Recommender Systems

程 健 研究员

jcheng@nlpr.ia.ac.cn

中国科学院自动化研究所
模式识别国家重点实验室

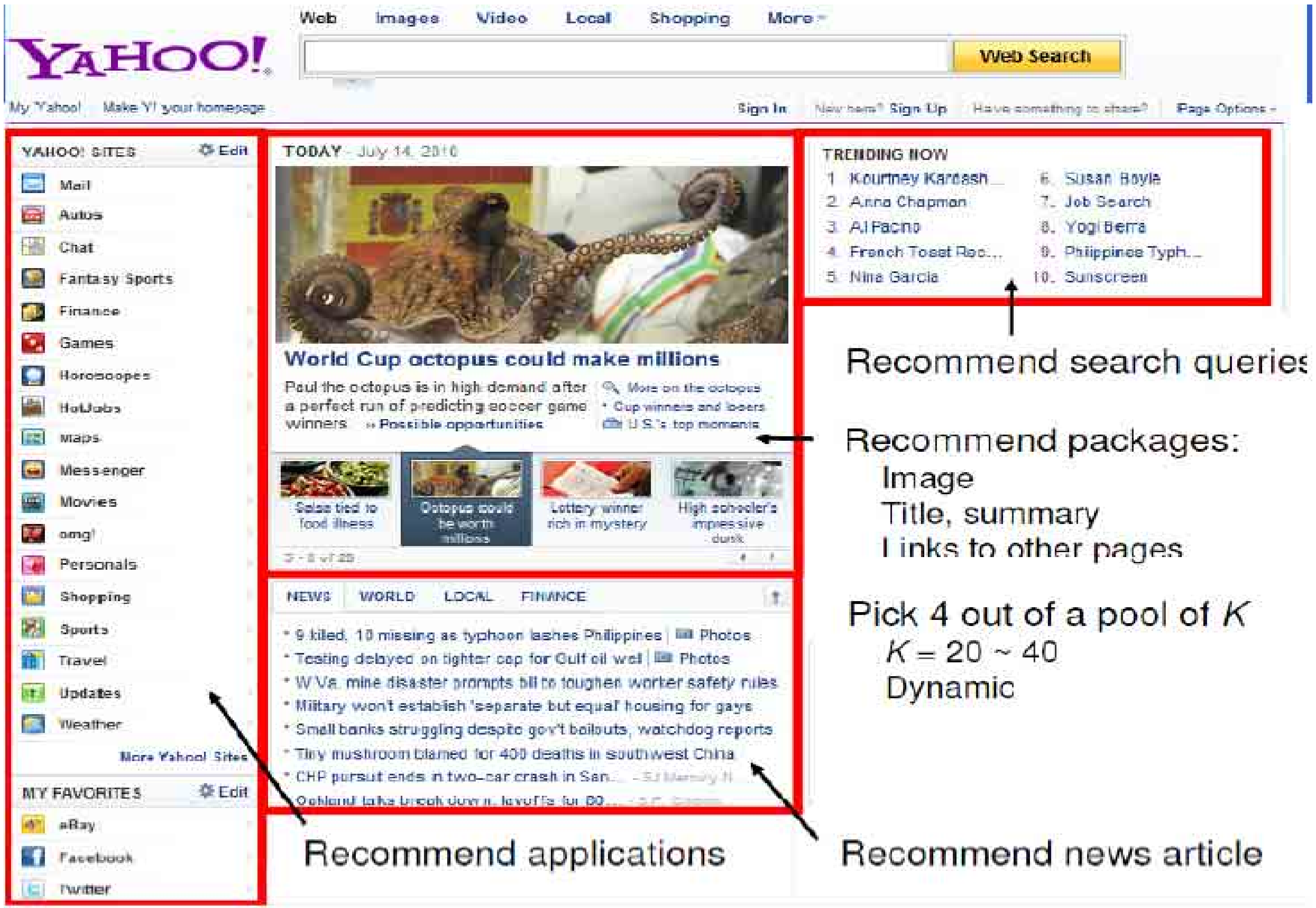
Index

- The Recommender Problem
- Traditional Methods
- Beyond Traditional Methods
- RS Systems
- Summary
- References

Index

- **<u>The Recommender Problem</u>**
- Traditional Methods
- Beyond Traditional Methods
- RS Systems
- Summary
- References

Everything is personalized



Everything is personalized

京东商城

Everything is personalized

Xavier Amatriain – August 2014 – KDD

Everything is personalized

- 视频网站

  YouTube 土豆 Hulu 奇艺视频 等

- 电子商务网站

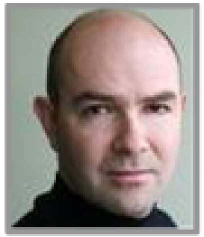
  淘宝，亚马逊等

- 社交网站

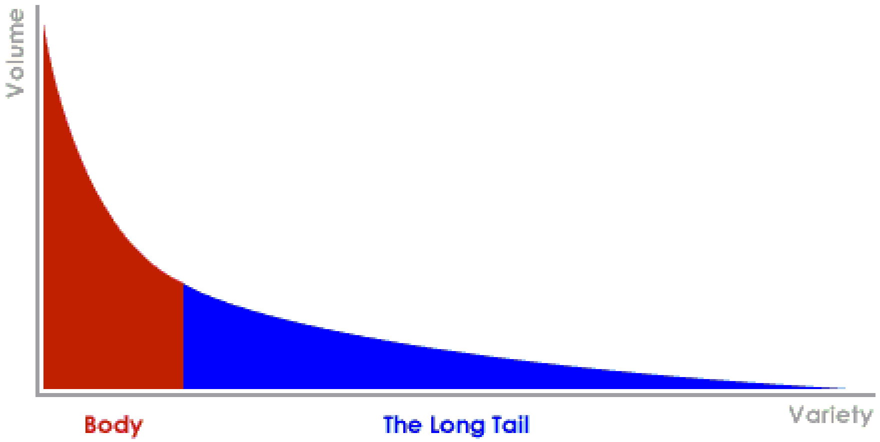
  Facebook 人人网 Twitter 微博 等

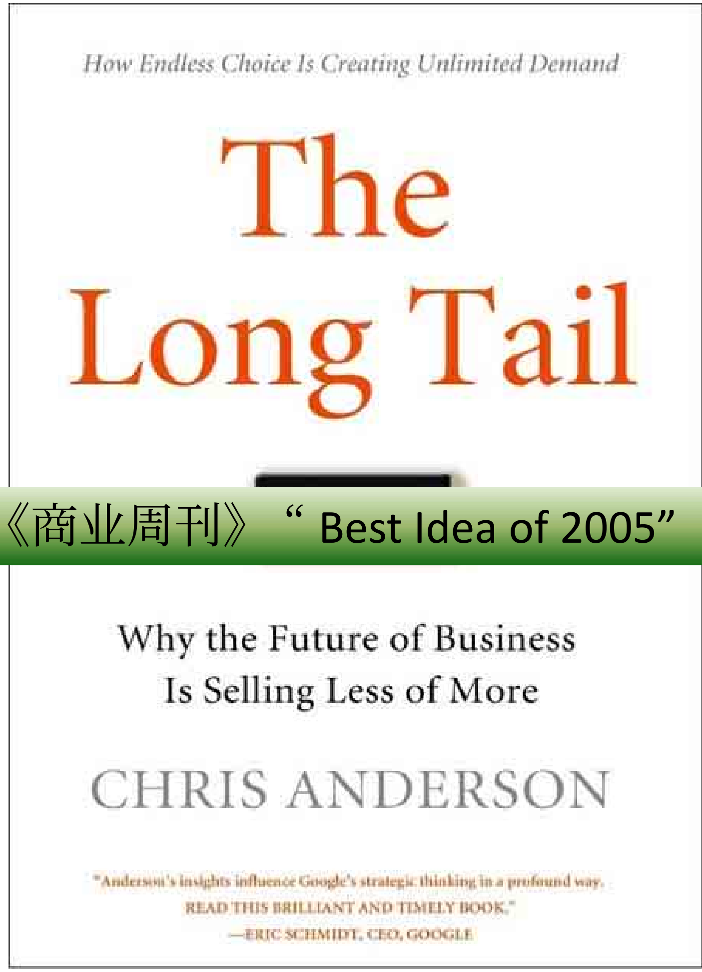
- ……

The Long Tail



Chris Anderson's
Web 2.0 Business Model:
The Long Tail





《商业周刊》“ Best Idea of 2005”

The Long Tail

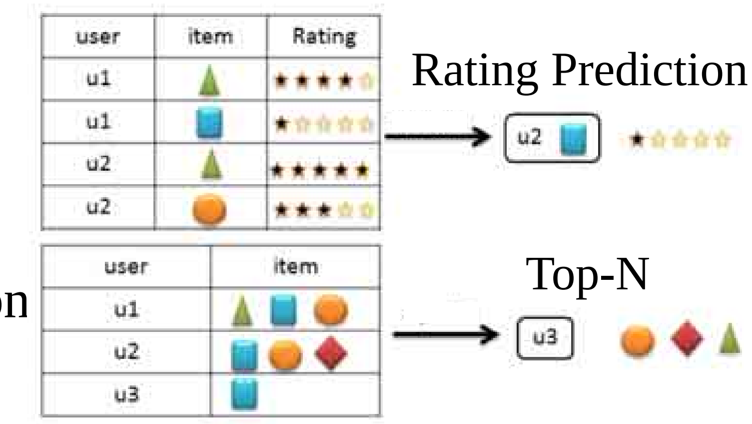
- Amazon: 35% 的销售来自推荐
- Google News: 推荐增加了38%的点击率
- Netflix:2/3的电影出租来自推荐

"We are leaving the age of information and entering the Age of Recommendation"

– The Long Tail (Chris Anderson)

RS Definition

- RS seen as a function
- Given:
  - ➢ User model (e.g. ratings, preferences, demographics, situational context)
  - ➢ Items (with or without description of item characteristics)
- Calculate:
  - ➢ Relevance score used for ranking
- Target:
  - ➢ Rating Prediction & Top-N Recommendation



- But:
  - ➢ Remember that relevance might be context dependent
  - ➢ Characteristics of the list itself might be important (diversity)

Performance Evaluation

- Measures for rating prediction
  - Mean absolute error

$$MAE = \frac{1}{|Test|} \times \sum_{(u,i) \in Test} |\hat{r}_{u,i} - r_{u,i}|$$

  - Root mean square error

$$RMSE = \sqrt{\frac{\sum_{(u,i) \in Test} (\hat{r}_{u,i} - r_{u,i})^2}{|Test|}}$$

Performance Evaluation

- Measures for top-N recommendation
  - NDCG(Normalized Discounted Cumulative Gain)

$$\mathrm{DCG_p} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2(i)}$$

定义不唯一

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(1+i)}$$

$$NDCG_p = \frac{DCG_p}{IDCG_p}$$

Ideal DCG

  - F_1 Score

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

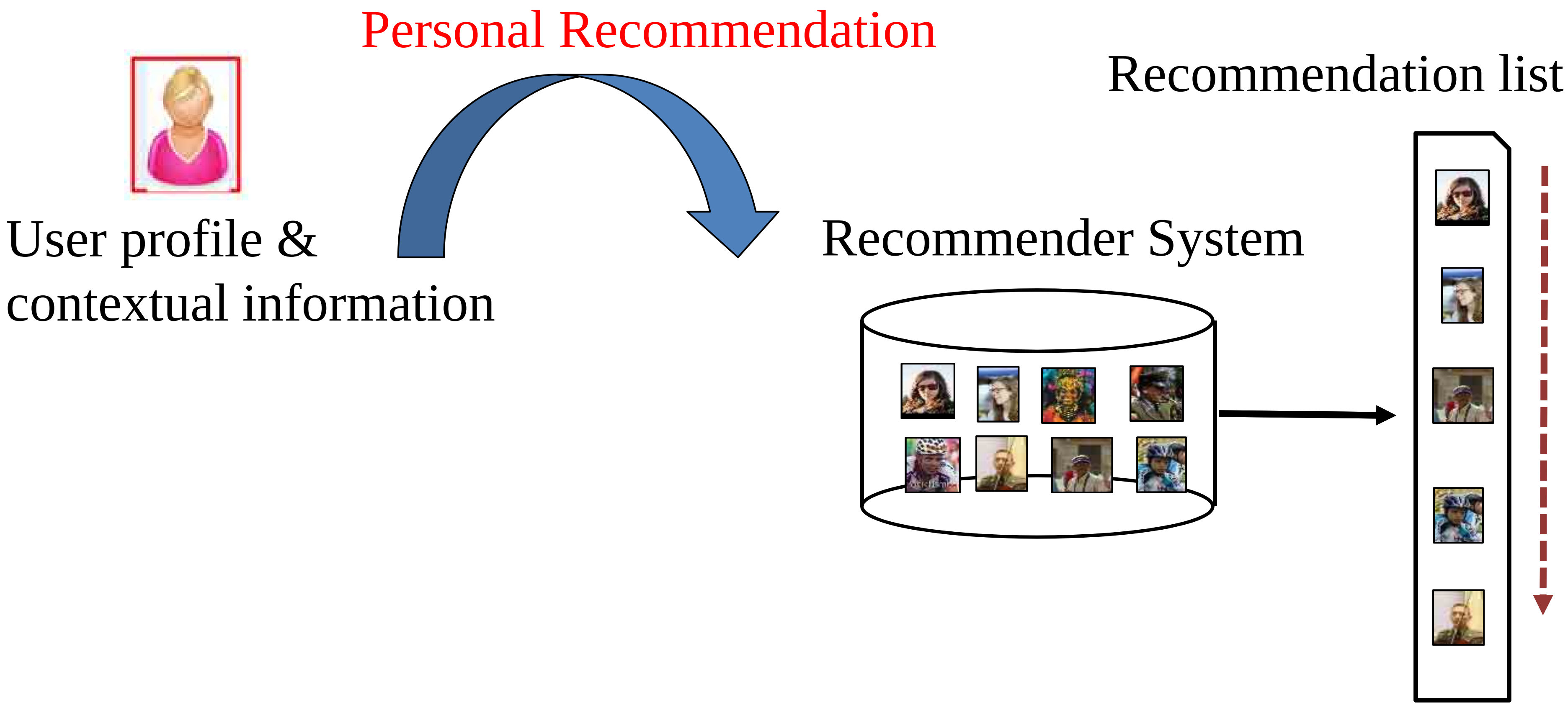
Key Problems

- Data sparsity：
  - Netflix Dataset: nearly 48,000 users and 1,700 items, only 1% observations
- Curse of dimensionality
  - Users' features can be represented as many ways
- Cold start:
  - Many new users sign in and many new items are added
- Personalization:
  - Different user has different taste

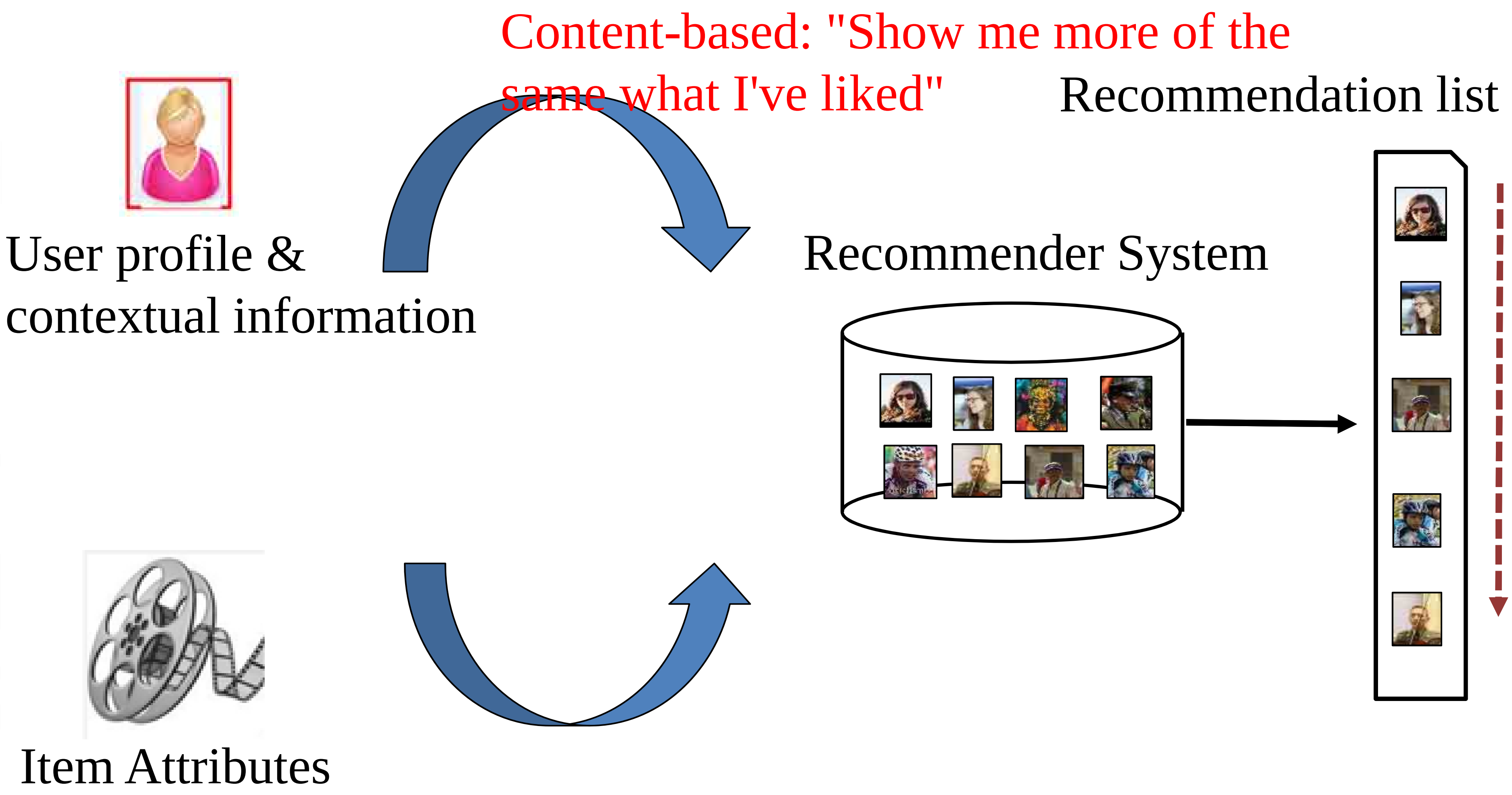
Index

- The Recommender Problem
- **Traditional Methods**
- Beyond Traditional Methods
- RS Systems
- Summary
- References

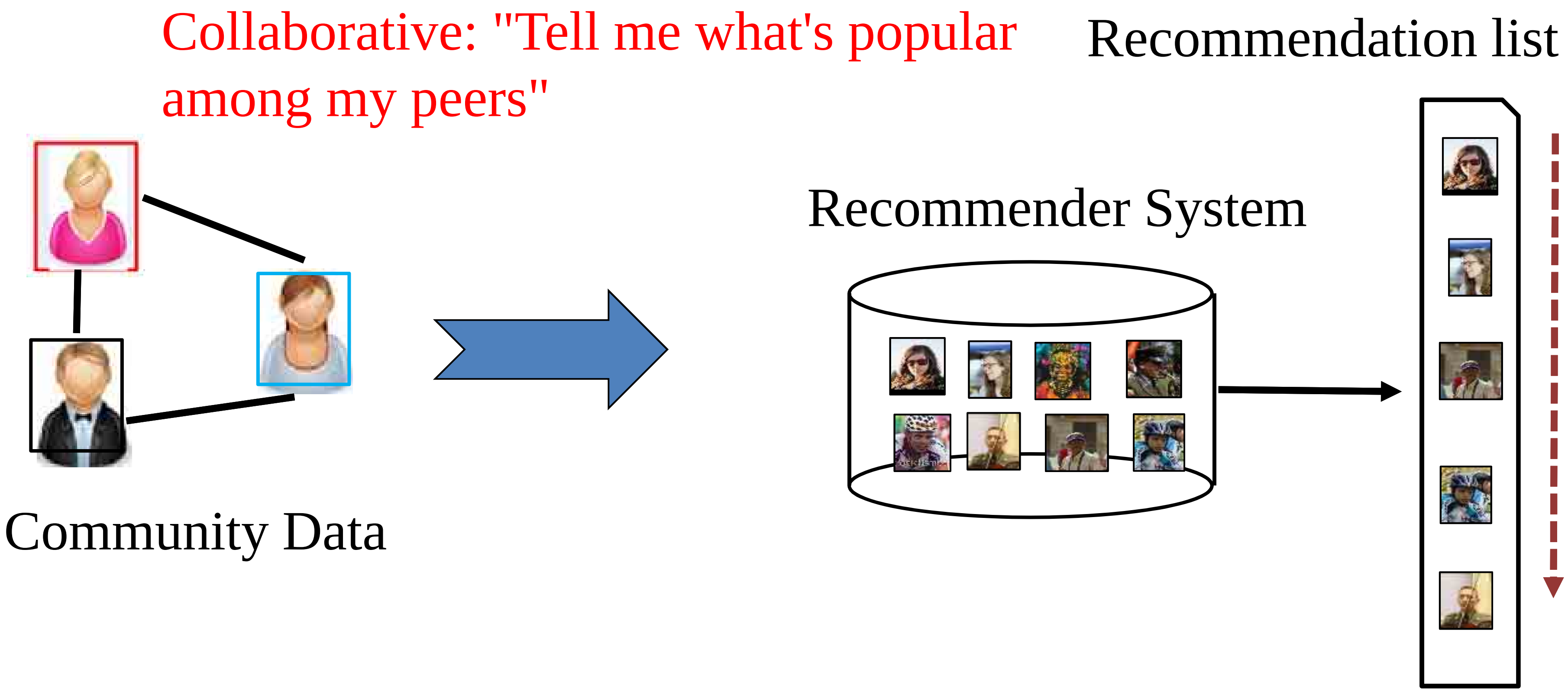
A glance of Paradigms for RS



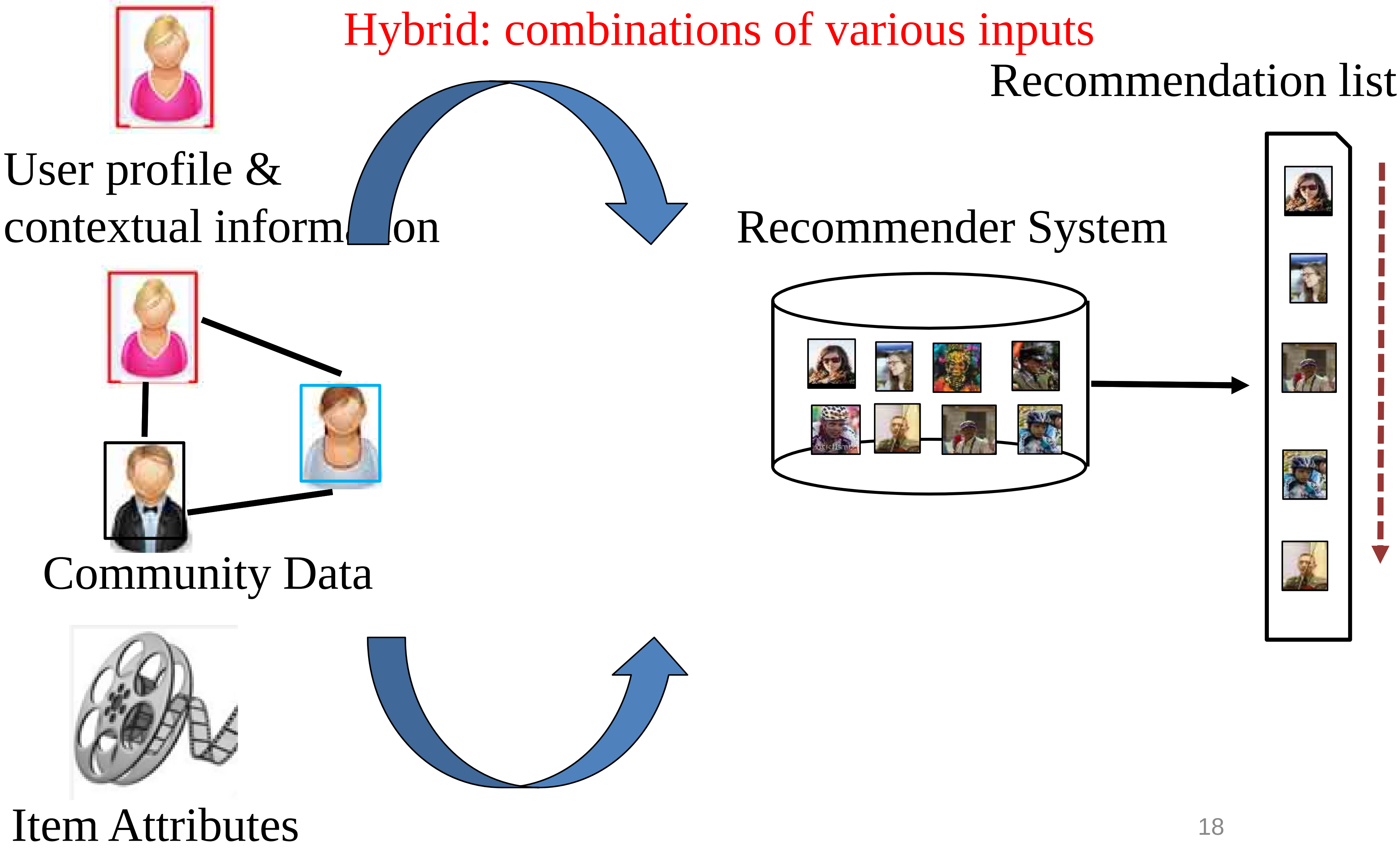
A glance of Paradigms for RS



A glance of Paradigms for RS



A glance of Paradigms for RS



Content-Based Recommendation

- Recommendations based on content of items rather than on other users' opinions/interactions
- Goal: recommend items similar to those the user liked
- Common for recommending text-based products (web pages, news messages)
- Items to recommend are "described" by their associated features (e.g. keywords)
- User Model structured in a "similar" way as the content: features/keywords more likely to occur in the preferred documents (lazy approach)
- The user model can be a classifier based on whatever technique (Neural Networks, Naïve Bayes...)

Content-Based Recommendation

- Content representation and item similarities

Express item features as:

- TF-IDF
- N-Gram
- LDA
- Word2Vec

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, Murder, Neo-nazism
...					

Title	Genre	Author	Type	Price	Keywords
...	Fiction, Suspense	Brunonia Barry, Ken Follet, ..	Paperback	25.65	detective, murder, New York

- Compute the similarity of an unseen item with the user profile based on the keyword features

Content-Based Recommendation

- Pros:
  - No need for data on other users: No cold-start or sparsity
  - Able to recommend to users with unique tastes
  - Able to recommend new and unpopular items
  - Can provide explanations by listing content-features

- Cons:
  - Requires content that can be encoded as meaningful features (difficult in some domains/catalogs)
  - Users represented as learnable function of content features
  - Difficult to implement serendipity
  - Easy to overfit (e.g. for a user with few data points)

Collaborative Filtering

- List of m Users and a list of n Items

- Each user has a list of items with associated opinion

opinion $\begin{cases} \text{Explicit (e.g. ratings)} \\ \text{Implicit (e.g. purchase records)} \end{cases}$

- Active user for whom the CF prediction task is performed

- Metric for measuring similarity between users

- Method for selecting a subset of neighbors

- Method for predicting a rating for items not currently rated by the active user.

Collaborative Filtering

users

items	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3	?		5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Collaborative Filtering

- memory-based CF
  - User-based CF
  - Item-based CF
- model-based CF
  - First develop a model of user
  - Type of model:
    1. Probabilistic (e.g. Bayesian Network)
    2. Clustering
    3. Rule-based approaches (e.g. Association Rules)
    4. Classification/Regression
    5. …

User-based CF

The basic steps:

1. Identify set of ratings for the target/active user
2. Identify set of users most similar to the target/active user according to a similarity function (neighborhood formation)
3. Identify the products these similar users liked
4. Generate a prediction
5. Based on this predicted rating recommend a set of top N products

User-based CF

- A collection of user $u_i, i = 1 \ldots \mathrm{m}$ and a collection of products $p_j, j = 1, \ldots, \mathrm{n}$
- An m $\times$ n matrix of ratings , with $r_{ij} = ?$ if user i did not rate product j
- Prediction for user i and product j is computed as $r_{ij}^{\,*} = K \sum_{r_{kj \neq ?}} u_{jk} r_{kj}$
- Similarity can be computed by Pearson correlation

$$u_{ik} = \frac{\sum_j (\mathrm{r}_{ij} - \mathrm{r}_i)(\mathrm{r}_{kj} - \mathrm{r}_k)}{\sqrt{\sum_j (\mathrm{r}_{ij} - \mathrm{r}_i)^2 \sum_j (\mathrm{r}_{kj} - \mathrm{r}_k)^2}}$$

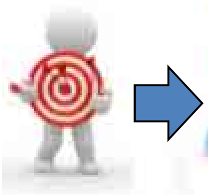
User-based CF Example

						sim(u,v)
2		2	4	5		NA
5		4			1	
		5		2		
	1		5		4	
		4			2	
4	5		1			NA

User-based CF Example

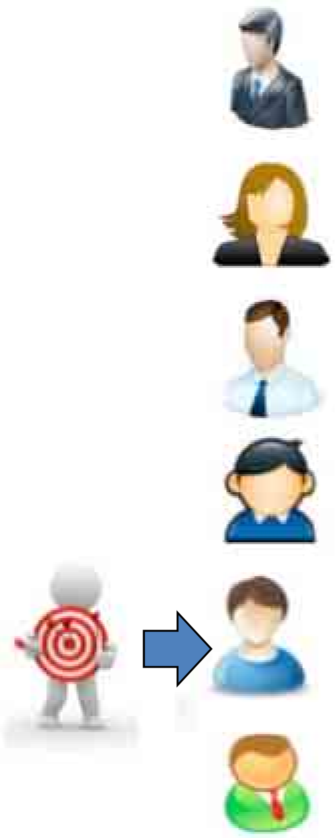
							sim(u,v)
	2		2	4	5		NA
	5		4			1	0.87
			5		2		
		1		5		4	
			4			2	
	4	5		1			NA

User-based CF Example



						sim(u,v)
2		2	4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	
		4			2	
4	5		1			NA

User-based CF Example



						sim(u,v)
2		2	4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	-1
		4			2	
4	5		1			NA

Item-based CF Example

The basic steps:

1. Look into the items the target user has rated
2. Compute how similar they are to the target item
3. Select k most similar items
4. Compute Prediction by taking weighted average on the target user’s ratings on the most similar items

Item Similarity Computation

- Similarity: find users who have rated items and apply a similarity function to their ratings

- Cosine-based Similarity (difference in rating scale between users is not taken into account)

$$sim(\mathrm{a}, \mathrm{b}) = \frac{a \bullet b}{|\mathrm{a}| \times |b|}$$

- Adjusted Cosine Similarity (takes care of difference in rating scale)

$$S(\mathrm{i}, \mathrm{j}) = \frac{\sum_{u} (\mathrm{r}_{ui} - \mathrm{r}_{u})(\mathrm{r}_{uj} - \mathrm{r}_{u})}{\sqrt{\sum_{u} (\mathrm{r}_{ui} - \mathrm{r}_{u})^{2} \sum_{u} (\mathrm{r}_{uj} - \mathrm{r}_{u})^{2}}}$$

Item Similarity Computation

- Alternative similarity metric

Correlation based	Cosine, Pearson Correlation, Adjusted Cosine,OLS coefficient
Distance based	Euclidean distance, Manhattan distance,Minkowski distance
Hash based	Mini Hash, Sim Hash
Topic based	PLSA, LDA
Graph based	Shortest Path, Random Walk, Item Rank

Model-based CF

Motivated by Netflix Prize (launched in Oct. 2006)

- Task:
  High quality recommendations
  for cinematch (RMSE=0.9525)
- Dataset:
  users: 480,000
  movies: 17,770
  rates ratio <1%

Improve by 10% = $1million!

Model-based CF

Motivated by Netflix Prize (launched in Oct. 2006)

- Measure:

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Model-based CF

Leaderboard

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43

Model-based CF

2009 Netflix Prize Results

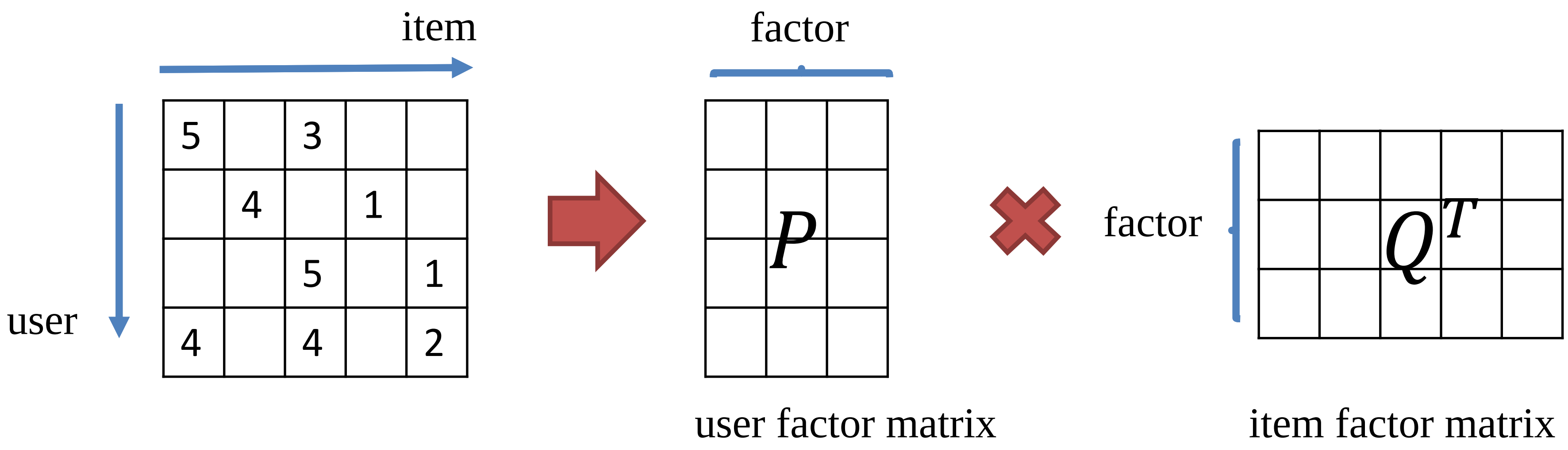
- Top 2 single algorithms:
  SVD/MF - Prize RMSE: 0.8914
  RBM - Prize RMSE: 0.8990

- Linear blend Prize RMSE: 0.88

- Currently in use as part of Netflix' rating prediction component

Matrix Factorization

- Basic idea



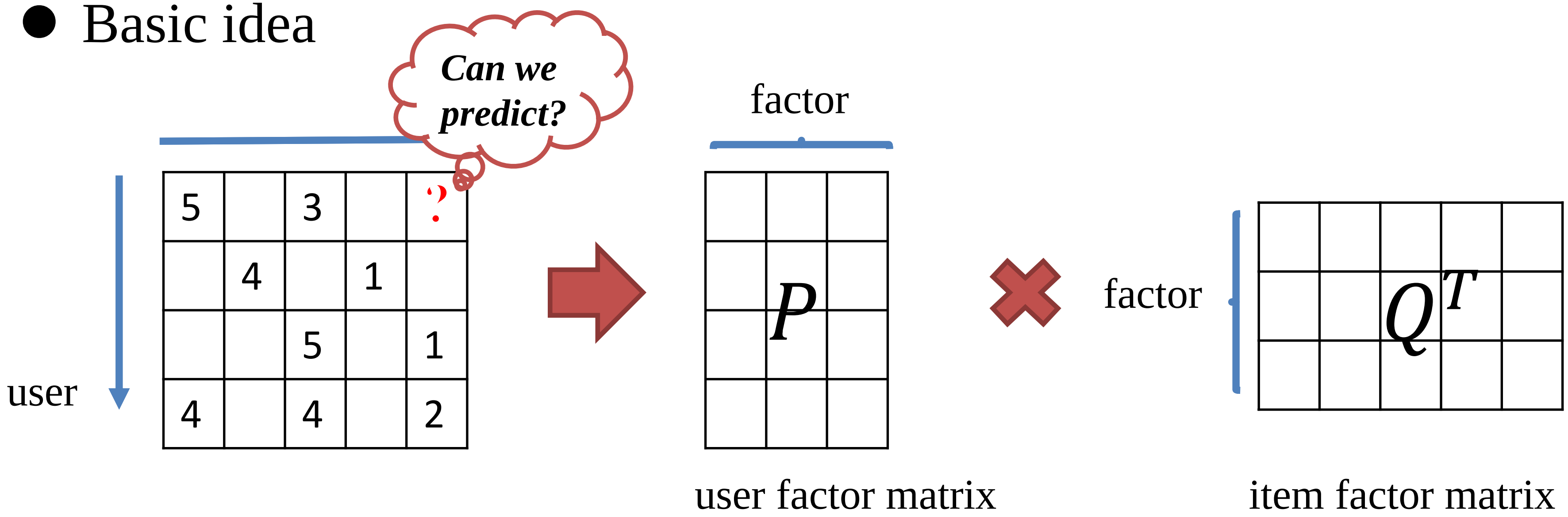
- factor size << dim of user/item

p_u	f_1	f_2	f_3	…	f_k
q_v	f_1'	f_2'	f_3'	…	f_k'

- User factor vectors $p_u \in R^f$ and item factor vector $q_v \in R^f$

Matrix Factorization

- Basic idea



- factor size << dim of user/item

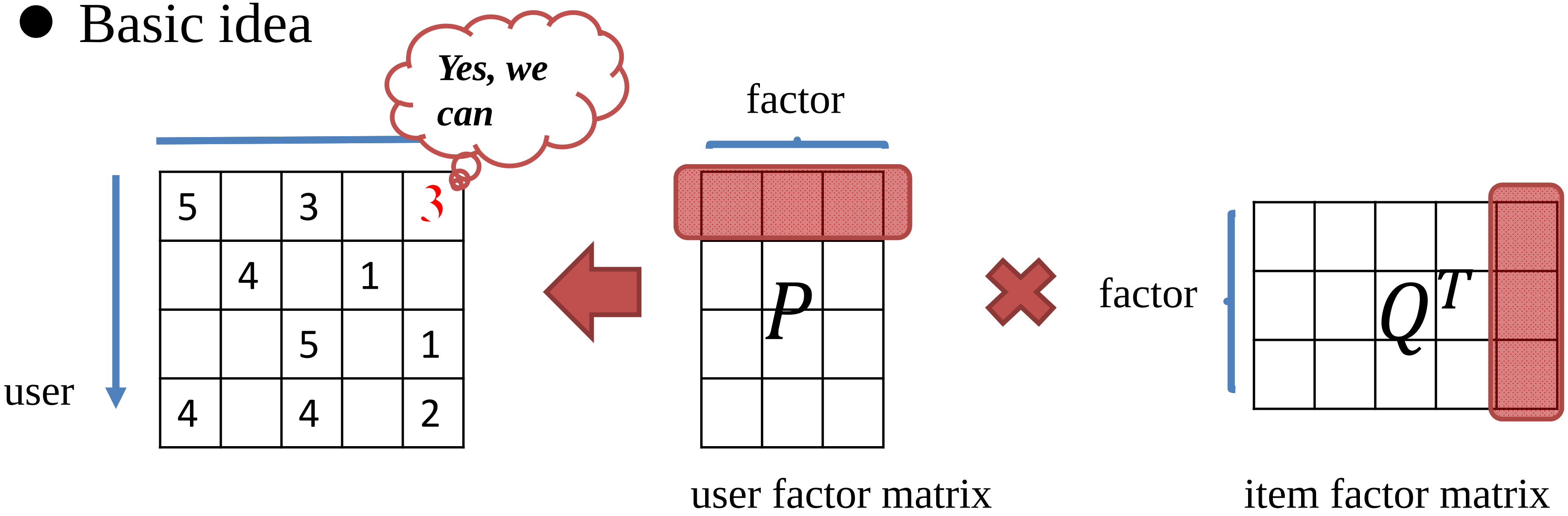
$$p_u \quad \boxed{f_1 \mid f_2 \mid f_3 \mid \dots \mid f_k}$$

$$q_v \quad \boxed{f'_1 \mid f'_2 \mid f'_3 \mid \dots \mid f'_k}$$

- User factor vectors $p_u \in R^f$ and item factor vector $q_v \in R^f$

Matrix Factorization

- Basic idea



- factor size << dim of user/item

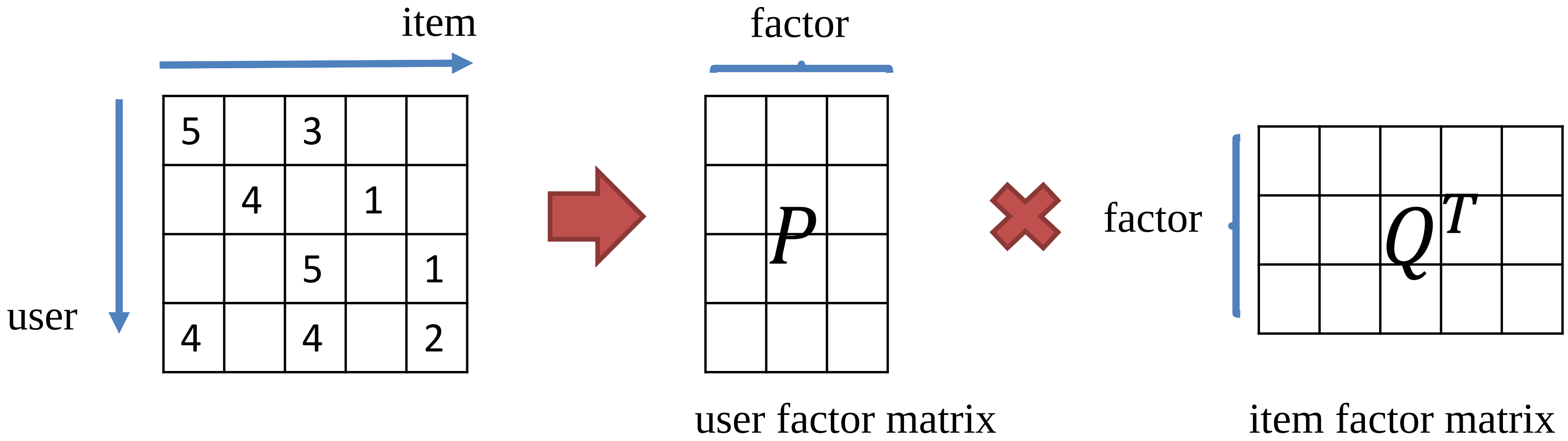
$$p_u \; \boxed{f_1 \mid f_2 \mid f_3 \mid \dots \mid f_k}$$

$$q_v \; \boxed{f'_1 \mid f'_2 \mid f'_3 \mid \dots \mid f'_k}$$

- User factor vectors $p_u \in R^f$ and item factor vector $q_v \in R^f$

Non-negative Matrix Factorization

- Both entries in factorized P and Q should be >= 0



- Explanation: real world data, i.e. images, has often been represented as non-negative values, while negative ones doesn't have any meanings.

$$p_u \; \begin{array}{|c|c|c|c|c|}\hline f_1 & f_2 & f_3 & \cdots & f_k \\ \hline \end{array} \; >=0$$

$$q_v \; \begin{array}{|c|c|c|c|c|}\hline f_1' & f_2' & f_3' & \cdots & f_k' \\ \hline \end{array} \; >=0$$

Non-negative Matrix Factorization

- ‘Orthogonal NMF’ == ‘Kernel K-Means Clustering’

Orthogonal NMF

$$\min_{F,G} \| X - FG^T \|_F^2, \text{s.t.} G^T G = I, G \geq 0$$

is equivalent to K-means clustering. Where each row of $G \in R^{n \times r}$ can be viewed as a probability distribution of the factors(clusters) .

Proof

1. Kernel K-means clustering tries to minimize $J = \sum_{k=1}^{K} \sum_{i \in C_k} \| \mathrm{x}_i - m_k \|^2$

By utilizing an indicator matrix $G = (\mathrm{g}_1, \ldots, \mathrm{g}_K), \mathrm{g}_k^T g_l = \delta_{kl}$, where

$g_k = (0,..,0,1,...,1,0,...,0)^T / n_k^{1/2}$, the above formulation can be transformed to

$$\max J(G) = \max Tr(G^T XXG), \text{s.t.} G^T G = I, G \geq 0$$

Non-negative Matrix Factorization

- ‘Orthogonal NMF’ == ‘Kernel K-Means Clustering’

2. We write the NMF formulation as

$$J = \| X - FG^T \|_F^2 = Tr(X^T X - 2F^T XG + F^T F)$$

the zero gradient condition $\partial J / \partial F = -2XG + 2F = 0$, given $F = XG$ then $J = Tr(X^T X - G^T XXG)$, the optimization can also be transformed to

$$\min_G Tr(-G^T XXG), \mathrm{s.t.} G^T G = I, G \geq 0$$

Further transform to

$$\max_G Tr(G^T XXG), \mathrm{s.t.} G^T G = I, G \geq 0$$

Which has the same form as Kernel K-means clustering

SVD for Rating Prediction

However,
- Some items are significantly higher rated…
- Some users rate substantially lower…
- All Ratings are high…

Thus,
- Add item offset…
- Add user offset…
- Add global offset…

- Baseline (bias) $b_{uv} = \mu + b_u + b_v$ (user & item deviation from average)
- Predict rating as $\hat{r}_{uv} = b_{uv} + p_u^T q_v$

SVD for Rating Prediction

- In order to prevent over-fitted problem, we add some regularized terms, such as:

$$SSE = \frac{1}{2}(\mathrm{r}_{uv} - \widehat{\mathrm{r}}_{uv})^2 + \lambda\left(\sum_u |\mathrm{p}_u|^2 + \sum_v |\mathrm{q}_v|^2\right)$$

- SVD++ asymmetric variation with implicit feedback

$$\hat{r}_{uv} = b_{uv} + q_v^{\,T}\left(|\mathrm{R(u)}|^{-\frac{1}{2}} \sum_{j \in R(\mathrm{u})} (\mathrm{r}_{uj} - \mathrm{b}_{uj})\,\mathrm{x}_j + |N(\mathrm{u})|^{-\frac{1}{2}} \sum_{j \in N(\mathrm{u})} y_j\right)$$

Where

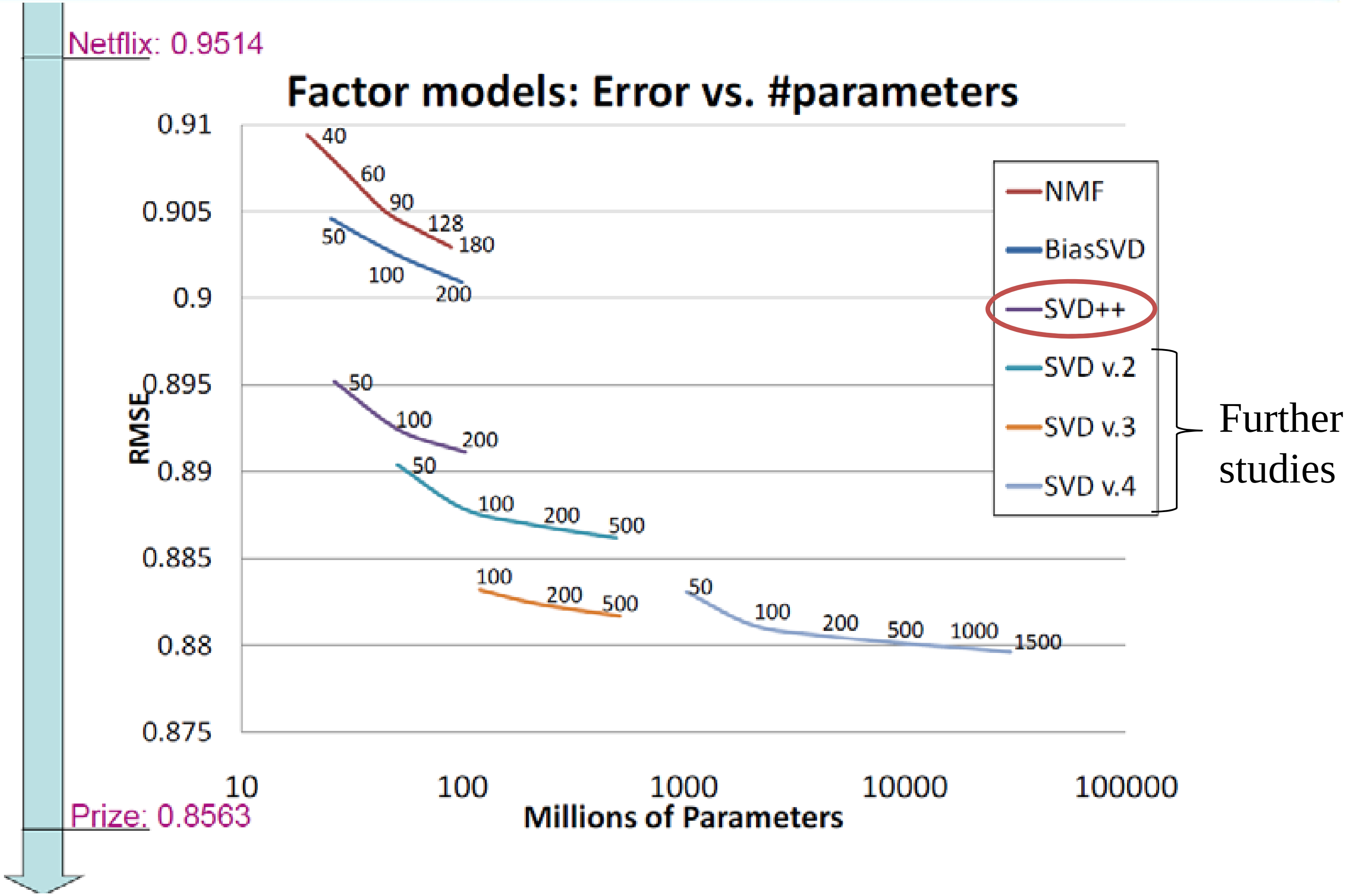
$q_v, x_v, y_v \in R^f$ are three item factor vectors

$\mathrm{R(u)}$ items rated by user u

$N(\mathrm{u})$ items for which the user has given implicit preference

Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems[J]. Computer, 2009 (8): 30-37.

SVD for Rating Prediction



Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems[J]. Computer, 2009 (8): 30-37.

Probabilistic Matrix Factorization

- From the view of probability to predict ratings, we assume factorized vectors of users and items are in line with the Gaussian distribution, user's preference for items is a combination of the probability of a series of problems, such as

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|U_i^T V_j, \sigma^2)\right]^{I_{ij}}$$

where

$$p(U|\sigma_U^2) = \prod_{i=1}^{N}\mathcal{N}(U_i|0, \sigma_U^2\mathbf{I}), \qquad p(V|\sigma_V^2) = \prod_{j=1}^{M}\mathcal{N}(V_j|0, \sigma_V^2\mathbf{I}).$$

Salakhutdinov R, Mnih A. Probabilistic matrix factorization. NIPS, 2008.

Probabilistic Matrix Factorization

- By adding regularized terms, the formulation can be shown as:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} \| U_i \|_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \| V_j \|_{Fro}^2,$$

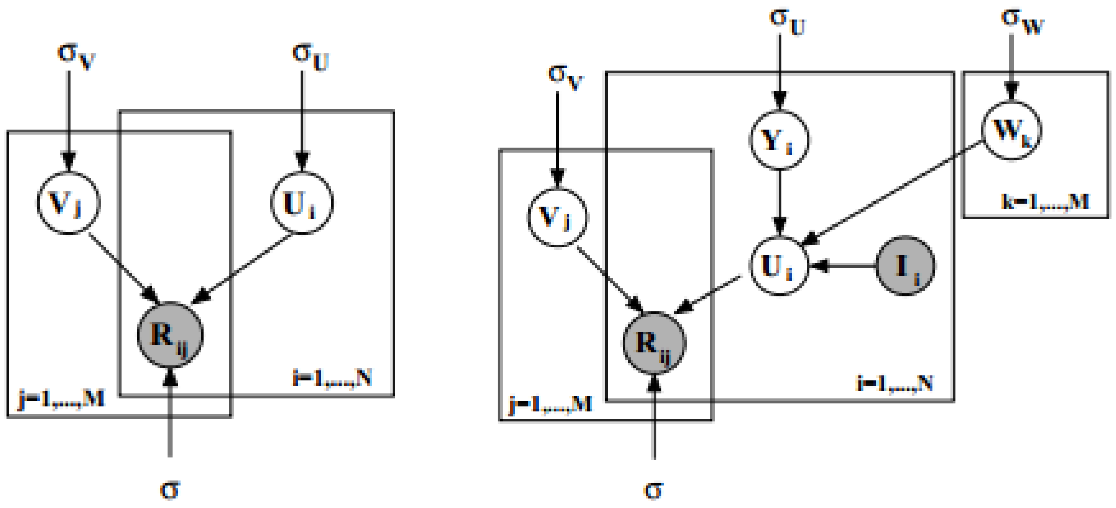


Figure 1: The left panel shows the graphical model for Probabilistic Matrix Factorization (PMF). The right panel shows the graphical model for constrained PMF.

Salakhutdinov R, Mnih A. Probabilistic matrix factorization. NIPS, 2008.

Probabilistic Matrix Factorization

- In order to normalize the scores (i.e. 1-5), the paper uses the following approach

$$g(x) = 1/(1+\exp(-x))$$

- Thus, the final formulation can be written as:

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|g(U_i^T V_j),\sigma^2)\right]^{I_{ij}}$$

- The implementation is adopted Gibbs Sampling Strategy

Salakhutdinov R, Mnih A. Probabilistic matrix factorization. NIPS, 2008.

Probabilistic Matrix Factorization

- Once a PMF model has been fitted, users with very few ratings will have feature vectors that are close to the prior mean so the predicted ratings for those users will be close to the movie average ratings.
- Let $W \in R^{D \times M}$ be a latent similarity constraint matrix. We define the feature vector for user i as

$$U_i = Y_i + \frac{\sum_{k=1}^{M} I_{ik} W_k}{\sum_{k=1}^{M} I_{ik}}.$$

- The corresponding Constrained PMF formulation can be shown as:

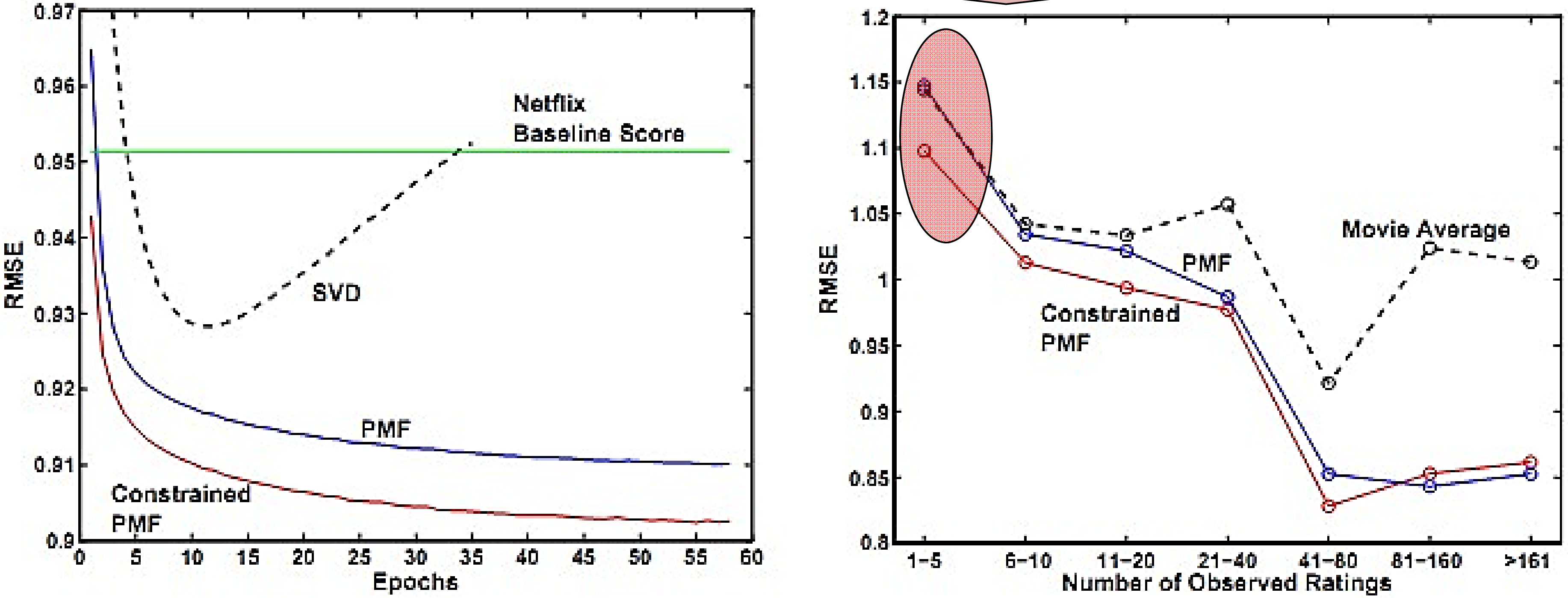
$$p(R|Y,V,W,\sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}\left(R_{ij} | g\left(\left[Y_i + \frac{\sum_{k=1}^{M} I_{ik} W_k}{\sum_{k=1}^{M} I_{ik}}\right]^T V_j\right), \sigma^2\right)\right]^{I_{ij}}$$

Salakhutdinov R, Mnih A. Probabilistic matrix factorization. NIPS, 2008.

Probabilistic Matrix Factorization

- Experiments

Very helpful for infrequent users



Salakhutdinov R, Mnih A. Probabilistic matrix factorization. NIPS, 2008.

Self-Representation Model

- Beyond matrix factorization, there is another form of modeling users'preference:

item

user

A

5		3		
	4		1	
		5		1
4		4		2

A

5		3		
	4		1	
		5		1
4		4		2

original matrix

W

0	.2	0	1	.3
.1	0	.7	0	.4
.3	.8	0	0	.1
0	0	0	0	.2
.6	0	.3	0	0

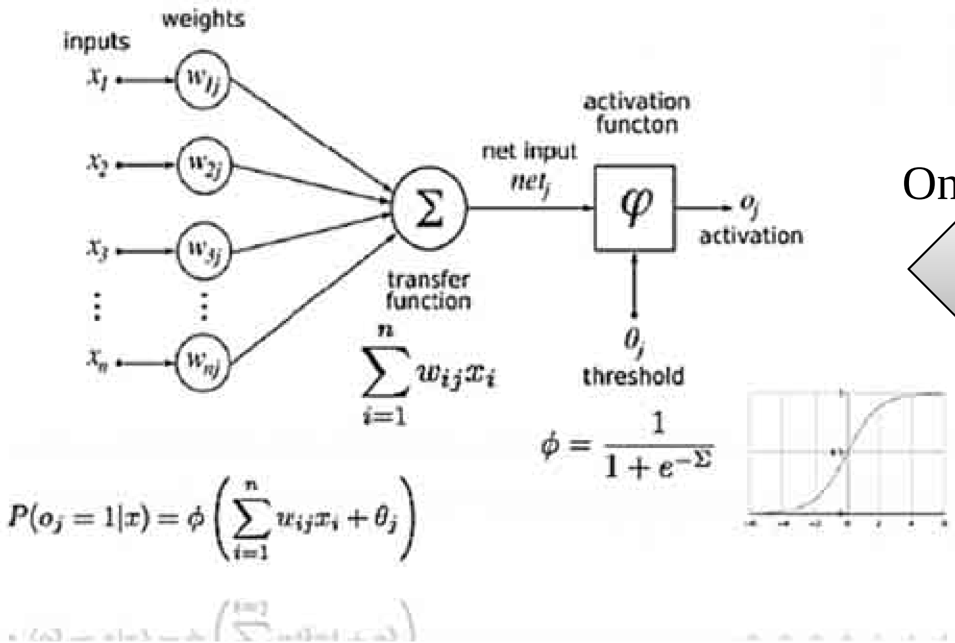
weight matrix

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & \frac{1}{2}\|A - AW\|_F^2 + \frac{\beta}{2}\|W\|_F^2 + \lambda\|W\|_1 \\ \text{subject to} \quad & W \geq 0 \\ & \operatorname{diag}(W) = 0, \end{aligned}$$

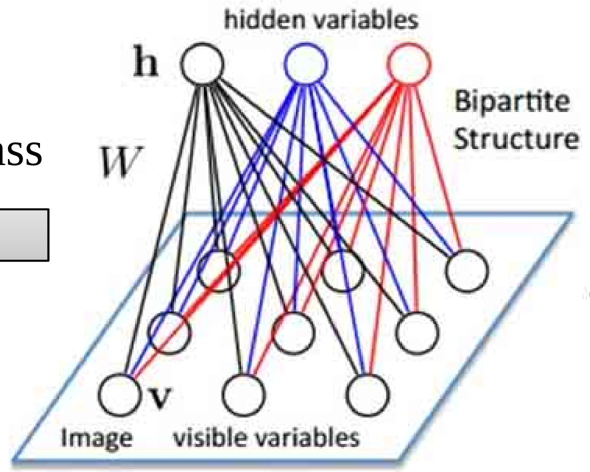
Ning X, Karypis G. Slim: Sparse linear methods for top-n recommender systems[C] ICDM 2011

Restricted Boltzmann Machines

- Each unit is a state that can be active or not active
- Each input to a unit is associated to a weight
- The transfer function $\sum$ calculates a score for every unit based on the weighted sum of inputs
- Score is passed to the activation function φ that calculates the probability of the unit to be active

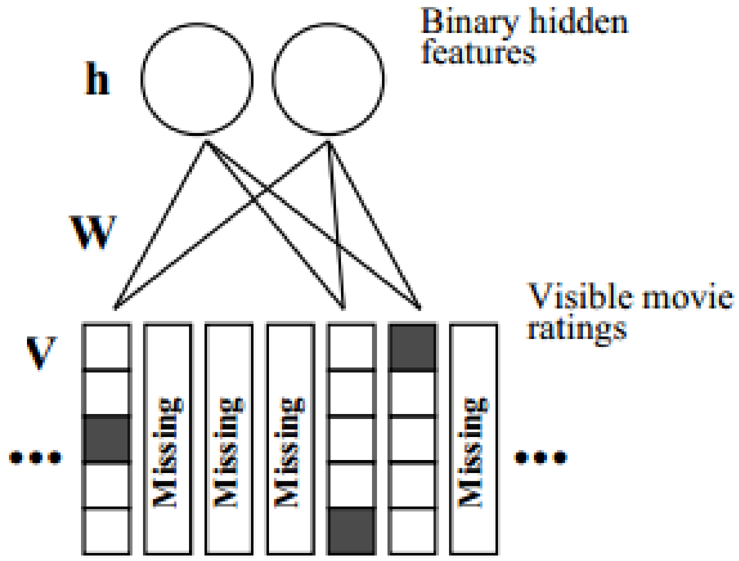


One pass



RBM for CF

- Each visible unit = an item
- Num of hidden units is a parameter
- In training phase, for each user:
  - If user rated item, v_i is activated
  - Activation states of v_i = inputs to h_j
  - Based on activation, h_j is computed
  - Activation state of h_j becomes input to v_i
  - Activation state of v_i is recalculated
  - Difference between current and past activation state for v_i used to update weights w_{ij} and thresholds



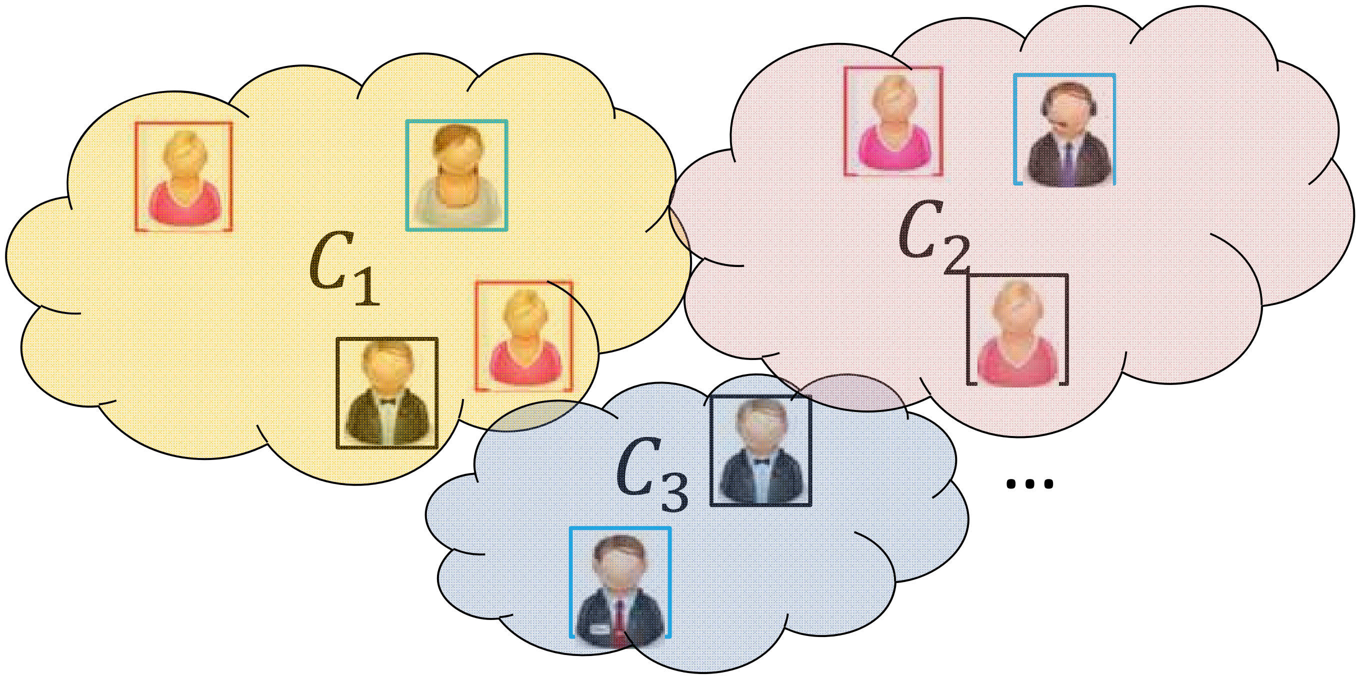
- In prediction phase:
  - For the items of the user the v_i are activated
  - Based on this the state of the h_j is computed
  - The activation of h_j is used as input to recompute the state of v_i
  - Activation probabilities are used to recommend items

Salakhutdinov R, Mnih A, Hinton G. Restricted Boltzmann machines for collaborative filtering. ICML, 2007

Clustering Based CF

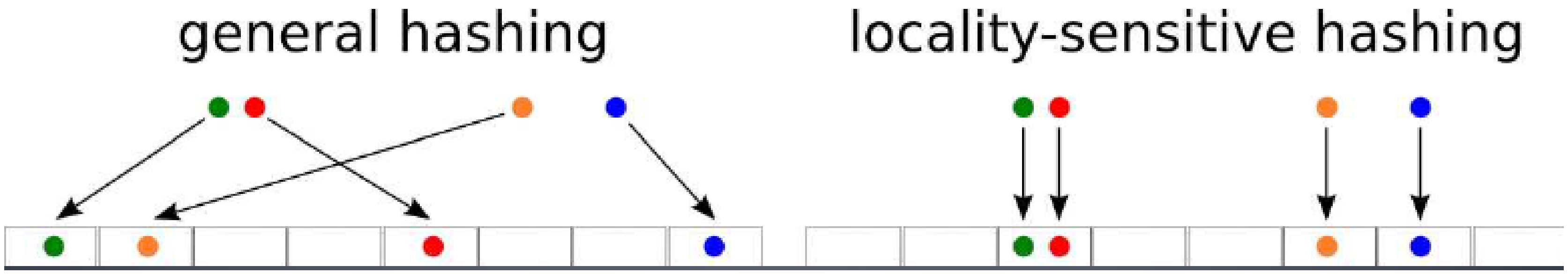
- Goal: cluster users and compute per-cluster "typical" preferences

- Users receive recommendations computed at the cluster level



LSH for clustering

- Method for grouping similar items in highly dimensional spaces
- Find a hashing function s.t. similar items are grouped in the same buckets
- Main application is Nearest-neighbors
  - Hashing function is found iteratively by concatenating random hashing functions
  - Addresses one of NN main concerns: performance



Classifiers for CF

- Classifiers are general computational models trained using positive and negative examples
- They may take in inputs:
  - Vector of item features (action / adventure)
  - Preferences of customers (like action / adventure)…
  - Relations among item
- E.g. Logistic Regression, Bayesian Networks, Support Vector Machines, Decision Trees, etc...
- Pros:
  - Versatile
  - Can be combined with other methods to improve accuracy
- Cons:
  - Need a relevant training set
  - May overfit

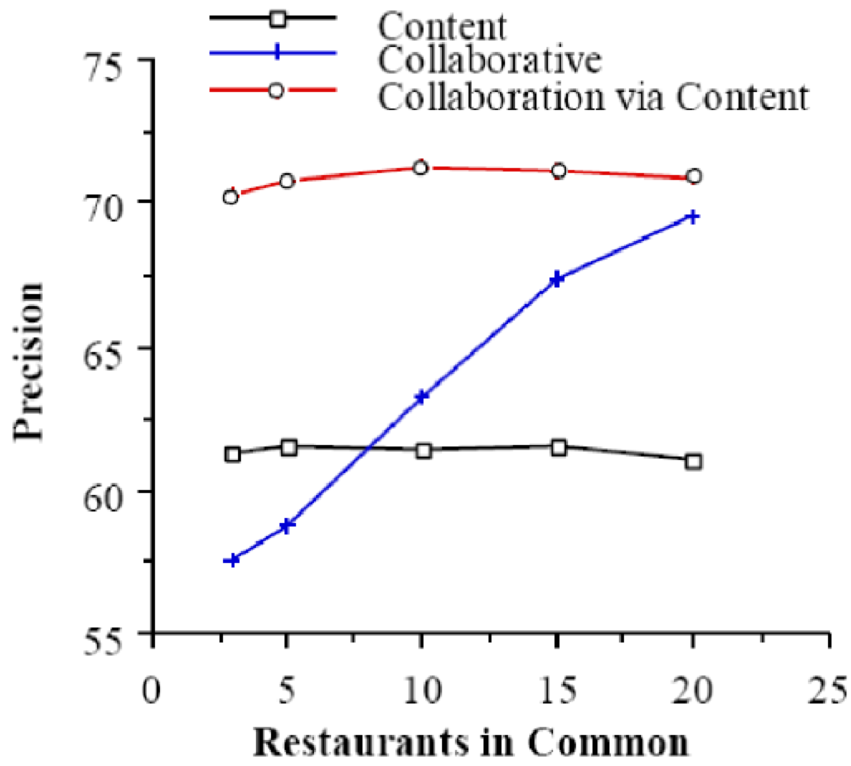
Limitations of CF

- Cold Start: There needs to be enough other users already in the system to find a match. New items need to get enough ratings.

- Popularity Bias: Hard to recommend items to someone with unique tastes. Tends to recommend popular items

Hybrid Approaches

- Content–based recommendation with Bayesian classifier
- Collaborative is standard using Pearson correlation:
- Collaboration via content uses the content-based user profiles



Hybridization Methods

Hybridization Method	Description
Weighted	Outputs from several techniques (in the form of scores or votes) are combined with different degrees of importance to offer final recommendations
Switching	Depending on situation, the system changes from one technique to another
Mixed	Recommendations from several techniques are presented at the same time
Feature combination	Features from different recommendation sources are combined as input to a single technique
Cascade	The output from one technique is used as input of another that refines the result
Feature augmentation	The output from one technique is used as input features to another
Meta-level	The model learned by one recommender is used as input to another

Index

Ranking

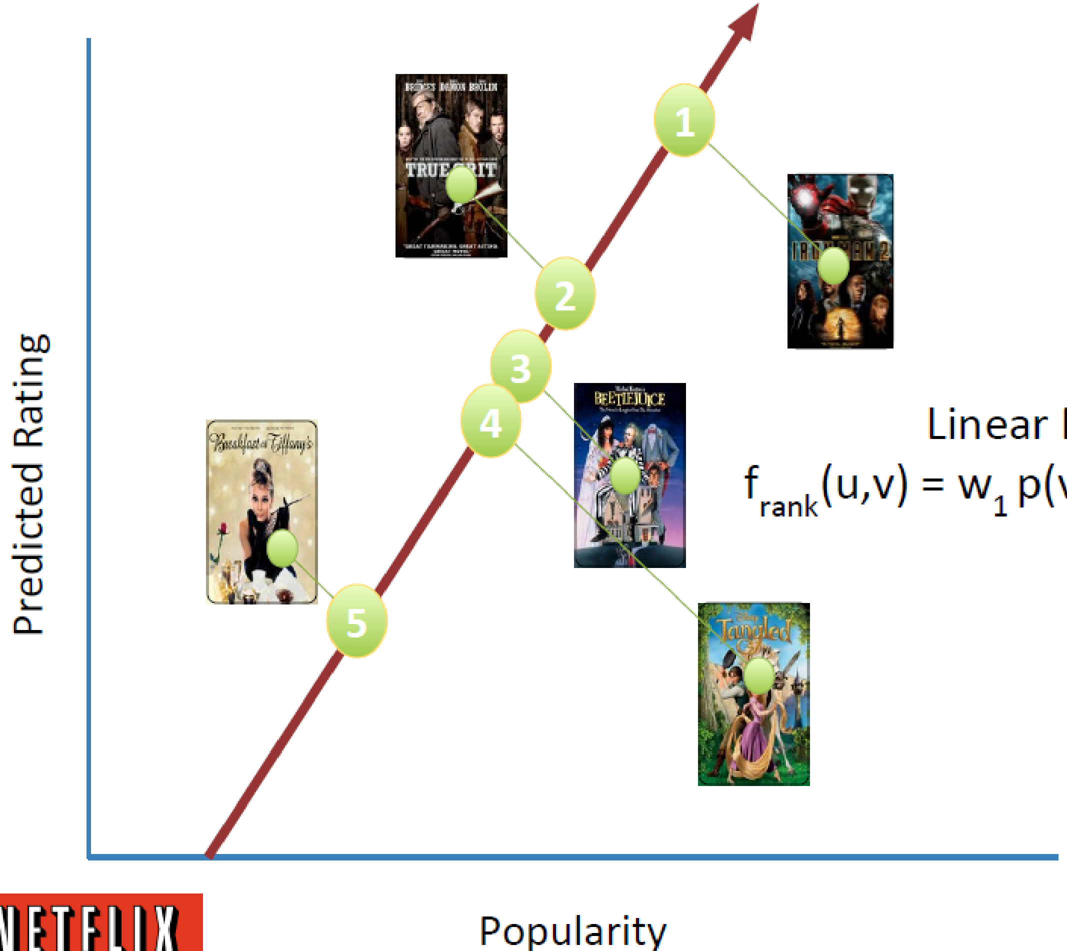
- Most recommendations are presented in a sorted list
- Recommendation can be understood as a ranking problem

4.7 4.6 4.5 4.5 4.5 4.5 4.5 4.5 4.5 4.5

Ranking

Ranking

Example: Two features, linear model



Linear Model:

$f_{rank}(u,v) = w_1\, p(v) + w_2\, r(u,v) + b$

Learning to rank - Metrics

- Quality of ranking measured using metrics as
  - Normalized Discounted Cumulative Gain
  - Mean Reciprocal Rank (MRR)
  - Fraction of Concordant Pairs (FCP)
  - Others: Precision, Recall, F-score
- Recent research on models that directly optimize ranking measures

Learning to rank - Approaches

- Pointwise:
  - Ranking function minimizes loss function defined on individual relevance judgment
  - Ranking score based on regression or classification
  - Ordinal regression, Logistic regression, SVM, GBDT, …
- Pairwise:
  - Loss function is defined on pair-wise preferences
  - Goal: minimize number of inversions in ranking
  - Ranking problem is then transformed into the binary classification problem
  - LambdaMart, RankSVM, RankBoost, RankNet, FRank…

Learning to rank - Approaches

- Listwise:
  - Indirect Loss Function
    a) RankCosine: similarity between ranking list and ground truth as loss function
    b) ListNet: KL-divergence as loss function by defining a probability distribution
  - Directly optimizing IR measures (difficult since they are not differentiable)
    a) Genetic Programming or Simulated Annealing
    b) Gradient descent on smoothed version of objective function (e.g. CLiMF c TFMAP)
    c) AdaRank uses boosting to optimize NDCG

Similarity as Recommendation

What is similarity?

- Similarity can refer to different dimensions
  - Similar in metadata/tags
  - Similar in user play behavior
  - Similar in user rating behavior
  - …
- You can learn a model for each of them and finally learn an ensemble

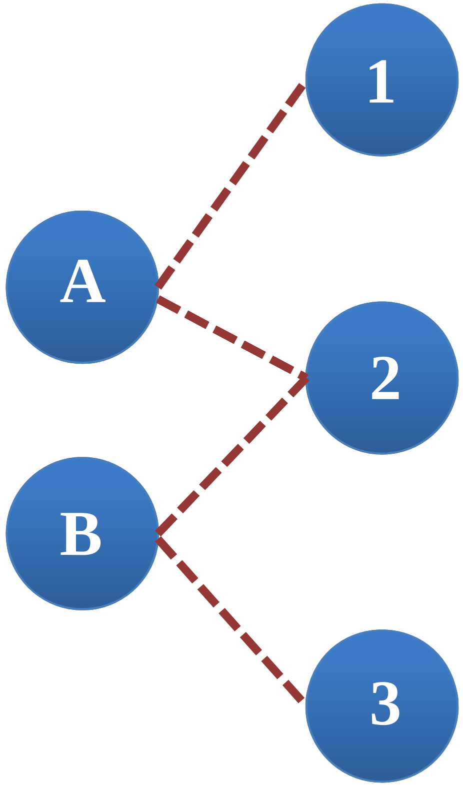
Graph-based similarities

Graph-based similarities

- SimRank “two objects are similar if they are referenced by similar objects.”

$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b))$$

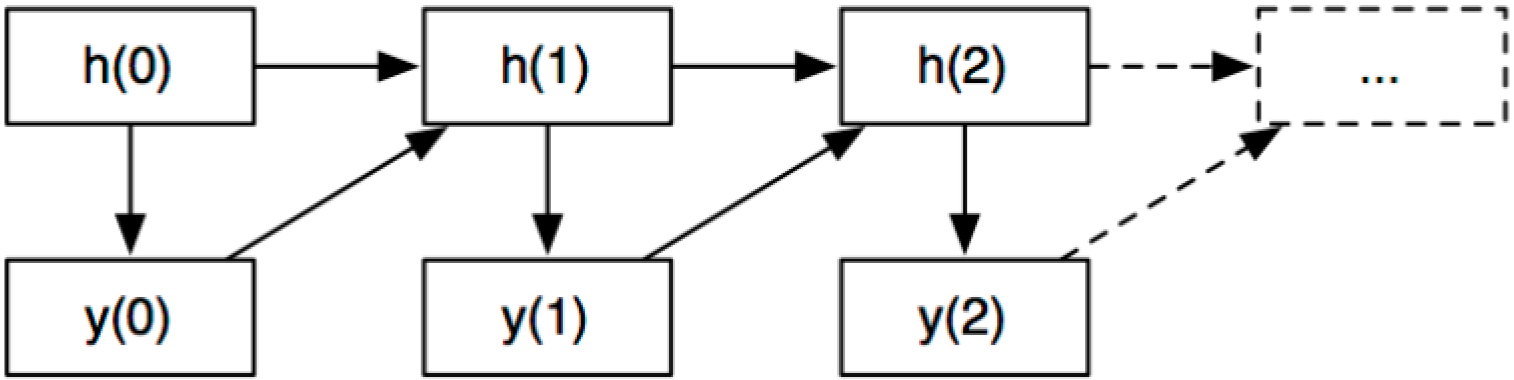
- We judge whether user A and B is similar via the relationship between their purchased item 1,2 and 3.



Jeh G, Widom J. SimRank: a measure of structural-context similarity[C] Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2002: 538-543.

Deep Learning for Recommendation

- RNNs have a simple model that tries to predict the next item given all previous ones. After predicting the item, the network gets to “know” what item it was, and incorporates this.

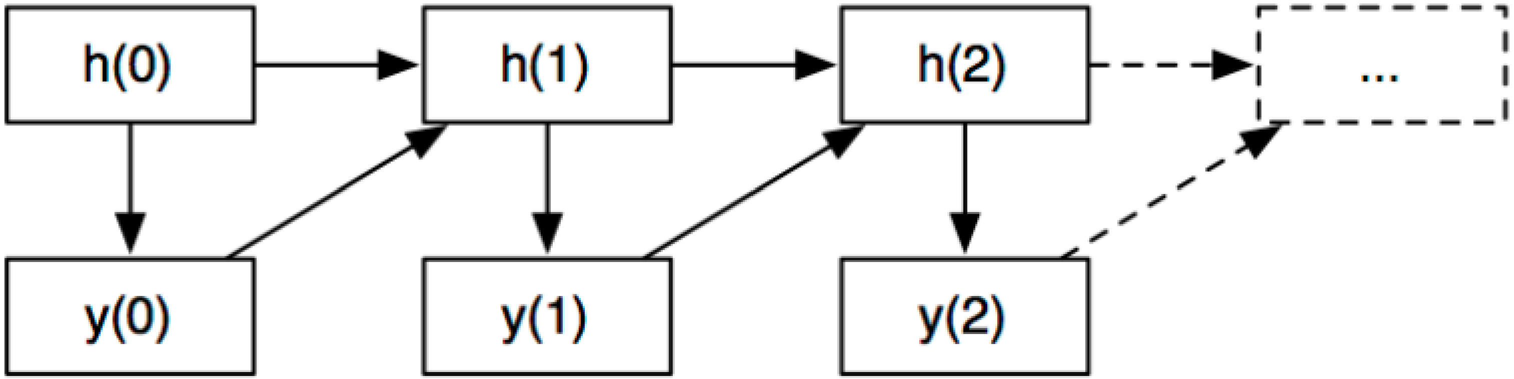


Deep Learning for Recommendation

- Predict the output given the hidden state. We need to model a Probability $P(\mathrm{y}_i \mid \mathrm{h}_i)$
- Observe the output y_t and feed it back into the next hidden state h_{i+1} In the most general form,

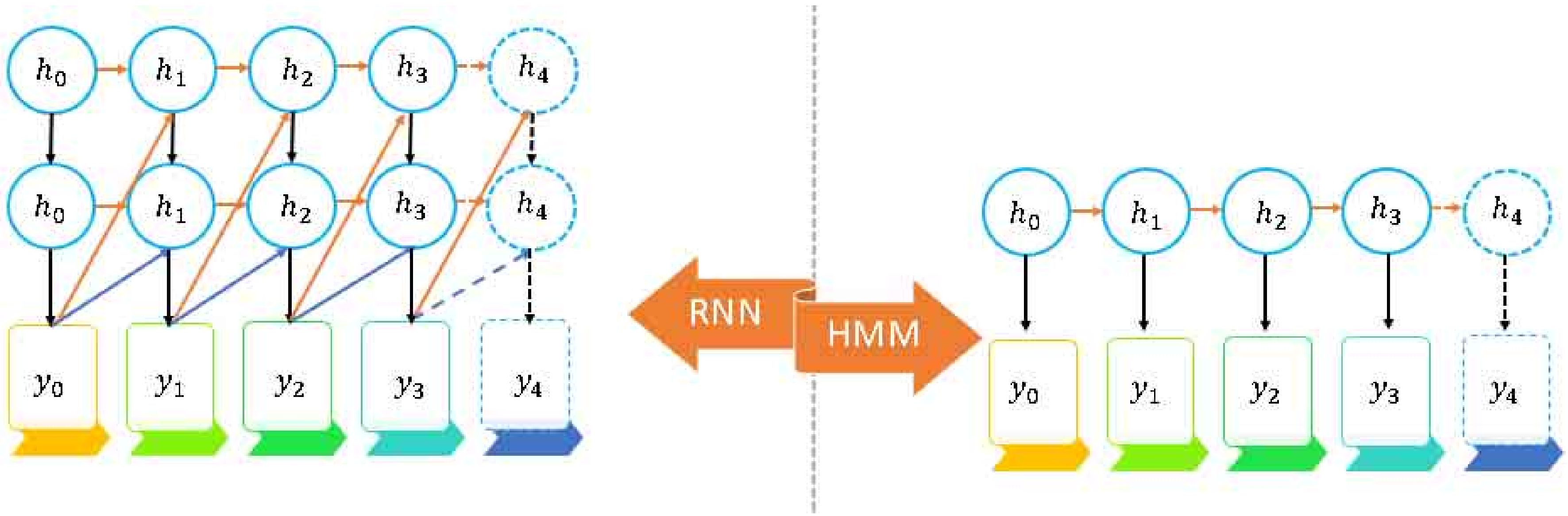
$$\mathrm{h}_{i+1} = f(\mathrm{a}(\mathrm{h}_i) + \mathrm{b}(\mathrm{y}_i))$$

In practice, f is generally some nonlinear function like sigmoid or tanh



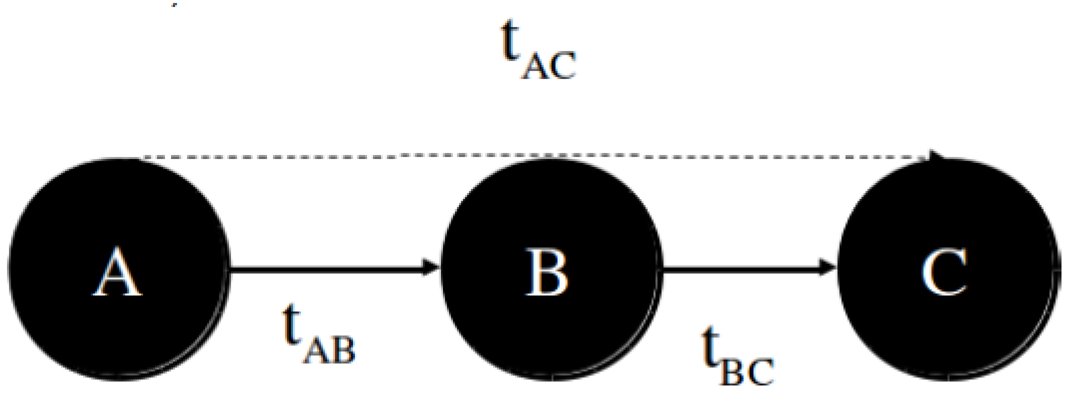
Deep Learning for Recommendation

- RNN do not make the Markov assumption and so can, in theory, take into account long-term dependencies
- The main advantages of using a RNN over HMM would be the greater representational power of neural networks and their ability to perform intelligent smoothing by taking into account syntactic and semantic features



Social Recommendations

- A social recommender system recommends items that are "popular" in the social proximity of the user
- Social proximity = trust (can also be topic-specific)
- Given two individuals - the source (node A) and sink (node C) - derive how much the source should trust the sink.
- Algorithm: Advogato, Appleseed, MoleTrust, TidalTrust

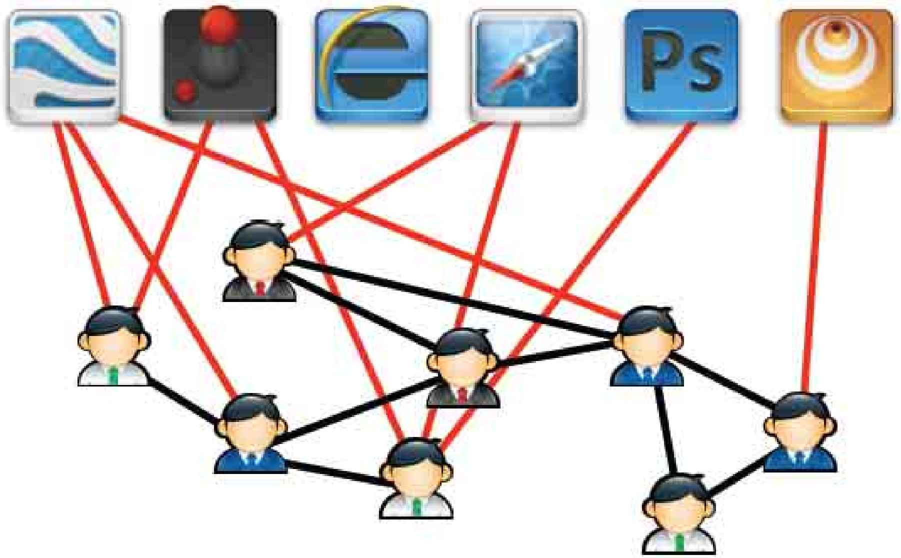


Social Recommendations

social network = friendship + interests

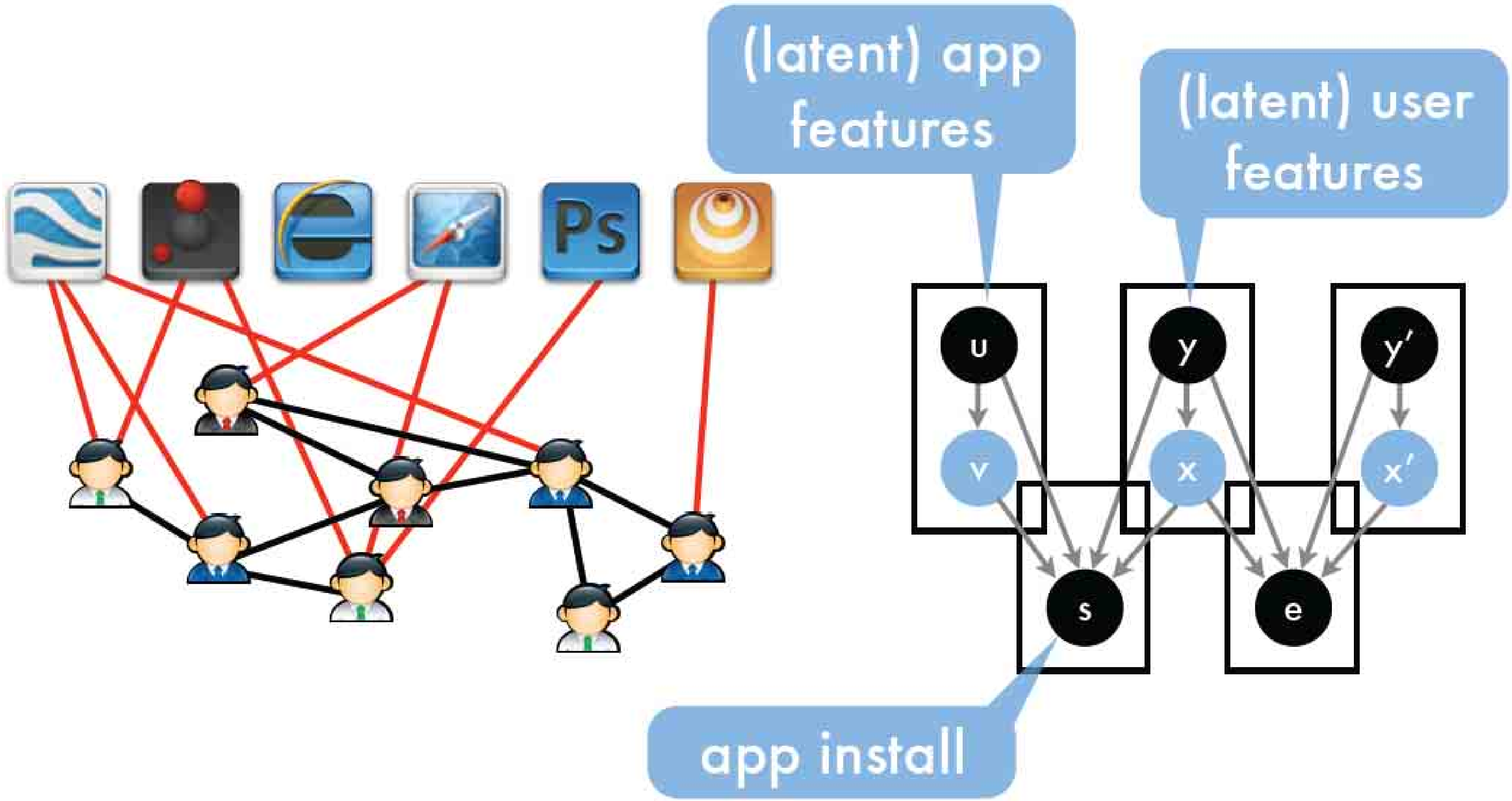
recommend users based on friendship & interests

recommend apps based on friendship & interests



users with similar interests are more likely to connect

Social Recommendations



Social Recommendations

$$\text{minimize} \quad \lambda_e \sum_{(i,j)} l(e_{ij}, x_i^\top x_j + y_i^\top W y_j) +$$

social

$$\lambda_a \sum_{(i,j)} l(a_{ij}, x_i^\top v_j + y_i^\top M u_j) +$$

app

reconstruction

$$\lambda_x \sum_i \gamma(x_i | y_i) + \lambda_v \sum_i \gamma(v_i | u_i) +$$

$$\lambda_W \|W\|^2 + \lambda_M \|M\|^2 + \lambda_A \|A\|^2 + \lambda_B \|B\|^2$$

regularizer

Social Recommendations

- Social connections can be used in combination with other approaches

- In particular, “friendships” can be fed into collaborative filtering methods in different ways

  - replace or modify user-user “similarity” by using social network information
  - use social connection as a part of the ML objective function as regularizer

Factorization Machines

- Generalization of regularized matrix (and tensor) factorization approaches combined with linear (or logistic) regression

- Problem: Each new adaptation of matrix or tensor factorization requires deriving new learning algorithms
  - Hard to adapt to new domains and add data sources
  - Hard to advance the learning algorithms across approaches
  - Hard to incorporate non-categorical variables

Rendle S. Factorization machines with libFM[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2012, 3(3): 57.

Factorization Machines

- Approach: Treat input as a real-valued feature vector
  - Model both linear and pair-wise interaction of k features (i.e. polynomial regression)
  - Traditional machine learning will overfit
  - Factor pairwise interactions between features
  - Reduced dimensionality of interactions promote generalization

- Combines "generality of machine learning/regression with quality of factorization models"

Factorization Machines

- Two categorical variables (u, i) encoded as real values:

Feature vector $\mathbf{x}$

	A	B	C	...	TI	NH	SW	ST	...
	User				Movie				
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...

- FM becomes identical to MF with biases:

$$f(\mathbf{x}) = b + w_u + w_i + \mathbf{v}_u^T \mathbf{v}_i$$

Factorization Machines

- Makes it easy to add a time signal

Feature vector **x**

	User				Movie					Time
	A	B	C	...	TI	NH	SW	ST	...	
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.2
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.6
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.61
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0.3
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0.5
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.1
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.8

- FM becomes as:

$$f(\mathbf{x}) = b + w_u + w_i + x_t w_t + \mathbf{v}_u^T \mathbf{v}_i + x_t \mathbf{v}_u^T \mathbf{v}_t + x_t \mathbf{v}_i^T \mathbf{v}_t$$

Index

- The Recommender Problem
- Traditional Methods
- Beyond Traditional Methods
- **RS Systems**
- Summary
- References

TOP 10 开源的推荐系统

● SVDFeature	http://svdfeature.apexlab.org/wiki/Main_Page
● libMF	http://www.csie.ntu.edu.tw/~cjlin/libmf/
● libFM	http://www.libfm.org/
● Lenskit	http://lenskit.grouplens.org/
● GraphLab	GraphLab - Collaborative Filtering
● Mahout	http://mahout.apache.org/
● Myrrix	http://myrrix.com/
● EasyRec	http://easyrec.org/
● Waffles	http://waffles.sourceforge.net/
● RapidMiner	http://rapidminer.com/

工业界的推荐系统

视频类：
Netflix：很多方法的融合
Hulu：主要是item based CF
Youtube：开始是random walk，后来改为类似item based CF的方法

图书类：
Amazon：好多方法都用了，主要是 item based CF

资讯类：
google news：用了CF和bayesian的方法。
digg：算法是 热门度+topic driven user based CF,

音乐类：
last.fm：用的是CF。
yahoo music：参考Koren的论文。
pandora：音乐基因项目，主要依赖专家标注。

社交类
facebook：算法叫Edgerank。
twitter：主要场景是推荐其它用户，参考官方介绍。

Widely used data

- Movie
  - MovieLens http://grouplens.org/datasets/movielens/
  - Netflix https://www.netflix.com/cn/
- Book
  - Amazon books http://www.amazon.com/b/ref=usbk_surl_books/?node=283155
  - Book-Crossing http://grouplens.org/datasets/book-crossing/
- Music
  - Last.fm http://www.last.fm/
- Food
  - Dianping http://www.dianping.com/
- Else…
  - Epinion http://www.datatang.com/data/11849

Index

- The Recommender Problem
- Traditional Methods
- Beyond Traditional Methods
- RS Systems
- **Summary**
- References

Summary

- For many applications such as Recommender Systems (but also Search, Advertising, and even Networks) understanding data and users is vital
- Algorithms can only be as good as the data they use as input
- Importance of User/Data Mining is going to be a growing trend in many areas in the coming years
- RS have the potential to become as important as Search is now
- RS are more than User Mining

Summary

- RS are fairly new but already grounded on well proven technology
  - Collaborative Filtering
  - Content Analysis
  - Machine Learning
  - Social Network Analysis
- However, there are still many open questions and a lot of interesting research to do!

Index

- The Recommender Problem
- Traditional Methods
- Beyond Traditional Methods
- Advanced Topics
- Summary
- **References**

References

1. [AT 2005] Adomavicius G., Tuzhilin, A. Toward the next generation of recommender systems: a survey of the state－of－the－art and possible extensions, IEEE TKDE, 17(6), 2005, pp.734－749.
2. [Lior 2010] "Recommender Systems Handbook." Ricci, Francesco, Lior Rokach, Bracha Shapira, and Paul B. Kantor. (2010).
3. [Jannach 2010] “Recommender systems: an introduction”. Jannach, Dietmar, et al. Cambridge University Press, 2010.
4. [G.A 2005] “Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions”. G. Adomavicious and A. Tuzhilin. 2005. IEEE Transactions on Knowledge and Data Engineering, 17 (6)
5. [B.S 2001] “Item-based Collaborative Filtering Recommendation Algorithms”, B. Sarwar et al. 2001. Proceedings of World Wide Web Conference.
6. [Koren 2007] “Lessons from the Netflix Prize Challenge.”. R. M. Bell and Y. Koren. SIGKDD Explor. Newsl., 9(2):75–79, December 2007.
7. [KS 2001] “Beyond algorithms: An HCI perspective on recommender systems”. K. Swearingen and R. Sinha. In ACM SIGIR 2001 Workshop on Recommender Systems
8. [J.B 1999] “Recommender Systems in E-Commerce”. J. Ben Schafer et al. ACM Conference on Electronic Commerce. 1999

References

9. [Hinton 2007] “Restricted Boltzmann machines for collaborative filtering”. R. Salakhutdinov, A. Mnih, and G. E. Hinton.In Proc of ICML ’07, 2007

10. [Z.C 2007] “Learning to rank: From pairwise approach to listwise approach”. Z. Cao and T. Liu. In In Proceedings of the 24th ICML, 2007.

11. [M.P 2007] “Content-based recommendation systems”. M. Pazzani and D. Billsus. In The Adaptive Web, volume 4321. 2007.

12. [J.L 2004] “Evaluating collaborative filtering recommender systems”. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. ACM Trans. Inf. Syst., 22(1): 5–53, 2004.

13.[A,K 2010] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. “Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering”. In Proc. of the fourth ACM Recsys, 2010.

14. [S.R 2011] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. “Fast context-aware recommendations with factorization machines”. In Proc. of the 34th ACM SIGIR, 2011.

15. [X.Z 2015] Xi Zhang, Jian Cheng, Shuang Qiu, Zhenfeng Zhu, Hanqing Lu. When Personalization meets Conformity: Collective Similarity based Multi-Domain Recommendation. SIGIR 2015.

16. [T.Y 2014] Ting Yuan, Jian Cheng .et.al. “Recommendation by Mining Multiple User Behaviors with Group Sparsity” AAAI Conference on Artificial Intelligence, 2014