{ui}(p_{ui} - r_{ui})^2}$$

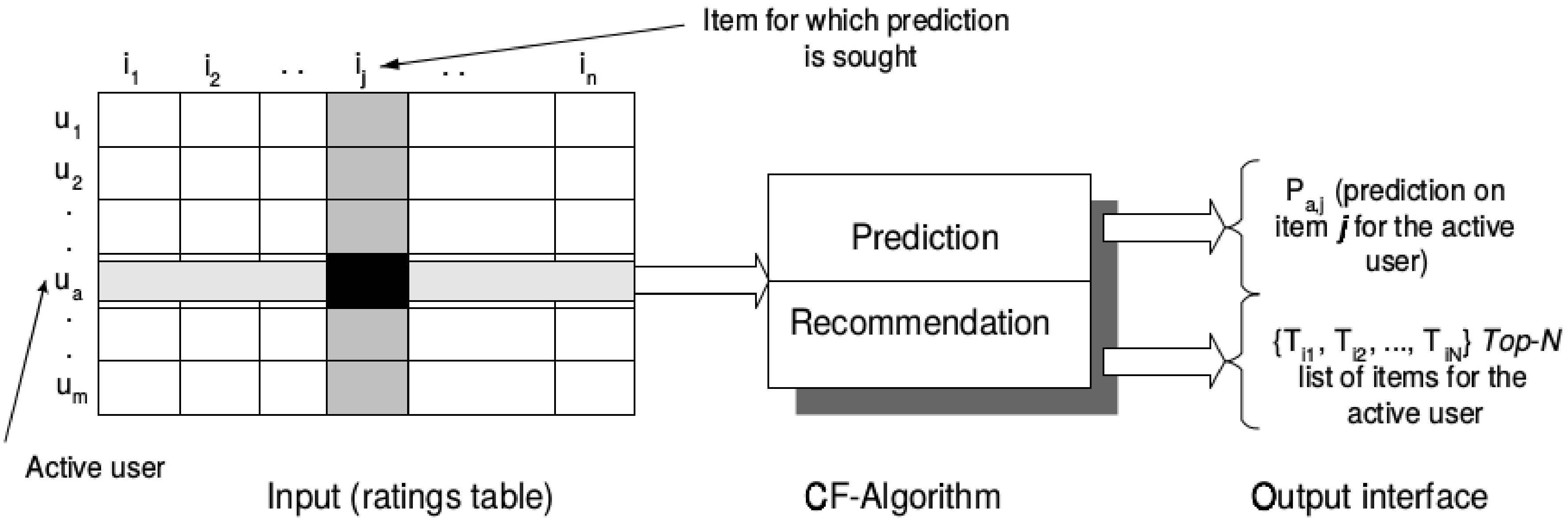
A/B testing


100% users
100-*x*% users
x% users
Original system
Recommender system
Evaluate (E.g pageview, CTR)

3. Colaborative Filtering using kNN

- Based on product user interaction matrix
- No training
- Recommend based on users
  - Find set *V* of similar users to user *u*
  - Compute ratings for item *i* based on ratings of user in *V* to item *i*

Recommend based on users



from B. Sarwar et al 2010

User similarity

$$sim(\mathbf{u},\mathbf{v}) = \frac{\sum_{i\in C}(r_{ui} - \bar{r}_{\mathbf{u}})(r_{vi} - \bar{r}_{\mathbf{v}})}{\sqrt{\sum_{i\in C}(r_{ui} - \bar{r}_{\mathbf{u}})^2}\sqrt{\sum_{i\in C}(r_{vi} - \bar{r}_{\mathbf{v}})^2}}$$

- *C*: set of item on that both user *u* and user *v* rated
- r_u: Average ratings of user *u* (only counting for items that u rated)

Predict ratings

$$p_{\mathbf{u}i} = \bar{r}_{\mathbf{u}} + \frac{\sum_{\mathbf{v} \in V} sim(\mathbf{u}, \mathbf{v})(r_{vi} - \bar{r}_{\mathbf{v}})}{\sum_{\mathbf{v} \in V} |sim(\mathbf{u}, \mathbf{v})|}$$

- *V*: top *k* similar users to user *u*

Example (k = 2)

	i_1	i_2	i_3	i_4
u_1	5	4	4	1
u_2	2	1		
u_3	5	4	4	?
u_4		1	2	5

r_1 = 14/4
r_2 = 3/2
r_3 = 13/3
r_4 = 8/3

sim(u_3,u_1) = ~0.492
sim(u_3,u_2) = ~0.948
sim(u_3,u_4) = ~0.919

p(u_3,i_4) = ~ 6.67

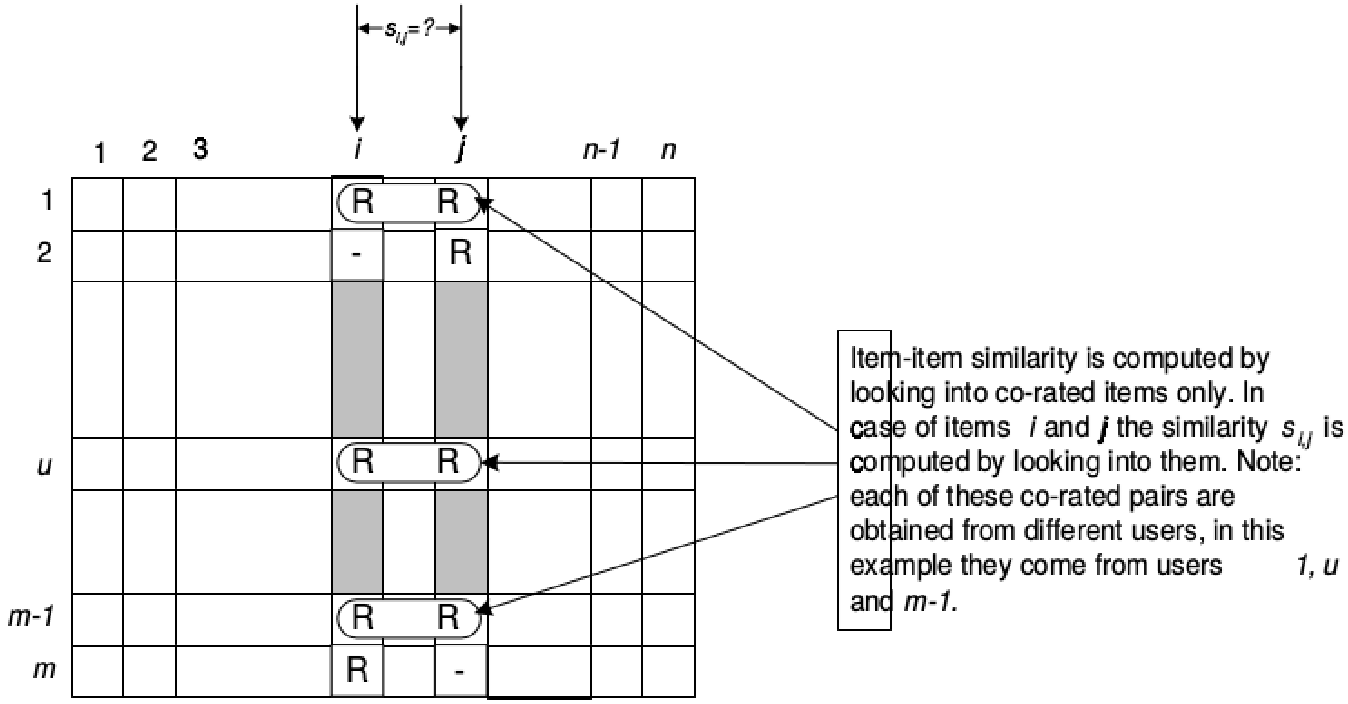
Disadvantages

- Regularly update the user vector when the user has a new transaction
- Requires computation on the entire set of users

Recommend products

- Product representation based on user-product interaction matrix
- Suitable for systems with product number << number of users
- Lower update frequency of product vectors
- Can calculate product-product similarity in advance

Product representation



Product similarity

$$sim(i,j) = \frac{\sum_{\mathbf{u}\in U}(r_{ui}-\bar{r}_{\mathbf{u}})(r_{uj}-\bar{r}_{\mathbf{u}})}{\sqrt{\sum_{\mathbf{u}\in U}(r_{ui}-\bar{r}_{\mathbf{u}})^2}\sqrt{\sum_{\mathbf{u}\in U}(r_{uj}-\bar{r}_{\mathbf{u}})^2}}$$

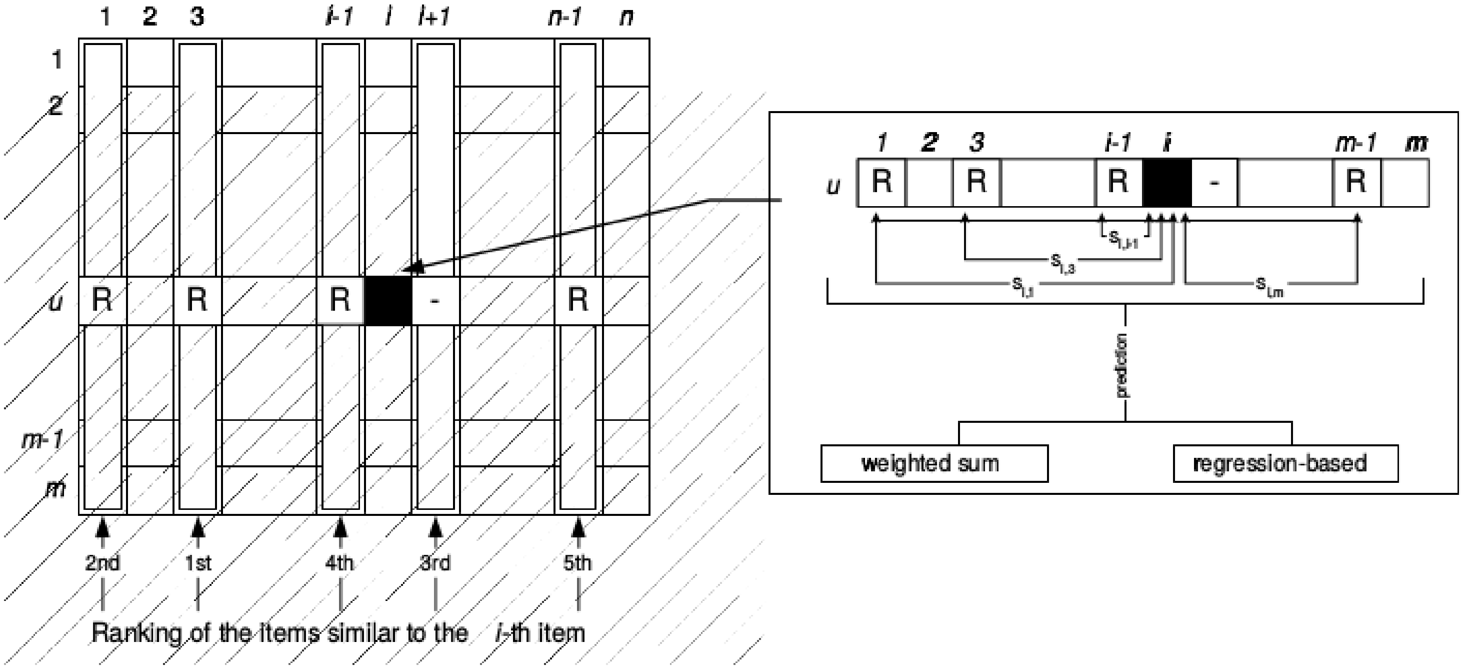
- *U*: set of users rated both item *i* and item *j*

Ratings prediction

$$p(\mathbf{u}, i) = \frac{\sum_{j \in J} r_{\mathbf{u},j} \times sim(i, j)}{\sum_{j \in J} sim(i, j)},$$

J: top *k* similar items to item *i*

Example



from B. Sarwar et al 2010

4. Colaborative Filtering using MF

- Based on the overall context of the user-product interaction
- Representing users, products in latent aspects
- Using Matrix Factorization technique
- Based on user and product vectors to make predictions

SVD

- User – movies matrix ***R***
- Factorize ***R*** into user-aspect matrix ***U*** and movies-aspect matrix ***M***
- Each user is represented by a K-dimensional vector, where K is the number of latent aspect
- Each movie is represented by a K-dimensional vector

$$\boldsymbol{R} \approx \boldsymbol{U}^{\mathrm{T}} \boldsymbol{M} . \qquad r_{ij} \approx \mathbf{u}_i^{\mathrm{T}} \mathbf{m}_j = \sum_{k=1}^{K} u_{ki} \times m_{kj} ,$$

Ratings predictions

$$p_{ij} = \sum_{k=1}^{K} u_{ki} \times m_{kj} \; .$$

- p_{ij}: predict ratings of user *i* to movie *j*
- $\boldsymbol{u}_i$: User vecrtor *i*
- m_j: movie vector *j*

Hàm lỗi:

$$e_{ij} = r_{ij} - p_{ij} \; .$$

e_{ij}^2

Model

- U, M is learning parameters
- R is training dataset
- Mean Square Error: $E_{ij} = e_{ij}^2 = (p_{ij} - r_{ij})^2$
- Learning technique: Gradient Descent

Gradient Descent

Derivative of the error function wrt u_{ki}

$$\frac{\partial (e_{ij})^2}{\partial u_{ki}} = 2e_{ij}\frac{\partial e_{ij}}{\partial u_{ki}} .$$

$$\frac{\partial e_{ij}}{\partial u_{ki}} = -\frac{\partial p_{ij}}{\partial u_{ki}} .$$

$$\frac{\partial (e_{ij})^2}{\partial u_{ki}} = 2e_{ij}(-m_{kj}) = -2(r_{ij} - p_{ij})m_{kj} .$$

Gradient Descent (Cont.)

Derivative of the error function wrt m_{kj}

$$\frac{\partial (e_{ij})^2}{\partial m_{kj}} = 2e_{ij}(-u_{ki}) = -2(r_{ij} - p_{ij})u_{ki} .$$

$$u_{ki}^{t+1} = u_{ki}^{t} - \gamma \frac{\partial (e_{ij})^2}{\partial u_{ki}} = u_{ki}^{t} + 2\gamma(r_{ij} - p_{ij})m_{kj}^{t} .$$

$$u_{ki}^{t+1} = u_{ki}^{t} + 2\gamma(r_{ij} - p_{ij})m_{kj}^{t} ,$$

$$m_{kj}^{t+1} = m_{kj}^{t} + 2\gamma(r_{ij} - p_{ij})u_{ki}^{t} .$$

Gradient Descent (cont.)

Update u_{ki}:

$$u_{ki}^{t+1} = u_{ki}^{t} + \gamma(2(r_{ij} - p_{ij})m_{kj}^{t} - \lambda u_{ki}^{t}) \ ,$$

Update m_{kj} :

$$m_{kj}^{t+1} = m_{kj}^{t} + \gamma(2(r_{ij} - p_{ij})u_{ki}^{t} - \lambda m_{kj}^{t}) \ .$$

λ: regularize
parameter
γ: learning rate

5. Neural Colaborative Filtering

- Representing the user, product in a hidden aspect level may not fully capture the complex nature of the user-product interaction
- Deep neural networks allow automatic learning of latent aspect levels from basic to abstract
- MF is basic form of neural network

Limitations of MF

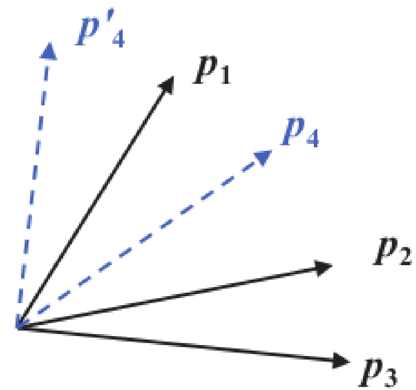
- MF preserve similarity of user vectors
- Assume that user similarity is measured using Jaccard metric
- User vector and item vector are both represented in the K-dimensional latent aspect space

Limitations MF (cont.)

- s_{23} (0.66) > s_{12} (0.5) > s_{13} (0.4)
- s_{41} (0.6) > s_{43} (0.4) > s_{42} (0.2)

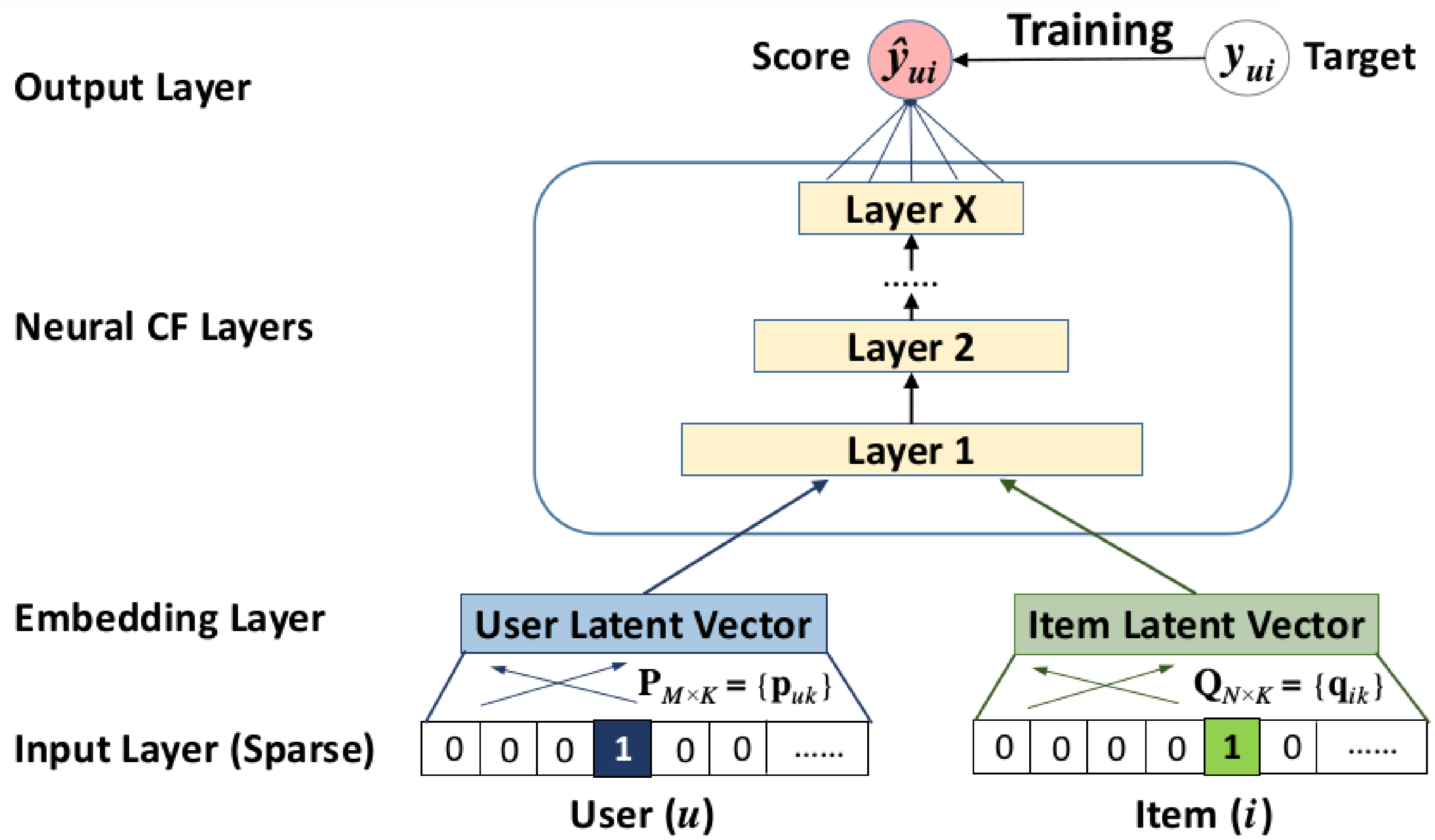
	i_1	i_2	i_3	i_4	i_5
u_1	1	1	1	0	1
u_2	0	1	1	0	0
u_3	0	1	1	1	0
u_4	1	0	1	1	1

items — users



from He et al. "*Neural Collaborative Filtering*". WWW 2017.

NCF architecture



Input layer

- The user is represented by a M-dimension one-hot vector
  - *M* is number of users
- Item are represented by a N-dimension one-hot vector where N is the number of products

Embedding layer

- Independently represent users and products
- M x K connection weight to represent user
- N x K connection weights to represent item
- E.g: $K = 16$

MLP

- User and item representations are feeded into a multi-layered MLP network to learn complex user and product interactions
- E.g:
  - *ReLU* activation function
  - 3 layers with deceasing size 32 → 16 → 8

Loss function

- Implicit feedback:
  - 1: user interacts with item
  - 0: user do not interacts with item
- Binary classification problem with probability factor
- Using the cross entropy loss function

$$-\sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} y_{ui}\log\hat{y}_{ui} + (1-y_{ui})\log(1-\hat{y}_{ui}).$$

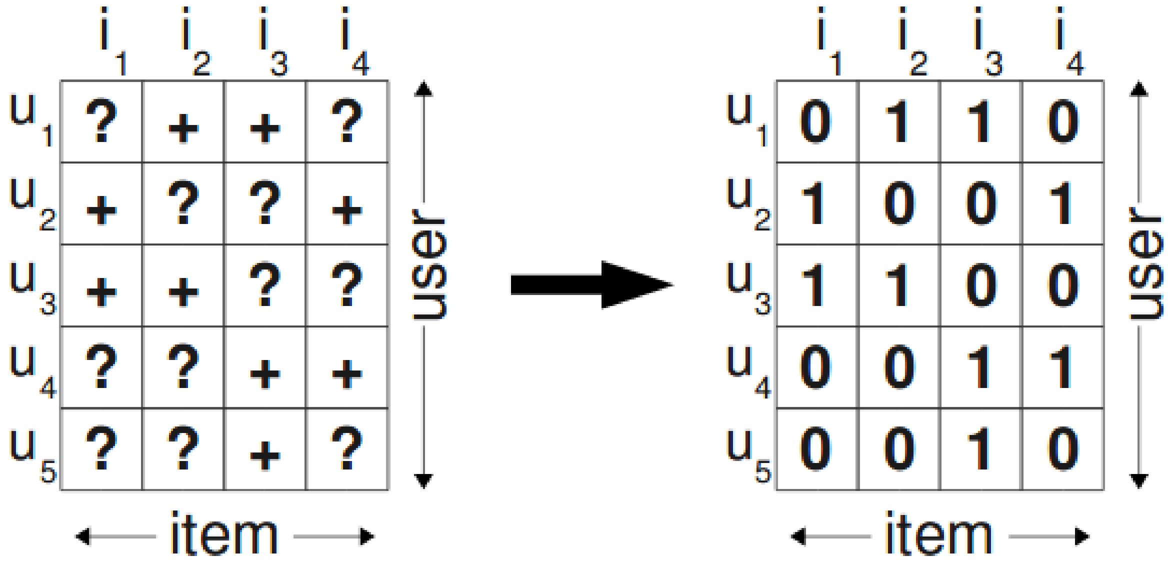
6. Session-based

- In many cases, it is difficult to identify users and collect reviews.
  - Small commercial websites
  - News sites
- session-based recommendation
  - Do not require user identification
  - Each transaction, including transaction order, is used as model training data

Problem definition

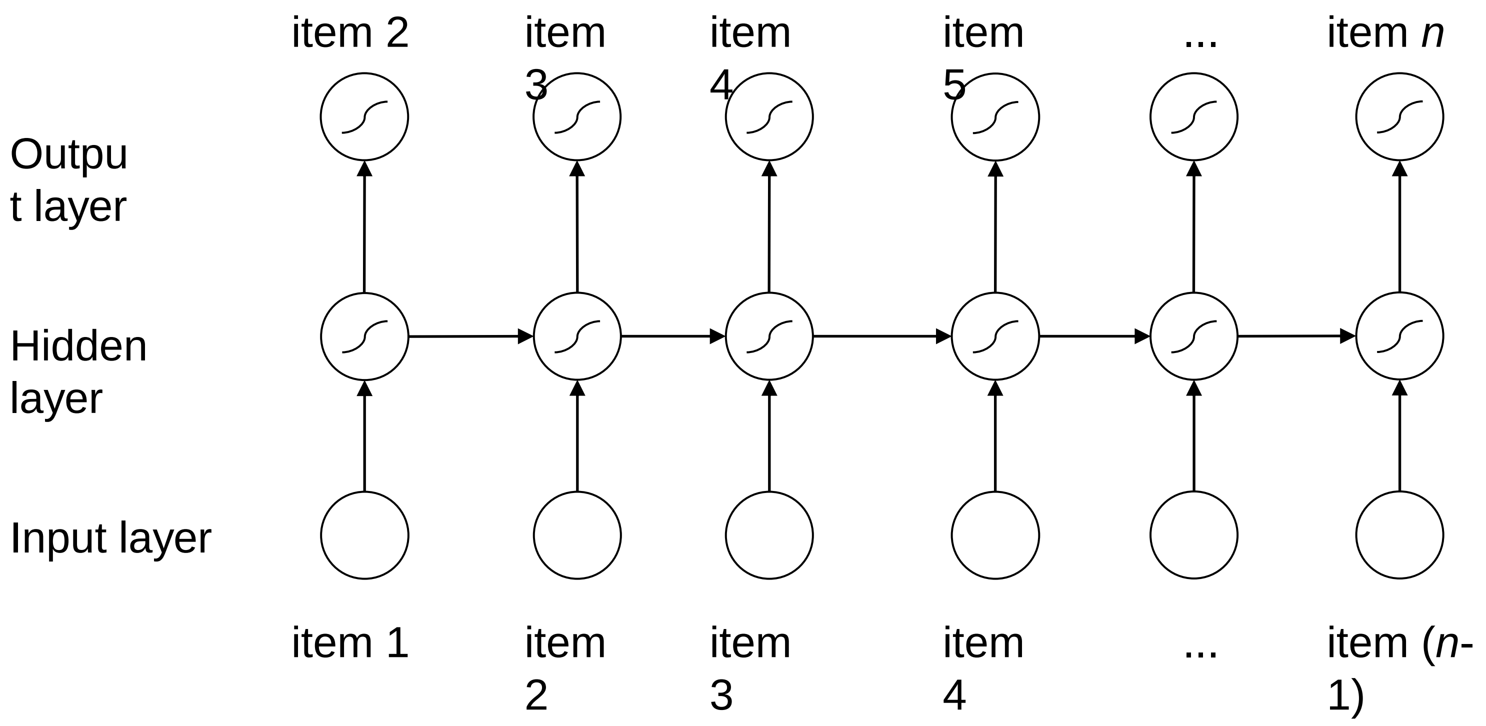
- Input: Stream of items $i_1, i_2, \ldots, i_{n-1}$
- Output: suggest next item
- Given a list of products with the highest probability $\Pr(i_n | i_1, i_2, \ldots, i_{n-1})$
- Suitable for implicit feedback

Implicit feedback



from RENDLE ET AL. "*BPR: Bayesian Personalized Ranking from Implicit Feedback*". 2009

RNN architecture



Input layer

- Represent each item at a time t as V-dimension one-hot vector
  - 1 at position of item, otherwise 0
  - V: number of distinct

Embedding layer

- Biến đổi biểu diễn one-hot thành biểu diễn *K* chiều
  - *K* number of neural in embedding layer
  - Number of connection weight between input and embedding layer *V* x *K*

Recurrent layer

- The recurrent layer stores historical information through recurrent links
- Many recurrent layer stacked together to learn high level abstract features
- Advanced model: LSTM or GRU

MLP layer

- The output of the recurrent layer is used as the input of the MLP layer to generate the prediction
- The MLP layer can include hidden layers to learn the nonlinear function
- The MLP has V neurons corresponding to V products

Loss function

- Pair-wise rank loss function

$$L = -\frac{1}{N}\sum_{j=1}^{N} log(\sigma(\hat{r}_i - \hat{r}_j))$$

- *N*: number of negative sample
- $\hat{r}_i$: ratings of positive sample *i*
- $\hat{r}_j$: ratings of negative sample *j*
- Implicit feedback hypothesis: item i is selected so i's priority is higher than any other item j

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!