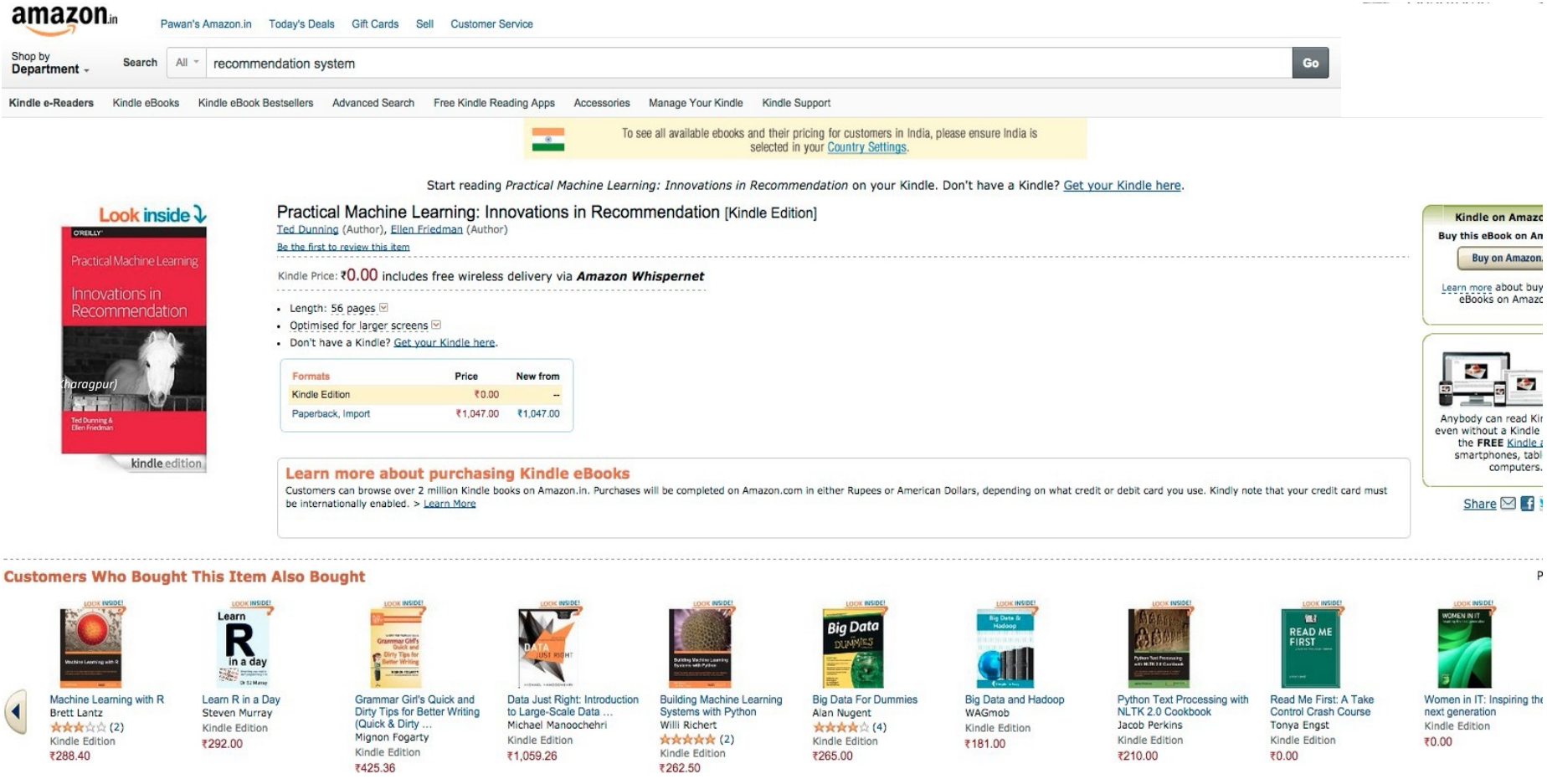
Recommendation Systems

Dr. Abhijnan Chakraborty
Department of Computer Science & Engg.,
Indian Institute of Technology Kharagpur

https://cse.iitkgp.ac.in/~abhijnan

Recommendation System?



Why use Recommender Systems?

Value for the customers

- Find things that are interesting
- Narrow down the set of choices
- Discover new things
- Entertainment ...

Value for the providers

- Additional and unique personalized service for the customer
- Increase trust and customer loyalty
- Increase sales, click through rates, conversion etc
- Opportunity for promotion, persuasion
- Obtain more knowledge about customers

Recommender System as a Function

What is given?

- User model: ratings, preferences, demographics, situational context
- Items: with or without description of item characteristics

Find

Relevance score: used for ranking

Final Goal

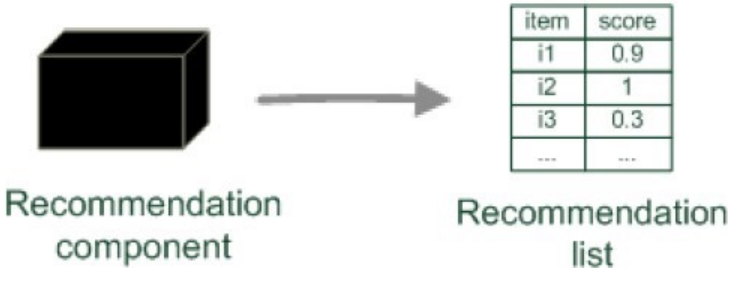
Recommend items that are assumed to be relevant

But

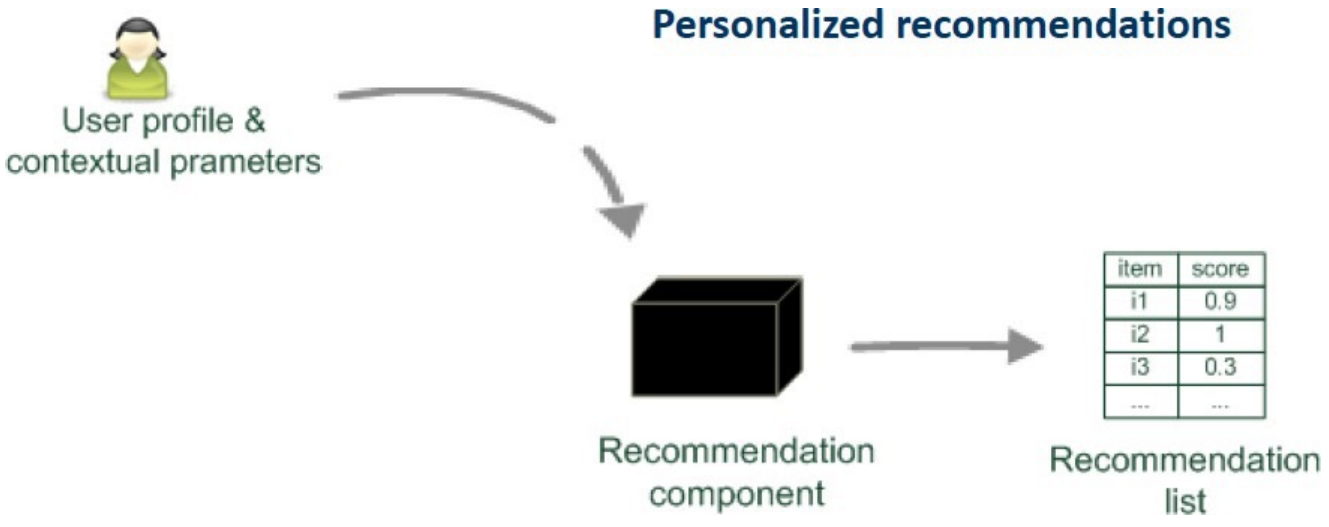
- Remember that relevance might be context-dependent
- Characteristics of the list might be important (diversity)

Paradigms of Recommender Systems

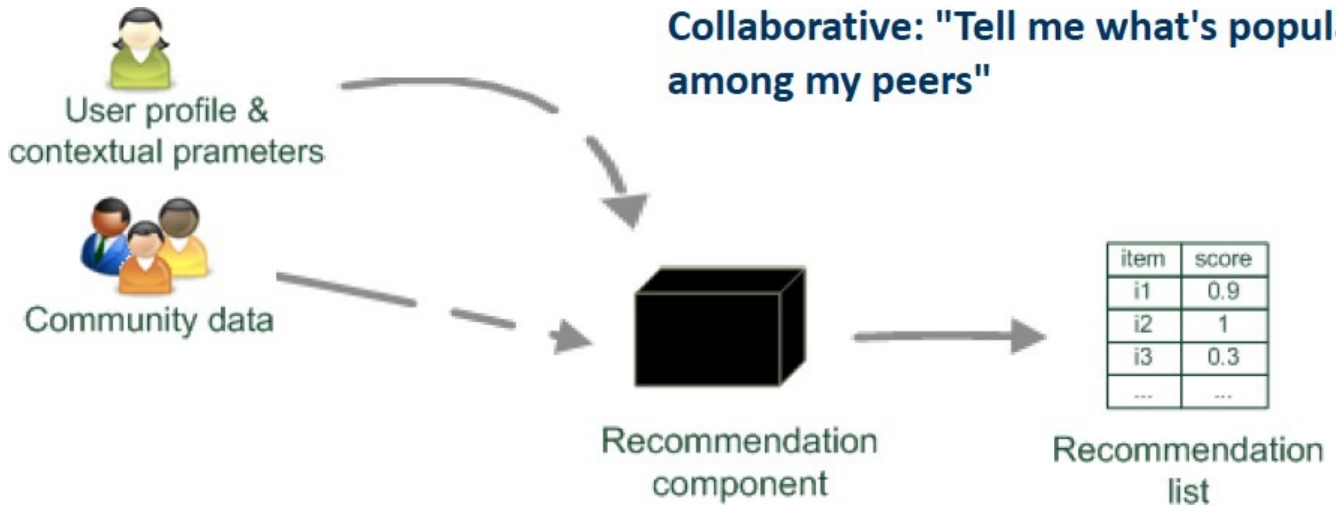
Recommender systems reduce information overload by estimating relevance



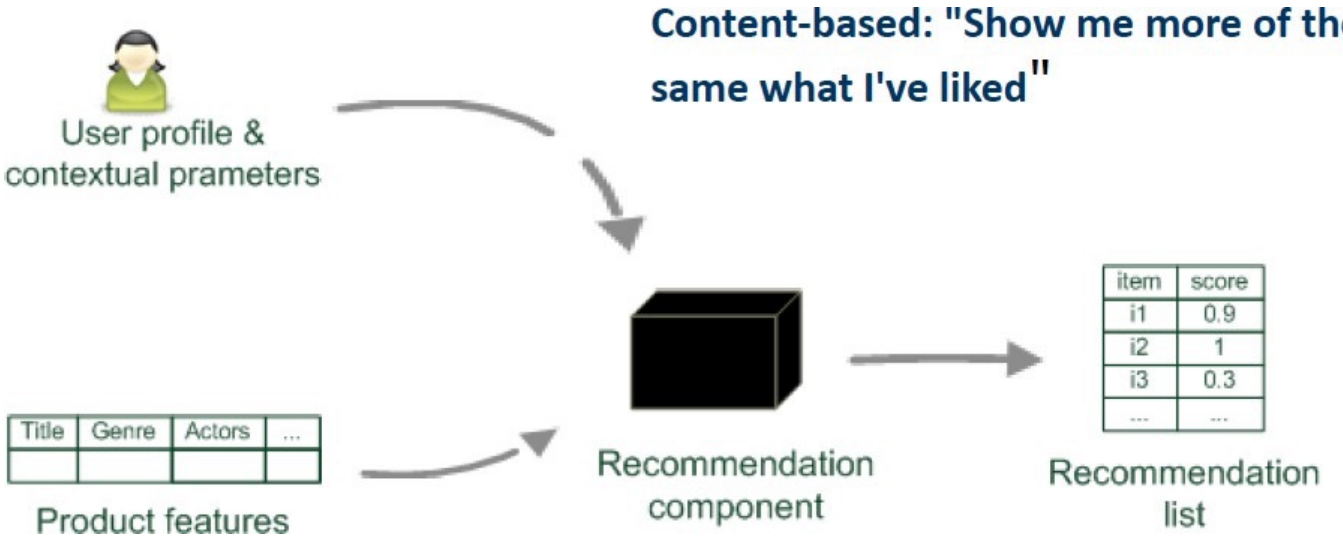
Paradigms of Recommender Systems



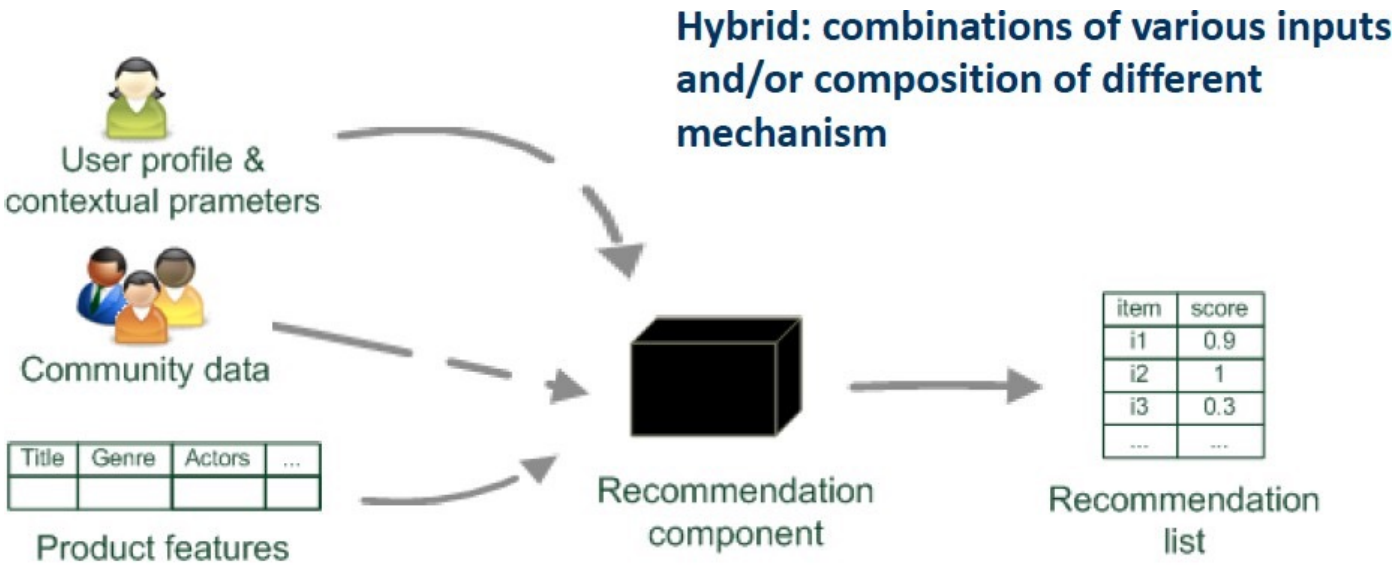
Paradigms of Recommender Systems



Paradigms of Recommender Systems



Paradigms of Recommender Systems



Sources for a Recommendation

- Content-based recommendation
  - Users and items are associated with features
  - Features are matched to infer interest

- Collaborative recommendations
  - Leverage user preferences in the form of ratings or other behavior
  - Recommend through similarity or latent factors

THE COLD START PROBLEM

New items have no ratings

and

New users have no history

Comparison across the Paradigms

	Pros 👍	Cons 👎
Collaborative	No knowledge-engineering effort, serendipity of results, learns market segments	Requires some form of rating feedback, cold start for new users and new items
Content-based	No community required, comparison between items possible	Content descriptions necessary, cold start for new users, no surprises

General idea of content-based recommendations

- Movies: recommend other movies with same director, actor, genre, as viewed ones
- Products: recommend other products in same category, brand, color, as purchased ones

Creating a Recommendation

- User is associated with some documents that describe his/her interests
  - Specified demographic profile
  - Specified interests at registration time
  - Descriptions of the items bought
- Items are also associated with semi-structured descriptions →

JBL GO lleva el sonido de calidad JBL a todas partes. GO es su solución de altavoz todo en uno y reproduce música en tiempo real vía Bluetooth desde smartphones y tabletas, gracias a su batería recargable. También cuenta con un práctico manos libres.

Potencia	**3 W**
Respuesta de Frecuencia	**180Hz – 20 kHz**
Tipo de altavoz	**Portátil**
Amplificador de sonido	**Integrado**

Possible Recommendation Methods

- **If no rating matrix is available**
  - k-nearest neighbor approach
    - Find the top-k items that are closest to the user (when items and users can be represented in the same space, e.g., dating apps)
  - Cosine similarity with tf-idf can be used
- **If a rating matrix is available**
  - Classification-based approach: training documents are those for which the user has specified binary preference
  - Regression-based approach in the case of ratings
- Limitations: depends on the quality of the features

Example: Regression-based Approach for Content-based Recommendation

Movie	Adventure	Action	Science-Fiction	Drama	Crime	Thiller		User 1	User 2
Star Wars IV	1	1	1	0	0	0		1	-1
Saving Private Ryan	0	0	0	1	0	0			
American Beauty	0	0	0	1	0	0			
City of Gold	0	0	0	1	1	0		-1	1
Interstellar	0	0	1	1	0	0		1	
The Matrix	1	1	1	0	0	1			1

...

We would do two regressions: one for the ratings of user 1 and another for user 2. (We can also do this for groups of users, e.g., by city and age)

How many rated movies would we need, as a minimum, to be able to do this?

Collaborative Filtering (CF)

The most prominent approach to generate recommendations

- Used by large, commercial e-commerce sites
- Well-understood, various algorithms and variations exist
- applicable in many domains (book, movies, ...)

Approach

Use the "wisdom of the crowd" to recommend items

Basic assumption and idea

- Users give ratings to catalog items (implicitly/explicitly)
- Customers with certain tastes in the past, might have similar tastes in the future

User-based Collaborative Filtering

- Given an active user *Alice* and an item *i* not yet seen by Alice
- The goal is to estimate Alice's rating for this item, e.g., by

User-based Collaborative Filtering

- Given an active user *Alice* and an item *i* not yet seen by Alice
- The goal is to estimate Alice's rating for this item, e.g., by
  - Find a set of users who liked the same items as Alice in the past and who have rated item *i*
  - Use the average of their ratings to predict if Alice will like item *i*
  - Do this for all items Alice has not seen and recommend the best-rated ones

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

User-based Collaborative Filtering

Key questions

- How do we measure similarity?
- How many neighbors should we consider?
- How do we generate a prediction from the neighbors' ratings?

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Popular Similarity Model

Pearson Correlation Coefficient

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \overline{r_a})(r_{b,p} - \overline{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \overline{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \overline{r_b})^2}}$$

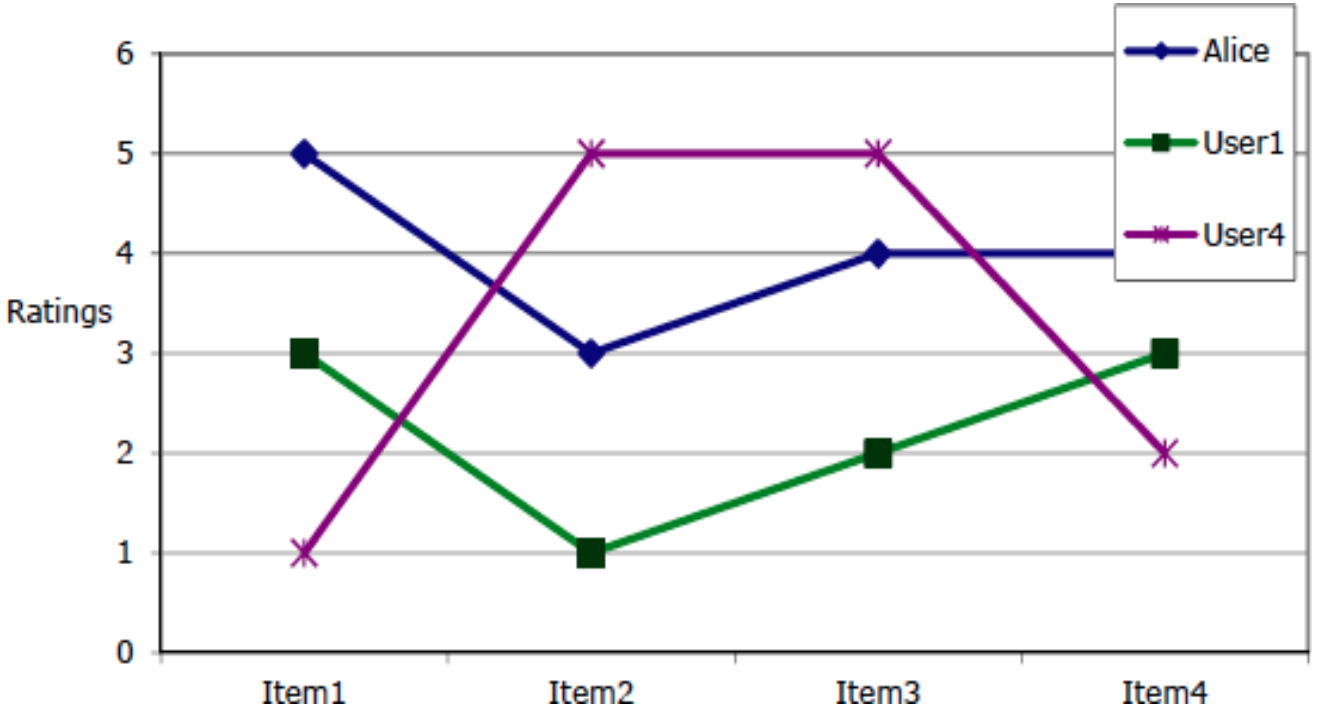
- a, b: users
- $r_{a,p}$: rating of user a for item p
- P: set of items, rated both by a and b
- $\overline{r_a}$, $\overline{r_b}$: user's average ratings
- Possible similarity values are between -1 to 1

For the example considered

- sim(Alice, User1) = 0.85
- sim(Alice, User4) = -0.79

Pearson Correlation

Takes difference in rating behavior into account



Making Predictions

- A common prediction function:

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

- Calculate, whether the neighbor's ratings for the unseen item *i* are higher or lower than their average
- Combine the rating differences - use similarity as a weight
- Add/subtract neighbor's bias from the active user's average and use this as a prediction

Item-based Collaborative Filtering

Basic Idea

Use the similarity between items to make predictions

For Instance

- Look for items that are similar to Item5
- Take Alice's ratings for these items to predict the rating for Item5

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Similarity Measure

- Ratings are seen as vector in *n*–dimensional space
- Similarity is calculated based on the angle between the vectors

$$sim(\vec{a},\vec{b}) = \frac{\vec{a}\cdot\vec{b}}{|\vec{a}| * |\vec{b}|}$$

- Adjusted cosine similarity: take average user ratings into account

$$sim(a,b) = \frac{\sum_{u\in U}(r_{u,a}-\overline{r_u})(r_{u,b}-\overline{r_u})}{\sqrt{\sum_{u\in U}(r_{u,a}-\overline{r_u})^2}\sqrt{\sum_{u\in U}(r_{u,b}-\overline{r_u})^2}}$$

Pre-processing for Item-based Filtering

- Calculate all pair-wise item similarities in advance
- The neighborhood to be used at run-time is typically rather small, because only those items are considered which the user has rated
- Item similarities are supposed to be more stable than user similarities

More on Ratings

Pure CF-based systems only rely on the rating matrix

Explicit ratings

- Most used (1 to 5, 1 to 10 response scales)
- **Research topics:** what about multi-dimensional ratings?
- **Challenge:** Sparse rating matrices, how to stimulate users to rate more items?

Implicit ratings

- Clicks, page views, time spent on some page, demo downloads ...
- Can be used in addition to explicit ones; question of correctness of interpretation

Data Sparsity Problems

Cold start problems

How to recommend new items? What to recommend to new users?

Straight-forward approach

Use another method (e.g., content-based, demographic or simply non-personalized) in the initial phase

Alternatives

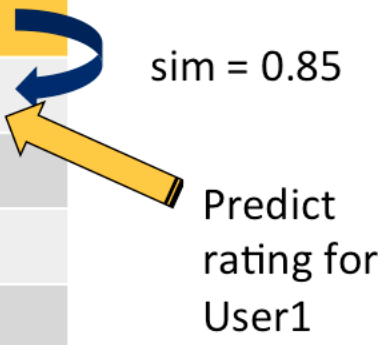
- Use better algorithms (beyond nearest-neighbor approaches)
- Example: Assume "transitivity" of neighborhoods

Example Algorithms for Sparse Datasets

Recursive Collaborative Filtering

- Assume there is a very close neighbor *n* of *u* who however has not rated the target item *i* yet.
- Apply CF-method recursively and predict a rating for item *i* for the neighbor *n*
- Use this predicted rating instead of the rating of a more distant direct neighbor

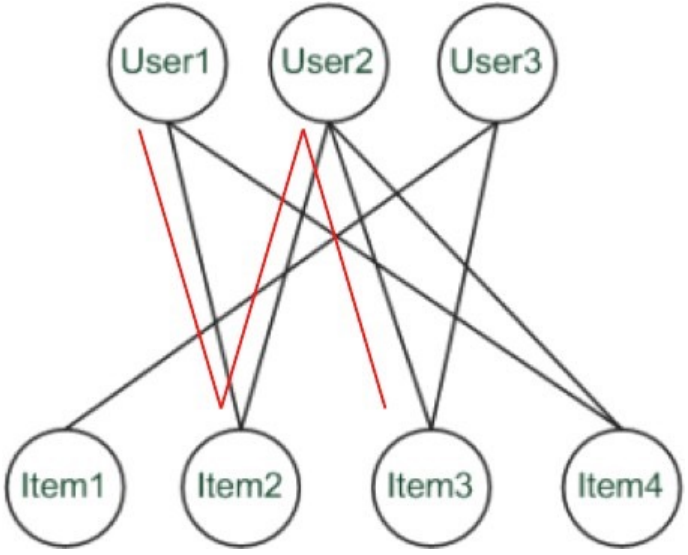
	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	?
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1



Example Algorithms for Sparse Datasets

Graph-based methods: Spreading activation

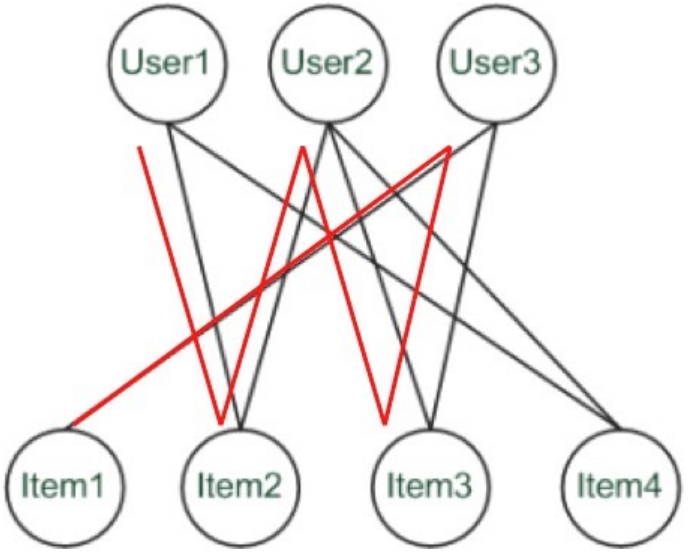
- Idea: Use paths of lengths 3 and 5 to recommend items
- Length 3: Recommend Item3 to User1



Example Algorithms for Sparse Datasets

Graph-based methods: Spreading activation

- Idea: Use paths of lengths 3 and 5 to recommend items
- Length 3: Recommend Item3 to User1
- Length 5: Item1 also recommendable



Matrix Factorization Methods

- Are shown to be superior to the classic nearest-neighbor techniques for product recommendations
- Allow the incorporation of additional information such as implicit feedback, temporal effects, and confidence levels

User-oriented Neighborhood Method

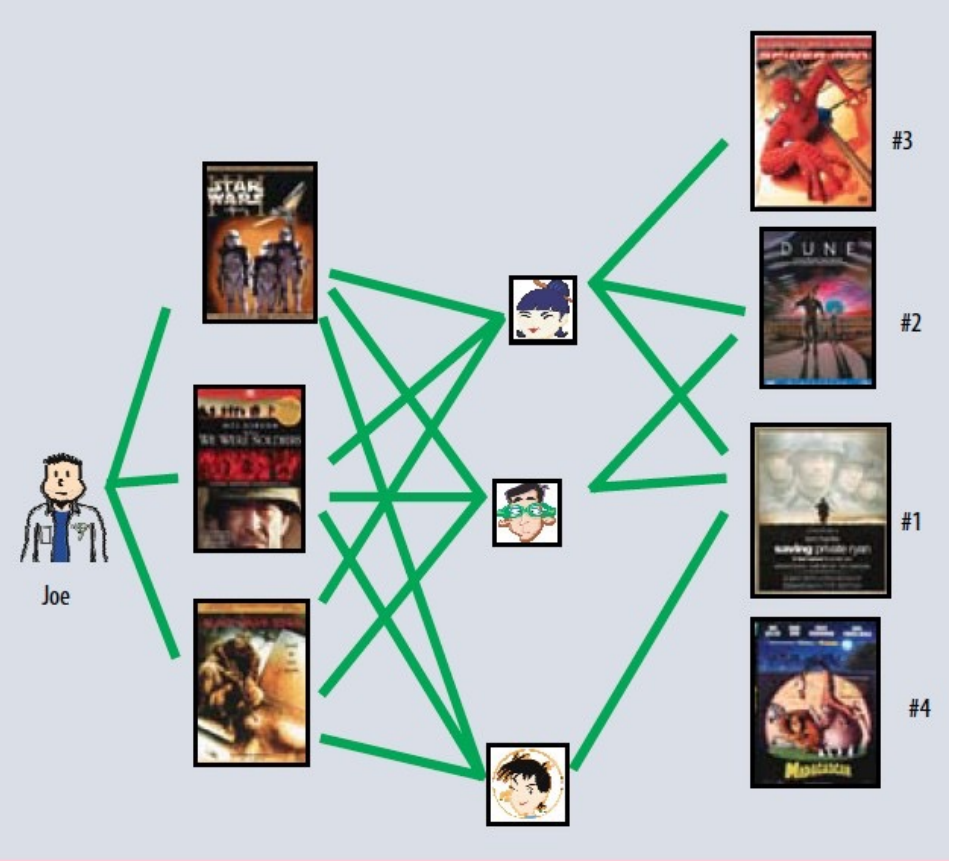


Figure 1. The user-oriented neighborhood method. Joe likes the three movies on the left. To make a prediction for him, the system finds similar users who also liked those movies, and then determines which other movies they liked. In this case, all three liked *Saving Private Ryan*, so that is the first recommendation. Two of them liked *Dune*, so that is next, and so on.

Latent Factor Approach

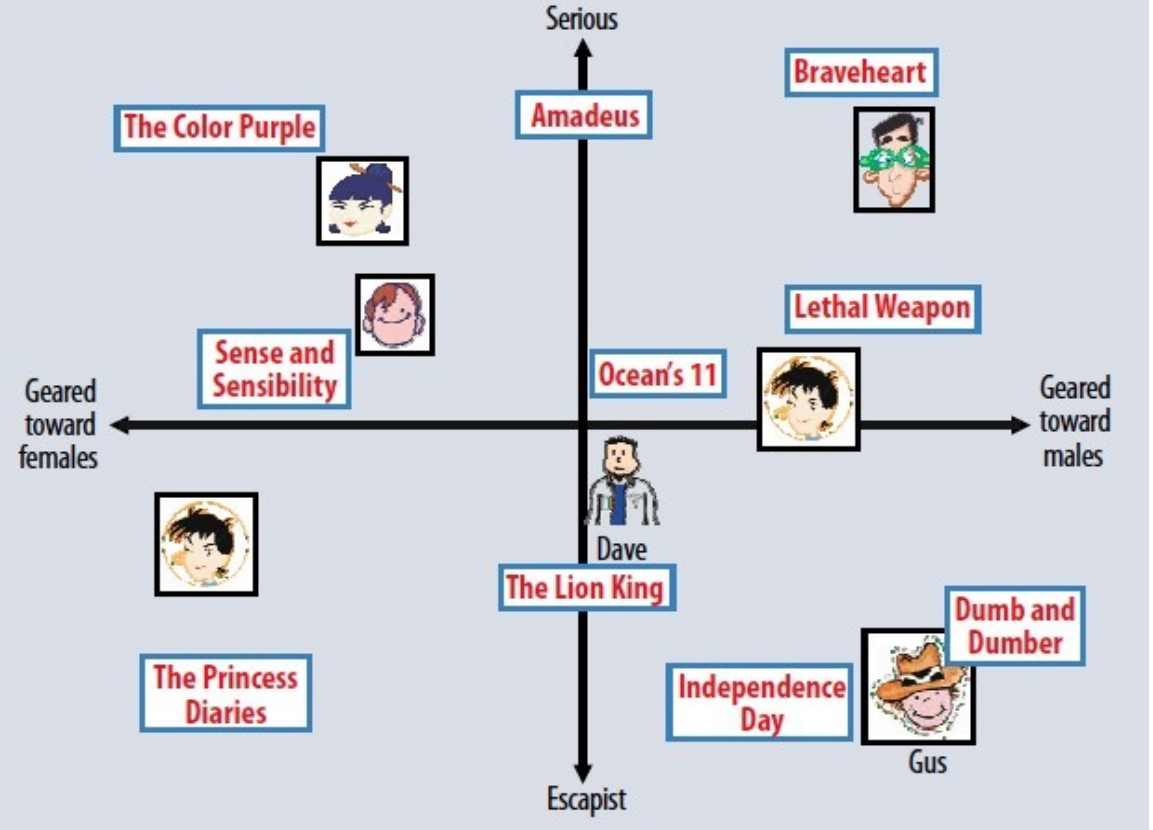


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

Matrix Factorization Methods

Basic Idea

- Both users and items are characterized by vectors of factors, inferred from item rating patterns
- High correspondence between item and user factors leads to a recommendation.

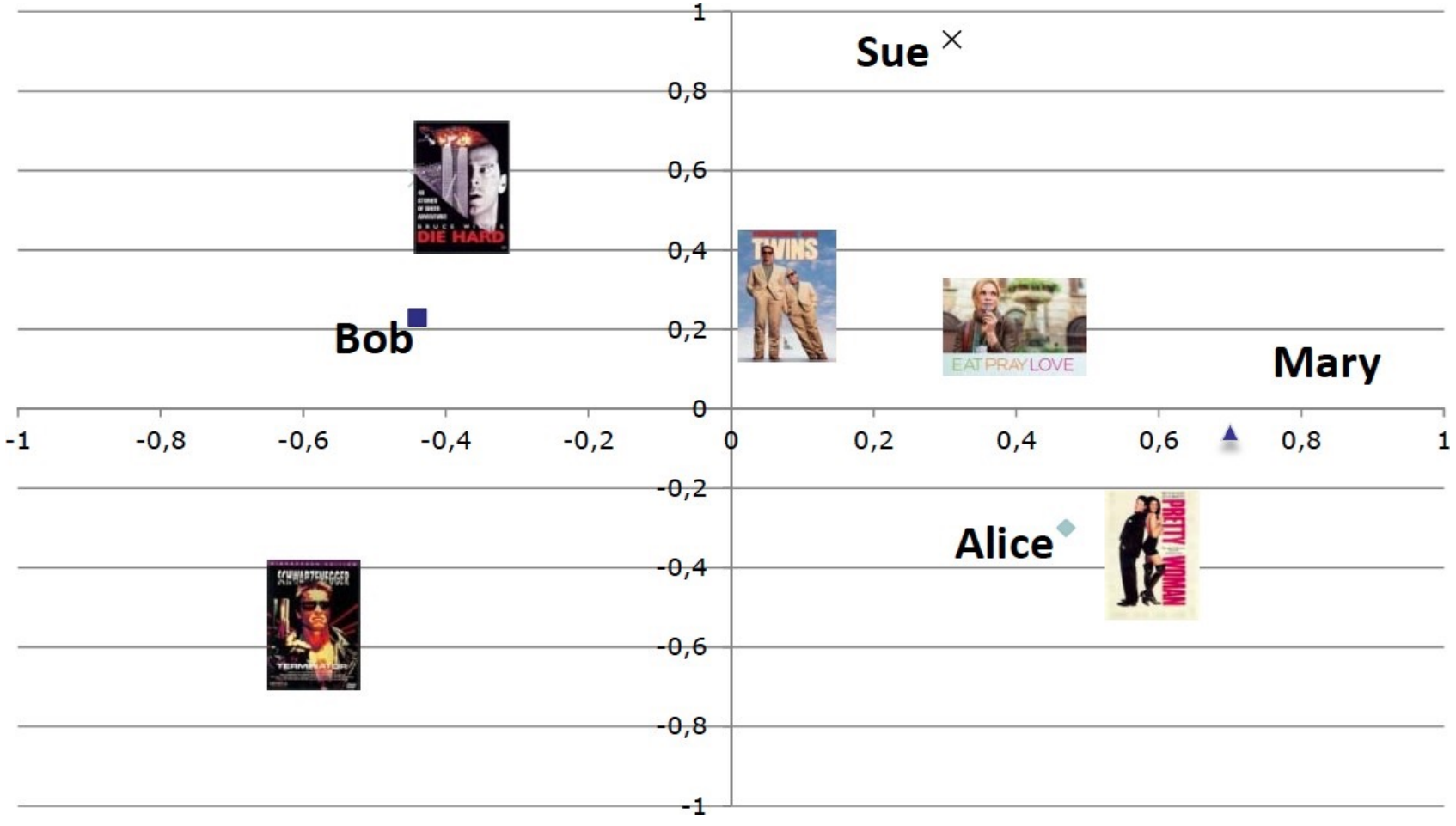
Using Singular Value Decomposition

- Let M be the matrix of user - item interactions
- Use SVD to get a k–rank approximation

$$M_k = U_k \times \Sigma_k \times V_k^T$$

- Prediction: $\hat{r_{ui}} = \overline{r_u} + U_k(u) \times \Sigma_k \times V_k{}^T(i)$

SVD: Example


Sue
Bob
Mary
Alice
DIE HARD
TWINS
EAT PRAY LOVE
PRETTY WOMAN
TERMINATOR
1
0,8
0,6
0,4
0,2
0
-0,2
-0,4
-0,6
-0,8
-1
-1
-0,8
-0,6
-0,4
-0,2
0
0,2
0,4
0,6
0,8
1

SVD: Example

- SVD: $M_k = U_k \times \Sigma_k \times V_k^T$

U_k	Dim1	Dim2
Alice	0.47	-0.30
Bob	-0.44	0.23
Mary	0.70	-0.06
Sue	0.31	0.93

V_k^T					
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

Σ_k	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

- Prediction: $\hat{r}_{ui} = \bar{r}_u + U_k(Alice) \times \Sigma_k \times V_k^T(EPL)$

  = 3 + 0.84 = 3.84

A Basic Matrix Factorization Model

- Both users and items are mapped to a joint latent factor space of dimensionality f
- User-item interactions are modeled as inner products in that space
- Each item i is associated with a vector $q_i \in R^f$, and each user u is associated with a vector $p_u \in R^f$
- q_i measures the extent to which the item possesses the factors, positive or negative
- p_u measures the extent of interest the user has in items that are high on the corresponding factors, positive or negative
- $q_i^T p_u$ captures the interaction between user u and item i
- This approximates user u's rating of item i, denoted by r_{ui}

$$\hat{r_{ui}} = q_i^T p_u$$

A Basic Matrix Factorization Model

Major Challenge

Computing the mapping of each item and user to factor vectors $q_i, p_u \in R^f$

The Learning Problem

To learn the factor vectors p_u and q_i, the system minimizes the regularized squared error on the set of known ratings:

$$min_{p^*,q^*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

where K is the set of (u,i) pairs for which r_{ui} is known.

This optimization problem is solved by using Stochastic Gradient Descent

Modifying the Basic Approach: Adding Biases

Matrix factorization is quite flexible in dealing with various data aspects and other application-specific requirements.

Adding Biases

- Some users might always give higher ratings than others, some items are widely perceived as better than others.
- Full rating value may not be explained solely by $q_i^T p_u$
- Identify the portion that individual user or item biases can explain

$$b_{ui} = \mu + b_i + b_u$$

- μ is the overall average rating, b_u and b_i indicate the observed deviations of user u and item i respectively, from the average

Modifying the Original Approach

Biases modify the interaction equation as

$$\hat{r_{ui}} = \mu + b_i + b_u + {q_i}^T p_u$$

Four components: global average, item bias, user bias, user-item interactions

The squared error function:

$$min_{p^*,q^*,b^*} \sum_{(u,i)\in K} (r_{ui} - \mu - b_i - b_u - {q_i}^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2 + {b_u}^2 + {b_i}^2)$$

Two Widely-Used Loss

Pointwise loss e.g., log loss

- Cast the recommendation task as a classification problem
- Rating Prediction, CTR Prediction …

Pairwise loss e.g., Bayesian Personalized Ranking (BPR) loss

- Cast the recommendation task as a ranking problem
- Top-N Recommendation, Preference Ranking …

$$Loss = \sum_{(u,i,j) \in O} - \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \|\Theta\|_2^2$$

Relative order between observed & unobserved interactions

Example of Collaborative Filtering in Python

Example: Grocery Shopping

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0	1	0	1	2	2
Fruits	2	3	1	1	2	2
Sweets	1	1	1	0	1	1
Bread	0	2	3	4	1	1
Coffee	0	0	0	0	1	0

- This purchase history indicates the number of time each person has purchased an item
- For clarity we're dealing with categories of items, but they can be the items themselves

In Python

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0	1	0	1	2	2
Fruits	2	3	1	1	2	2
Sweets	1	1	1	0	1	1
Bread	0	2	3	4	1	1
Coffee	0	0	0	0	1	0

Python code

```
V = np.array(
            [[0,1,0,1,2,2],
            [2,3,1,1,2,2],
            [1,1,1,0,1,1],
            [0,2,3,4,1,1],
            [0,0,0,0,1,0]])

V = pd.DataFrame(V, columns=['John',
'Alice', 'Mary', 'Greg', 'Peter',
'Jennifer'])

V.index = ['Vegetables', 'Fruits',
'Sweets', 'Bread', 'Coffee']
```

Matrix Factorization ($V \simeq WH$)

Matrix W (items x factors) with possible names for each factor added for legibility

	Fruits pickers	Bread eaters	Veggies
Vegetables	0.00	0.04	2.74
Fruits	1.93	0.15	0.47
Sweets	0.97	0.00	0.00
Bread	0.00	2.66	1.18
Coffee	0.00	0.00	0.59

Python code

```
from sklearn.decomposition
import NMF
nmf = NMF(3)
nmf.fit(V)
H =
pd.DataFrame(np.round(nmf.comp
onents_,2), columns=V.columns)
H.index = ['Fruits pickers',
'Bread eaters',  'Veggies']
W =
pd.DataFrame(np.round(nmf.tran
sform(V),2), columns=H.index)
W.index = V.index
```

Matrix W (items x factors)

	Fruits pickers	Bread eaters	Veggies
Vegetables	0.00	0.04	2.74
Fruits	1.93	0.15	0.47
Sweets	0.97	0.00	0.00
Bread	0.00	2.66	1.18
Coffee	0.00	0.00	0.59

Possible names for each factor added for legibility: these names are **not needed for the method to work**

Matrix H (factors x people)

	John	Alice	Mary	Greg	Peter	Jennifer
Fruits pickers	1.04	1.34	0.55	0.26	0.89	0.90
Bread eaters	0.00	0.60	1.12	1.36	0.03	0.07
Veggies	0.00	0.35	0.00	0.34	0.77	0.69

Reconstruction

Original matrix (V)

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0	1	0	1	2	2
Fruits	2	3	1	1	2	2
Sweets	1	1	1	0	1	1
Bread	0	2	3	4	1	1
Coffee	0	0	0	0	1	0

Reconstructed matrix (W H)

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0.00	0.98	0.04	0.99	2.11	1.89
Fruits	2.01	2.84	1.23	0.87	2.08	2.07
Sweets	1.01	1.30	0.53	0.25	0.86	0.87
Bread	0.00	2.01	2.98	4.02	0.99	1.00
Coffee	0.00	0.21	0.00	0.20	0.45	0.41

```
reconstructed = pd.DataFrame(np.round(np.dot(W,H),2), columns=V.columns)
reconstructed.index = V.index
```

Recommendation

Original matrix (V)

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0	1	0	1	2	2
Fruits	2	3	1	1	2	2
Sweets	1	1	1	0	1	1
Bread	0	2	3	4	1	1
Coffee	0	0	0	0	1	0

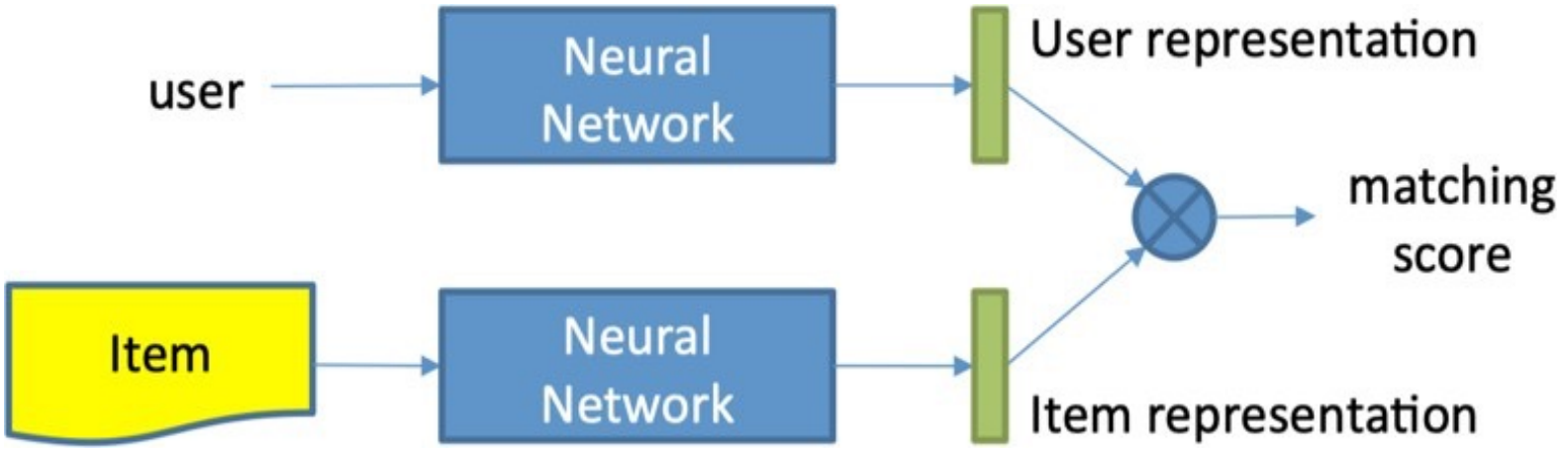
Reconstructed matrix (W H)

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0.00	0.98	0.04	0.99	2.11	1.89
Fruits	2.01	2.84	1.23	0.87	2.08	2.07
Sweets	1.01	1.30	0.53	0.25	0.86	0.87
Bread	0.00	2.01	2.98	4.02	0.99	1.00
Coffee	0.00	0.21	0.00	0.20	0.45	0.41

If you were to recommend one product to someone, what would you recommend and to whom?

Neural Network Meets CF

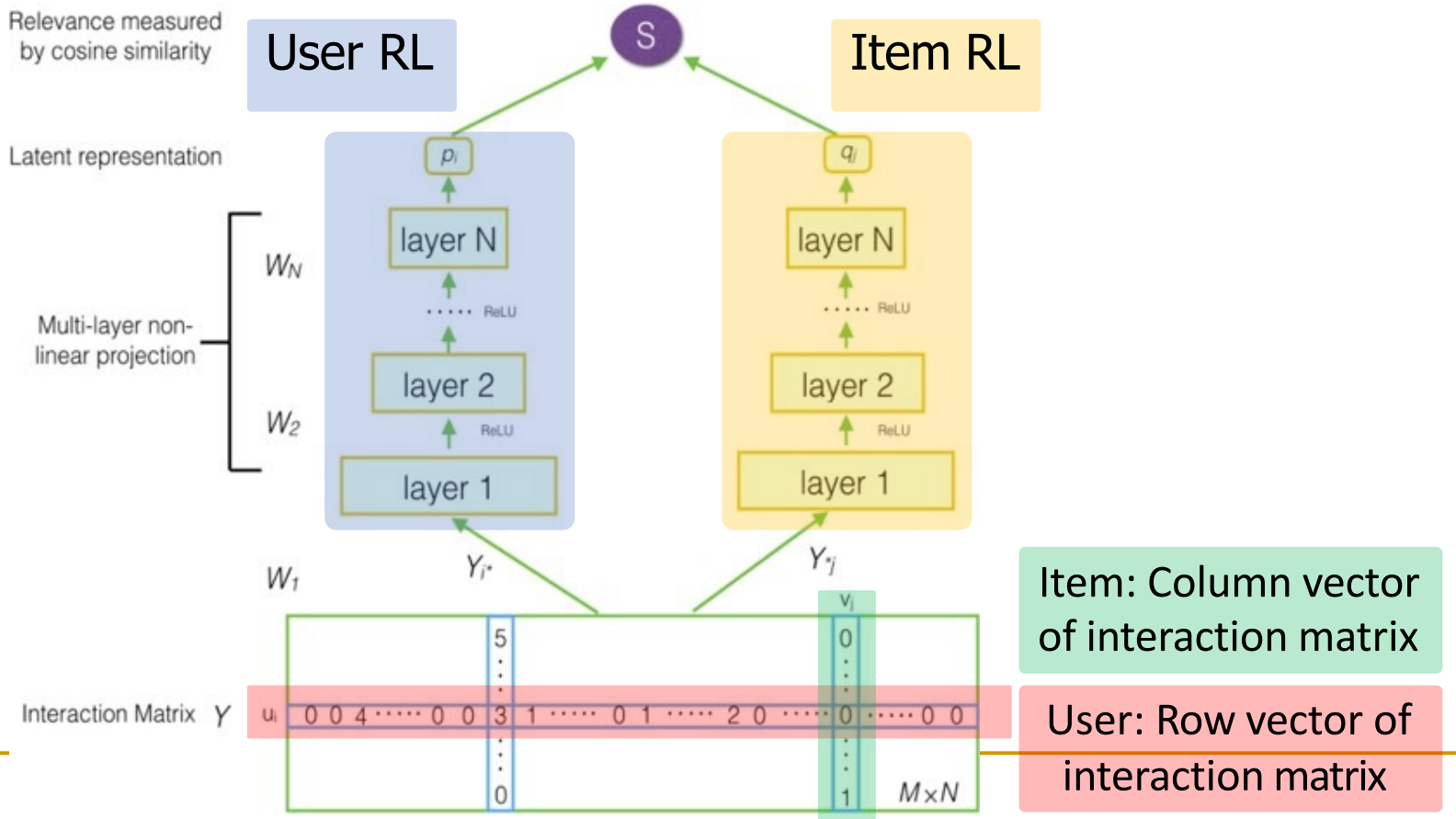
- Methods of representation learning
  - Enhance representation ability/expressiveness of models



Deep Matrix Factorization (IJCAI'17)

Representation Learning → Multi-layer perceptron

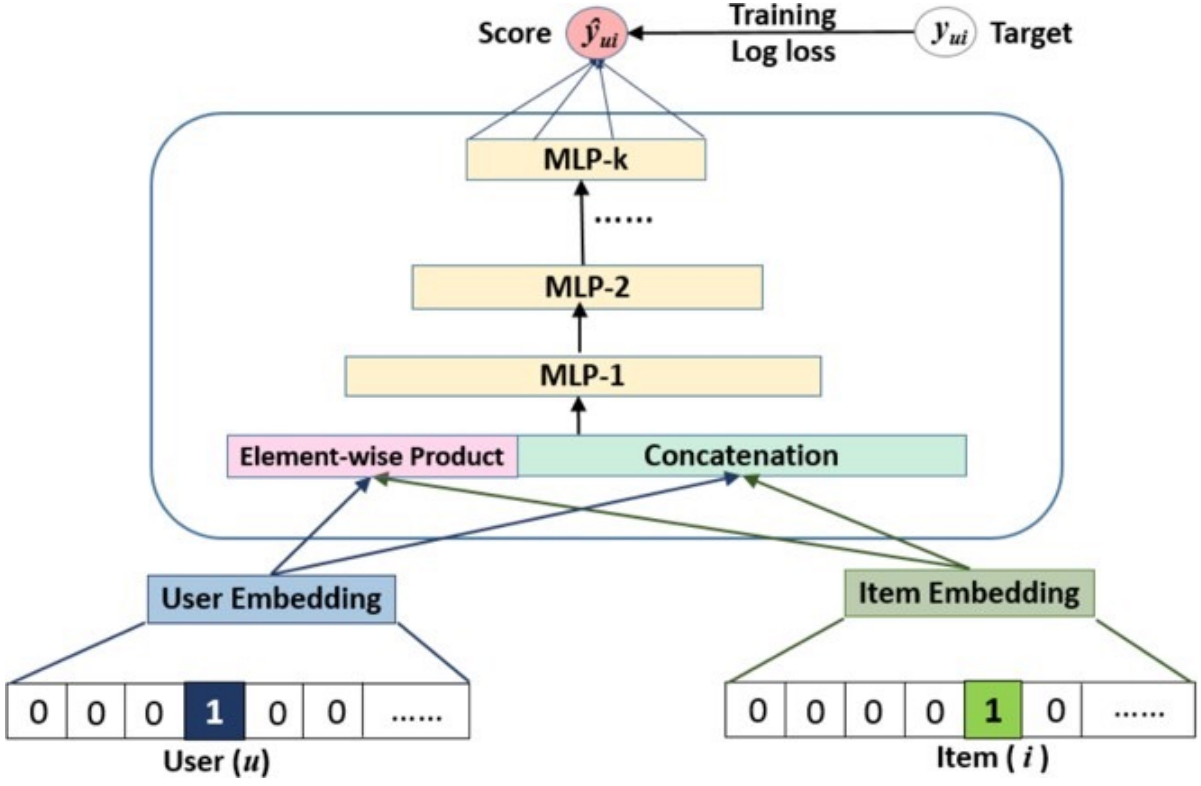
- Deep Neural Networks are adopted to learn representations of users & items



Neural Matrix Factorization (WWW'17)

Interaction Modeling → MF + MLP over users and items

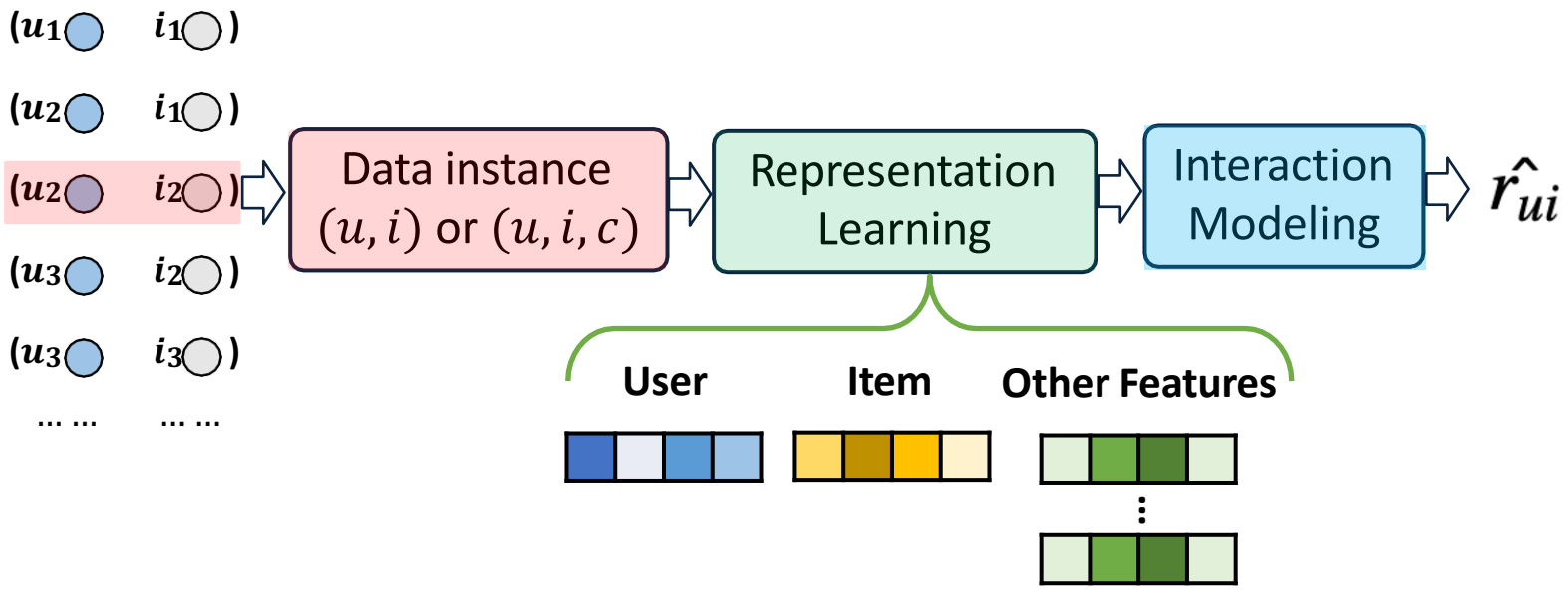
- MF uses inner product to capture the low-rank relation
- MLP is more flexible in learning the matching function



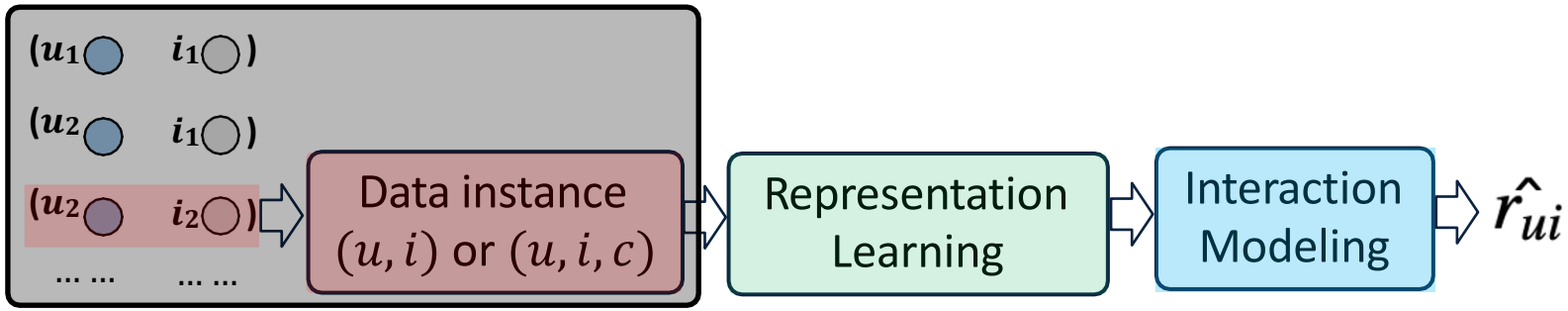
A General Paradigm

Transform each observation, a user-item pair (u, i) or with additional information (u, i, c) into a separate data instance

- Initiate representations for each feature → Representation Learning
  - Add whatever features you want
- Perform predictions based on interactions → Interaction Modeling
  - Design whatever networks you like

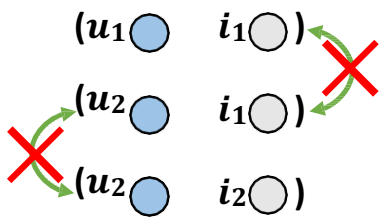


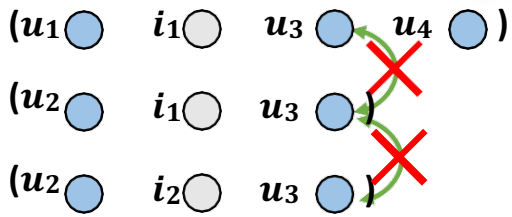
Information Isolated Island



Treating each observation as an independent instance

- Forgoing relationships among instances





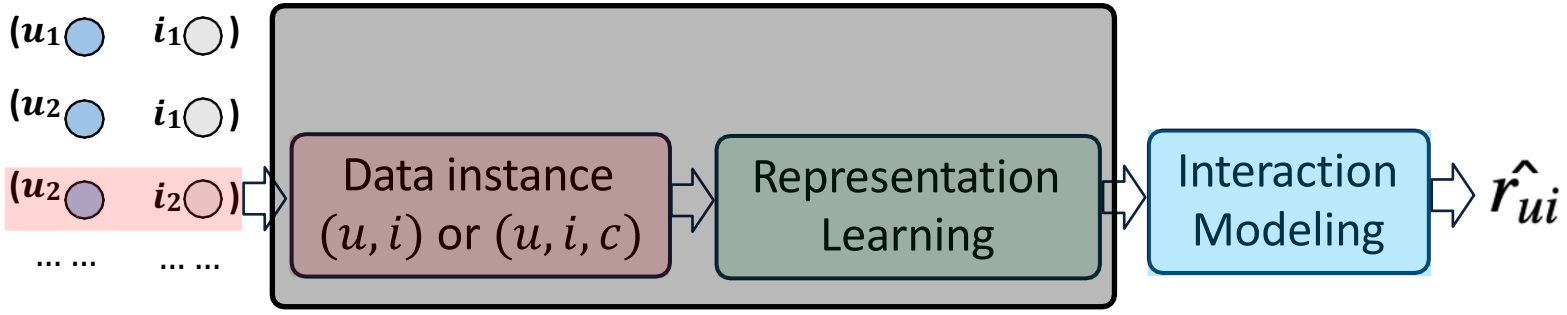
User-Item Interactions

- Behavior similarity among users
- Audience similarity among items

User-Item Interactions + Social Ties

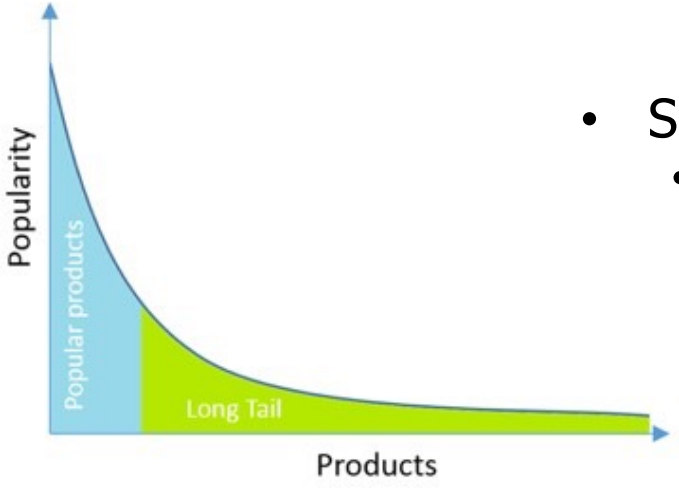
- Shared friends as bridge among users
  → mouth marketing

Information Isolated Island



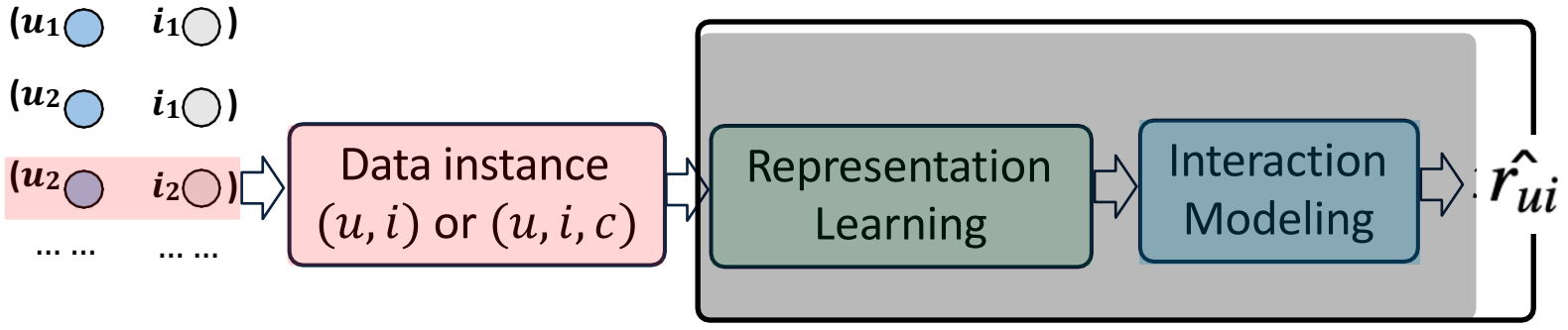
Treating each observation as an independent instance

- Limited Representation Ability
  - Instance representation is dependent only on its own features



- Suffer from sparsity issue
  - inactive users, unpopular items, infrequent features → insufficient information to learn optimal representation

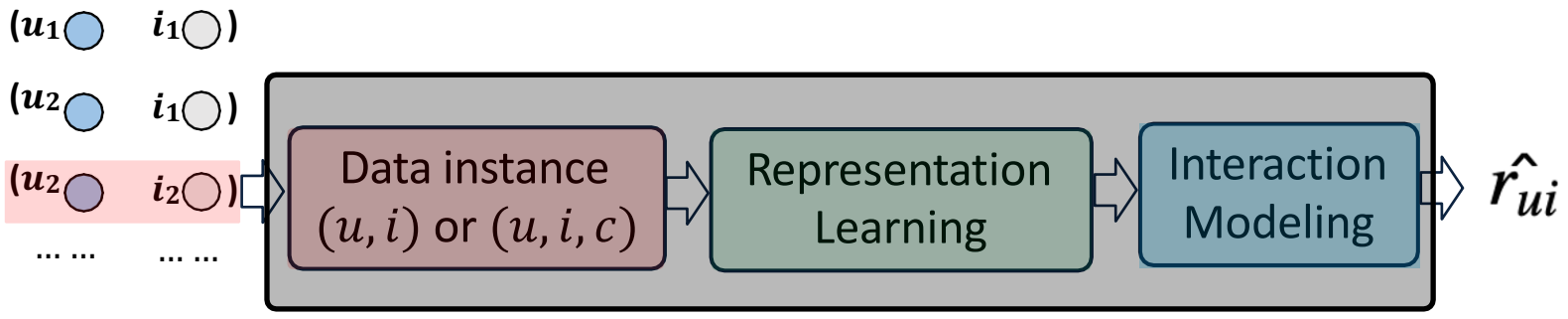
Information Isolated Island



Treating each observation as an independent instance

- Suboptimal Model Capacity
  - Suboptimal representations lead to unsatisfactory interaction model, especially for unseen (user-item or feature) interactions

Information Isolated Island



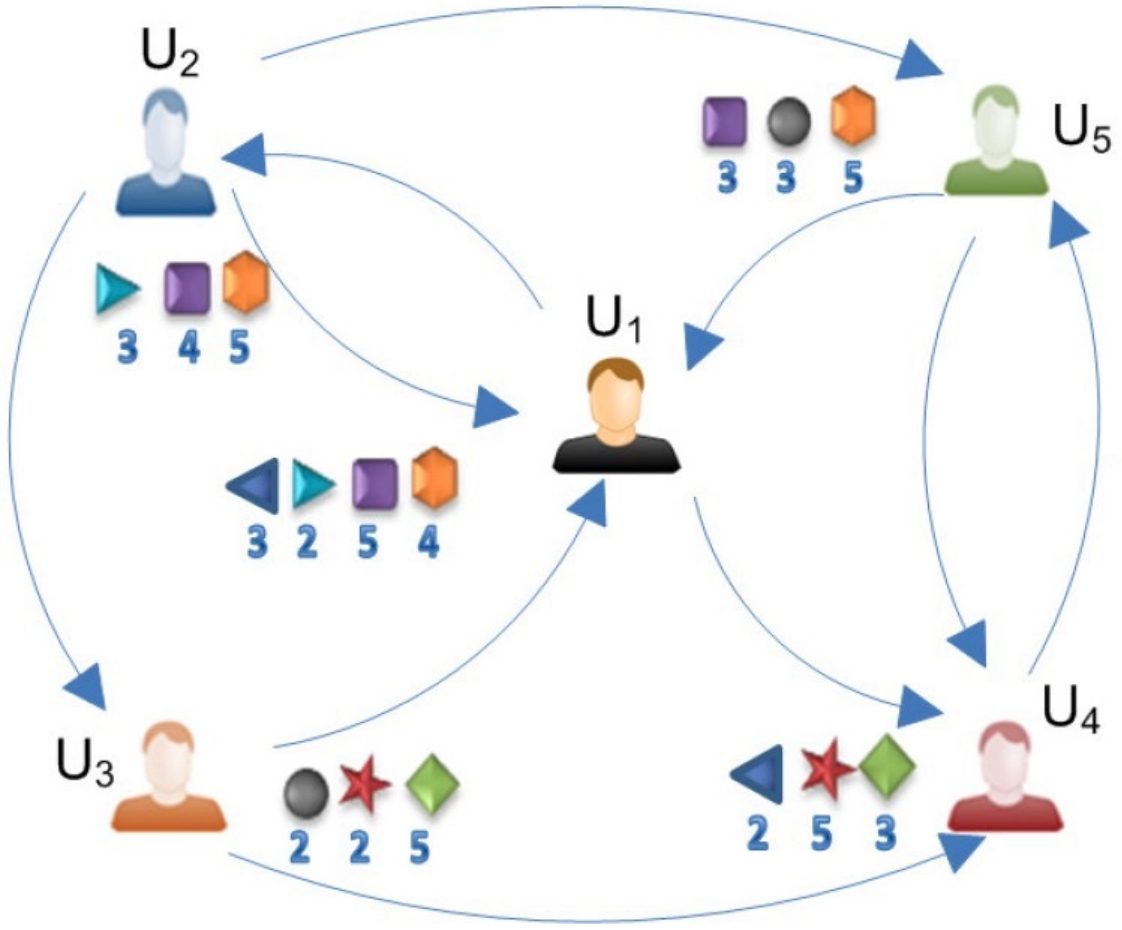
Treating each observation as an independent instance

- Components work as a black-box
  - hardly exhibit the reasons behind a recommendation
  - Make the decision-making process opaque to understand

Why i_1 is recommended to u_1? Which one is more important?

- Collaborative Signals?
- Mouth Marketing?
- Item Knowledge?

Recommendation in Social Networks



Effects in Social Networks

Social Influence

Ratings are influenced by ratings of friends, i.e., friends are more likely to have similar ratings than strangers

Benefits

- Can deal with cold-start users, as long as they are connected to the social network
- Exploit social influence, correlational influence, transitivity

Memory Based Approaches

- Explore the network to find raters in the neighborhood of the target user
- Aggregate the ratings of these raters to predict the rating of the target user
- Different methods to calculate the "trusted neighborhood" of users

TidalTrust

- Modified breadth-first search in the network
- Consider all raters v at the shortest distance from the target user u
- Trust between u and v:

$$t_{u,v} = \frac{\sum_{w \in N_u} t_{u,w} t_{w,v}}{\sum_{w \in N_u} t_{u,w}}$$

where N_u denotes the set of (direct) neighbors (friends) of u

- Trust depends on all connecting paths

Trust between direct neighbors

Can be based on profile similarity or a value provided by the users themselves.

TidalTrust

Predicted Rating

$$\hat{r}_{u,i} = \frac{\sum_{v \in raters} t_{u,v} r_{v,i}}{\sum_{v \in raters} t_{u,v}}$$

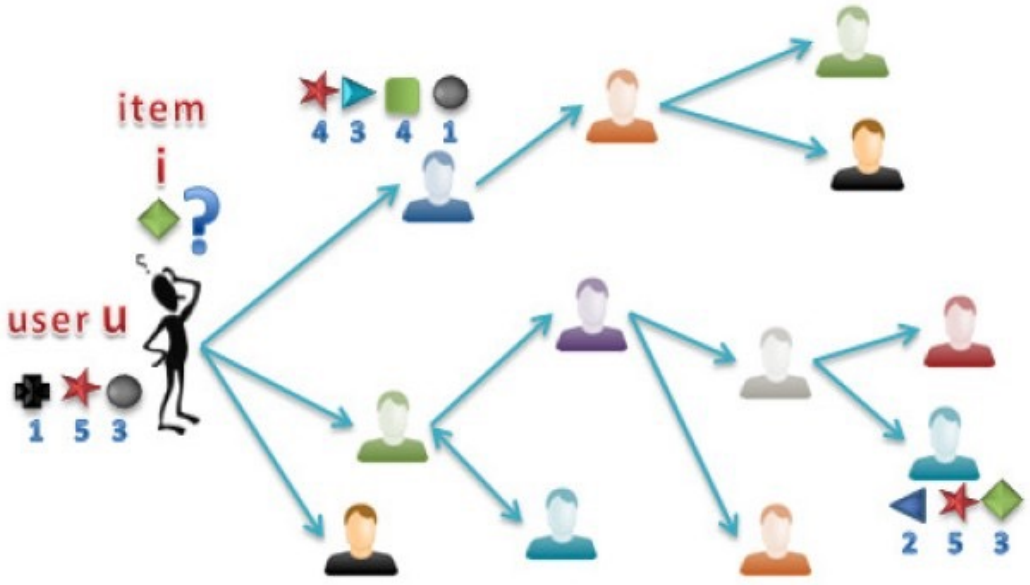
$r_{v,i}$ denotes rating of user v for item i

Shortest distance?

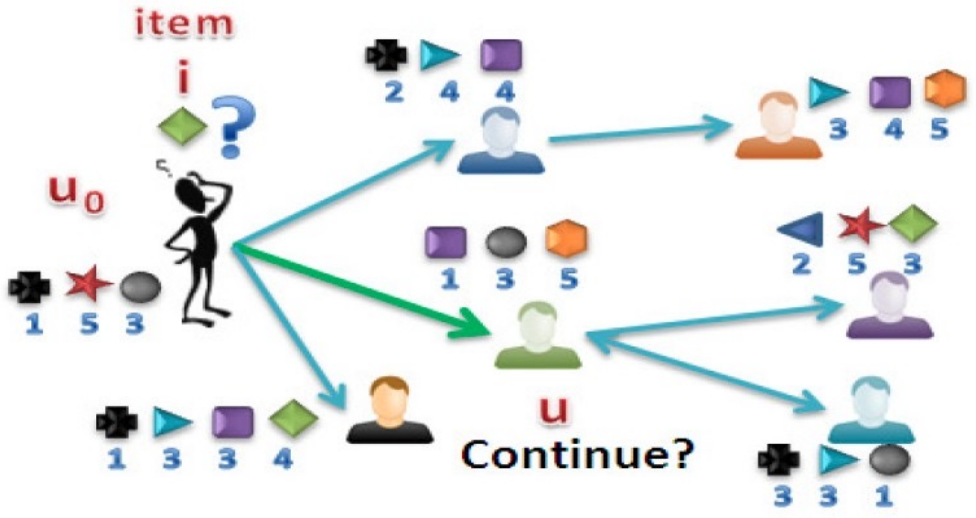
- Efficient
- Taking a short distance gives high precision and low recall
- One can consider raters up to a maximum-depth d, a trade-off between precision (and efficiency) and recall

TrustWalker

- How far to explore the network?
  - Trade-off between precision and coverage
- Instead of far neighbors who have rated the target item, use near neighbors who have rated similar items



Random Walk Starting from a Target User u_0



At step k, at node u

- If u has rated i, return $r_{u,i}$, otherwise
- With probability $\varphi_{u,i,k}$: stop random walk, randomly select item j rated by u and return $r_{u,j}$
- With probability $1 - \varphi_{u,i,k}$: continue the random walk to a direct neighbor of u

Selecting $\varphi_{u,i,k}$

- $\varphi_{u,i,k}$ gives the probability of staying at u to select one of its items at step k, while we are looking for a prediction on target item i
- This probability should be related to the similarities of the items rated by u and the target item i, consider the maximum similarity
- The deeper we go into the network, the probability of continuing random walk should decrease, so $\varphi_{u,i,k}$ should increase with k

$$\varphi_{u,i,k} = \max_{j \in RI_u} sim(i,j) \times \frac{1}{1 + e^{-\frac{k}{2}}}$$

where RI_u denotes the set of items rated by user u

Selecting $\varphi_{u,i,k}$

Selecting sim(i,j)

Let $UC_{i,j}$ be the set of common users, who have rated both items i and j, we can define the correlation between items i and j as:

$$corr(i,j) = \frac{\sum_{u \in UC_{i,j}} (r_{u,i} - \overline{r_u})(r_{u,j} - \overline{r_u})}{\sqrt{\sum_{u \in UC_{i,j}} (r_{u,i} - \overline{r_u})^2} \sqrt{\sum_{u \in UC_{i,j}} (r_{u,j} - \overline{r_u})^2}}$$

Taking the effect of common users

The size of the common users is also important. For the same value of corr(i,j), if number of common users, $|UC_{i,j}|$, is higher, the similarity should be higher

$$sim(i,j) = \frac{1}{1 + e^{-\frac{|UC_{i,j}|}{2}}} \times corr(i,j)$$

When does a random walk terminate?

Three alternatives

- Reaching a node which has expressed a rating on the target item i
- At some user node u, decide to stay at the node and select one of the items rated by u and return the rating for that item as result of the random walk
- The random walk might continue forever, so terminate when it is very far ($k > max-depth$). What value of k ?
- "six-degrees of separation"

How to recommend a rating?

Perform several random walks, as described before and the aggregation of all ratings returned by different random walks are considered as the predicted rating

Estimated rating for source user u on target item i:

$$\hat{r_{u_0,i}} = \sum_{\{(v,j)|R_{v,j}\}} P(XY_{u,i} = (v,j))r_{v,j}$$

$XY_{u,i}$ is the random variable for stopping the random walk at node v and selecting item j rated by v

Social Matrix Factorization

Intuition

Can we incorporate the Social information in the matrix factorization methods?

Recollect the Matrix factorization problem

$$min_{p*,q*} \sum_{(u,i)\in K} (r_{ui} - \hat{r_{ui}})^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

where r_{ui} is the actual rating given by user u to item i, $\hat{r_{ui}}$ approximates user u's rating of item i, simplest of the expression being $q_i^T p_u$, though other biases can also be incorporated.

Social Matrix Factorization

Basic Idea

Neighbors in the social network may have similar interests.

Incorporating social factors

- Let the social network information be represented by a matrix $S \in R^{u_0 \times u_0}$, where u_0 is the number of users.
- $S_{u,v} \in (0, 1]$ denotes the directed and weighted social relationship of user u with user v
- Each of the rows of the social matrix S is normalized to 1, resulting in the new matrix S^*, such that $\sum_v S^*_{u,v} = 1$ for each user u

Modified objective function

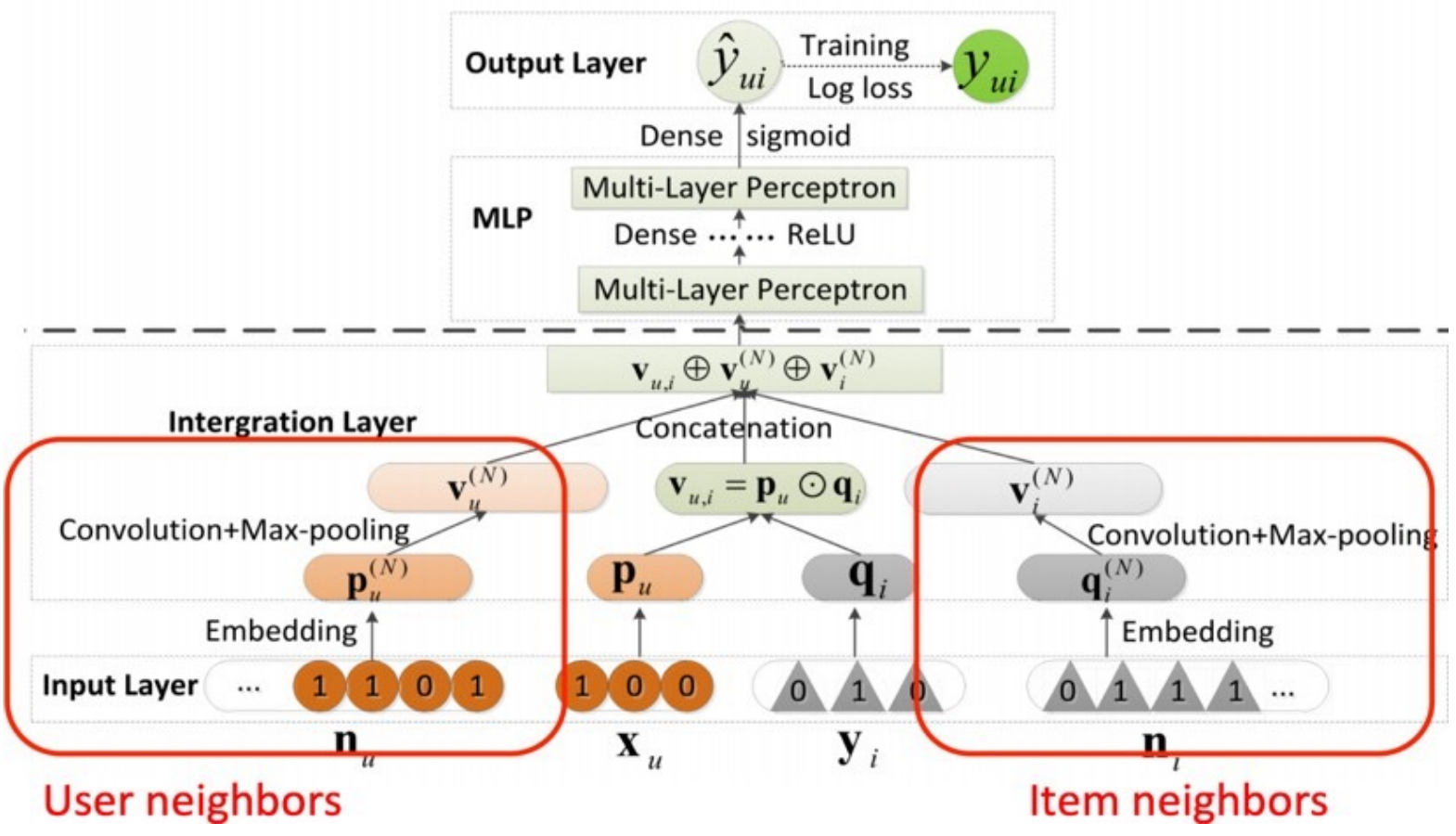
$$min_{p^*,q^*} \sum_{(u,i)\in K} (r_{ui} - \hat{r_{ui}})^2 + \beta \sum_{\text{all } u} ((p_u - \sum_v S^*_{u,v} p_v)(p_u - \sum_v S^*_{u,v} p_v)^T)$$

$$+\lambda(||q_i||^2 + ||p_u||^2)$$

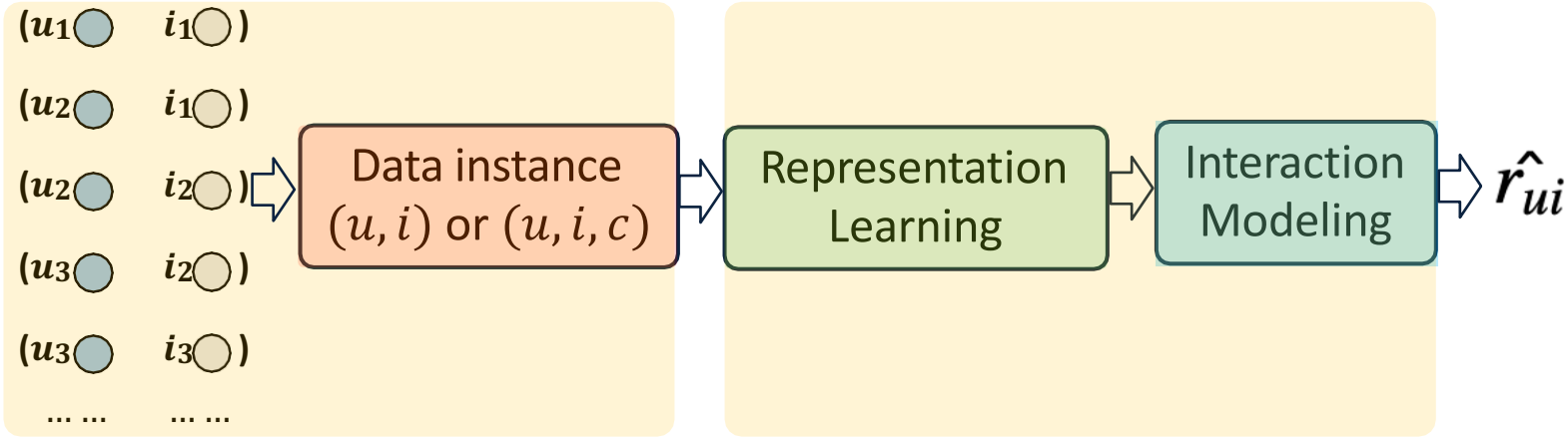
Neighbor-based Neural CF (CIKM'17)

Interaction Modeling → MF + MLP over user and item neighbors

- Feeding user and item neighbors into the NCF framework

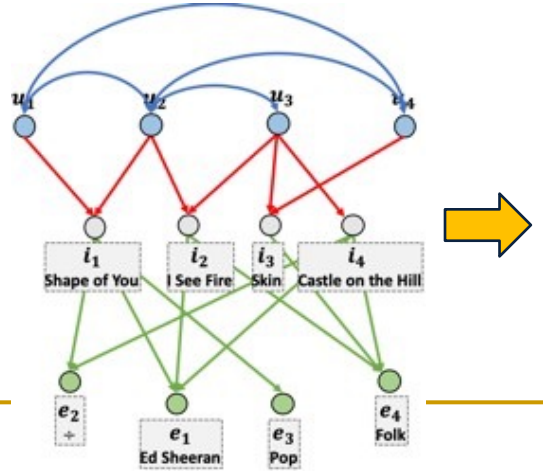


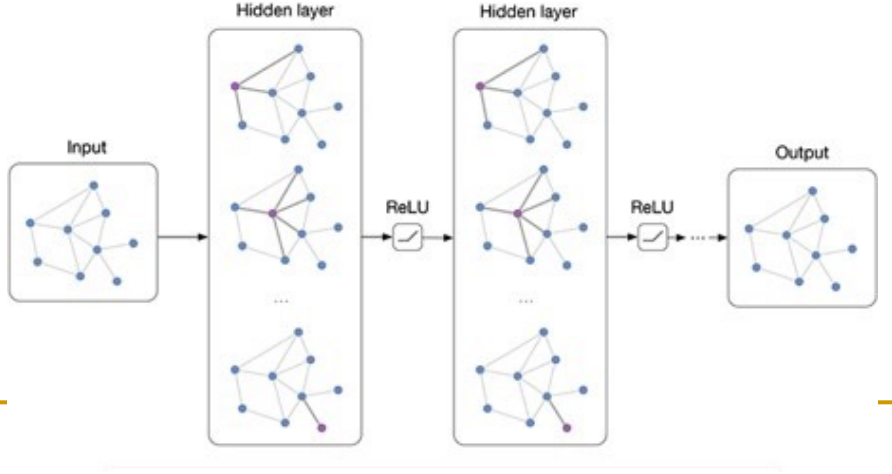
Through Graph Neural Nets



Explore & Exploit Relations among Instances

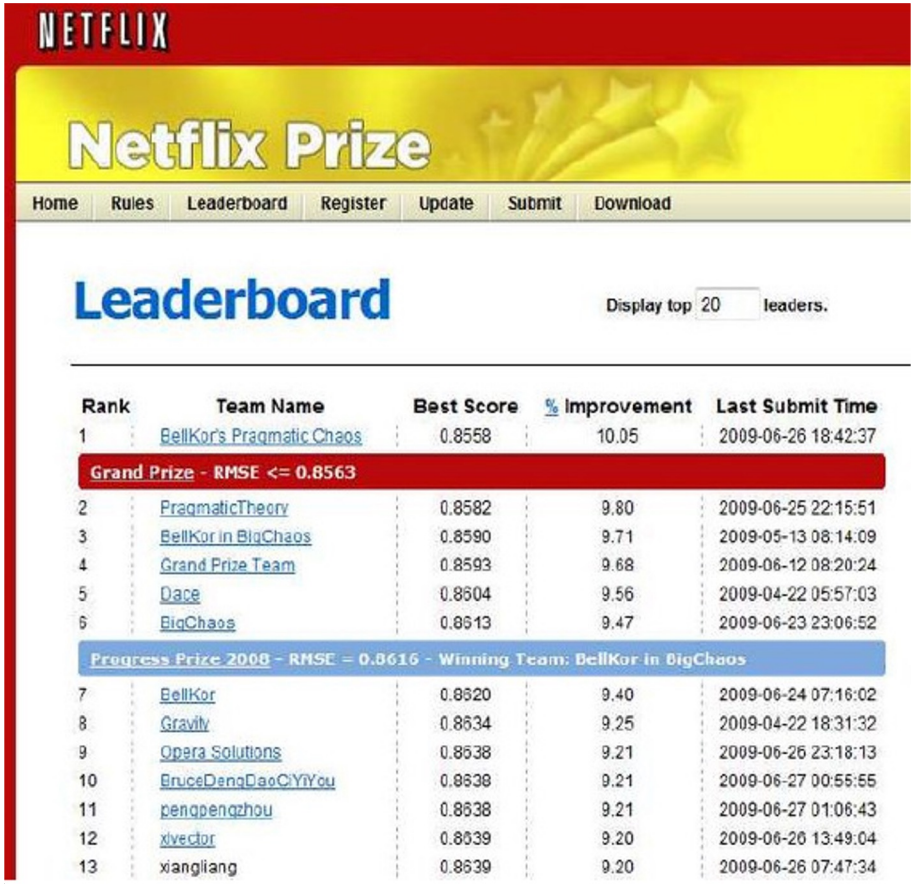
Apply Techniques of Graph Learning & Reasoning





Netflix Prize (2009)

- Netflix offered $1,000,000 to anyone beating their algorithm by 10% in **RMSE**
- Provided 100M (user, movie) ratings for training
- Held a testing set and allowed one guess/day on the testing set to create a leader board



NETFLIX

Netflix Prize

Home | Rules | Leaderboard | Register | Update | Submit | Download

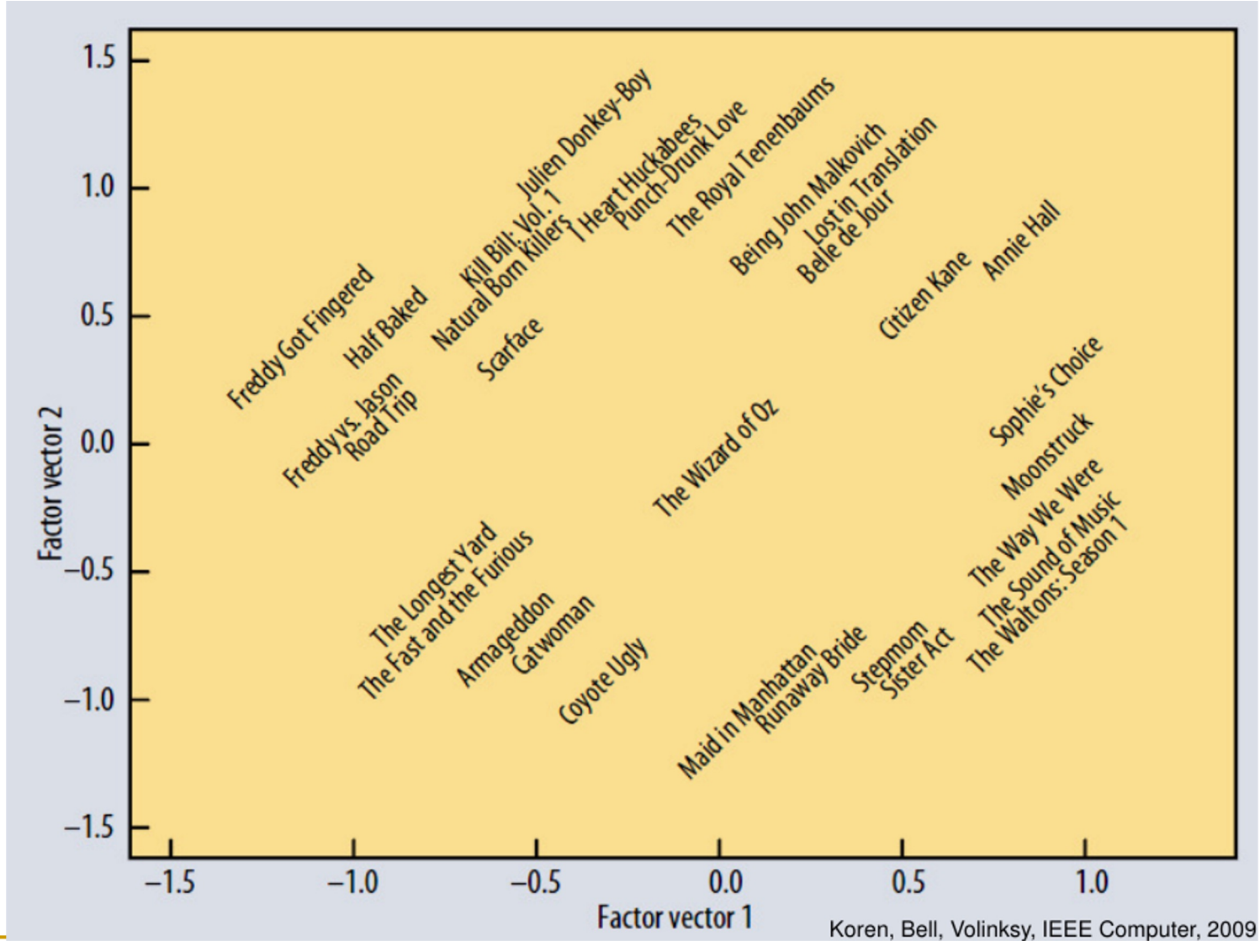
Leaderboard

Display top 20 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	BellKor's Pragmatic Chaos	0.8558	10.05	2009-06-26 18:42:37
Grand Prize - RMSE <= 0.8563				
2	PragmaticTheory	0.8582	9.80	2009-06-25 22:15:51
3	BellKor in BigChaos	0.8590	9.71	2009-05-13 08:14:09
4	Grand Prize Team	0.8593	9.68	2009-06-12 08:20:24
5	Dace	0.8604	9.56	2009-04-22 05:57:03
6	BigChaos	0.8613	9.47	2009-06-23 23:06:52
Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos				
7	BellKor	0.8620	9.40	2009-06-24 07:16:02
8	Gravity	0.8634	9.25	2009-04-22 18:31:32
9	Opera Solutions	0.8638	9.21	2009-06-26 23:18:13
10	BruceDengDaoCiYiYou	0.8638	9.21	2009-06-27 00:55:55
11	pengpengzhou	0.8638	9.21	2009-06-27 01:06:43
12	xivector	0.8639	9.20	2009-06-26 13:49:04
13	xiangliang	0.8639	9.20	2009-06-26 07:47:34

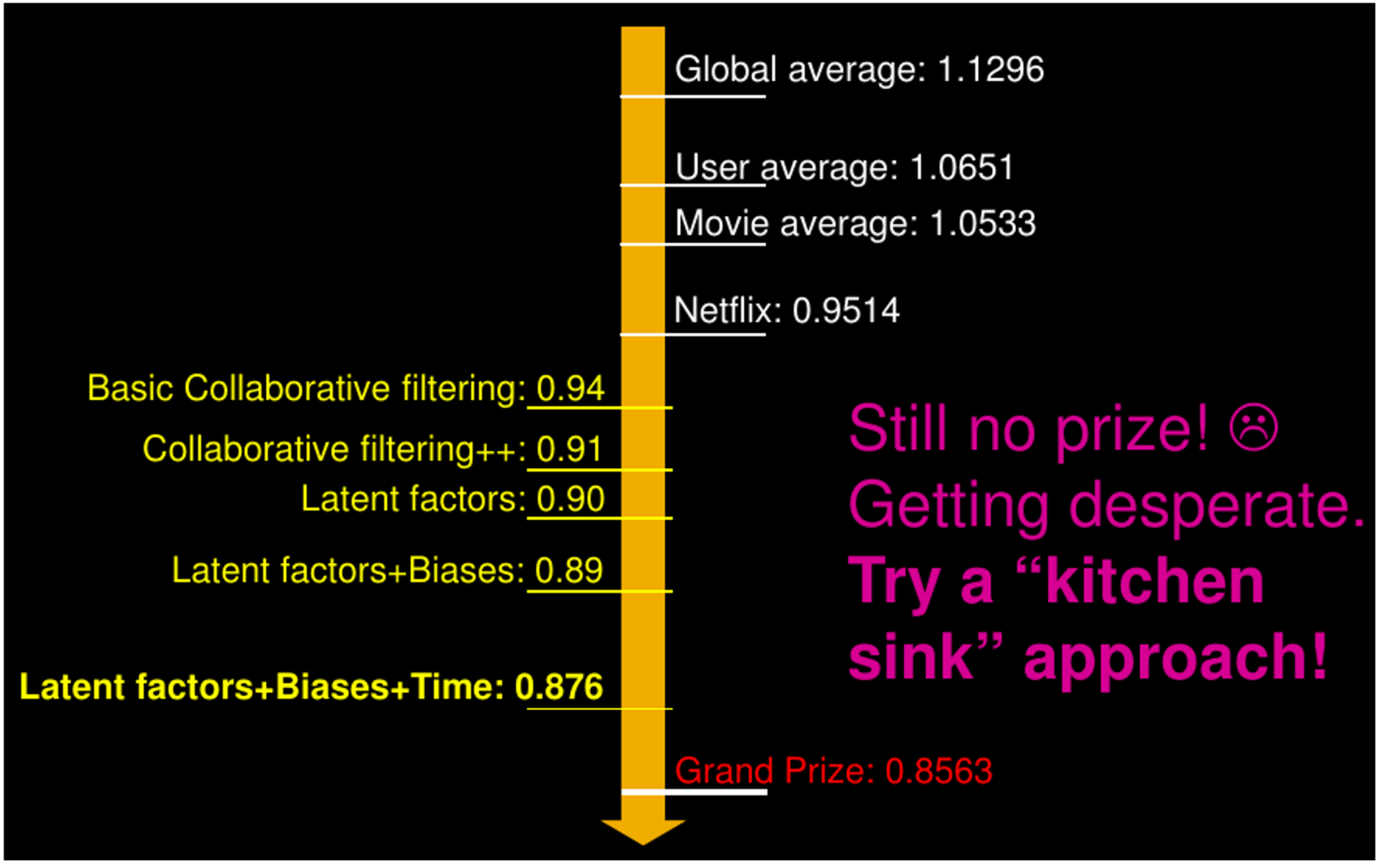
Latent Factors

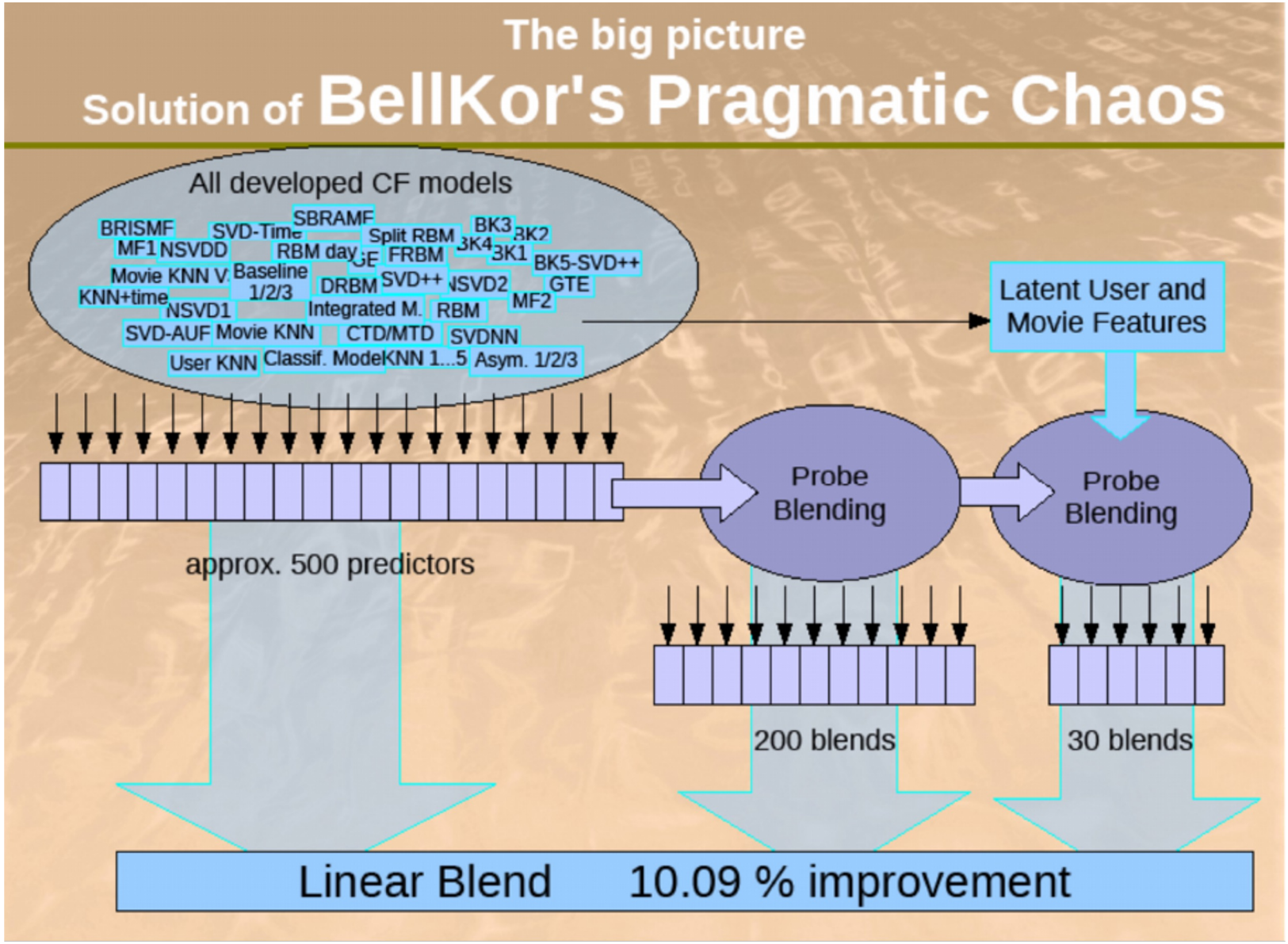
In latent factor space, similar movies are mapped to similar points



Koren, Bell, Volinksy, IEEE Computer, 2009

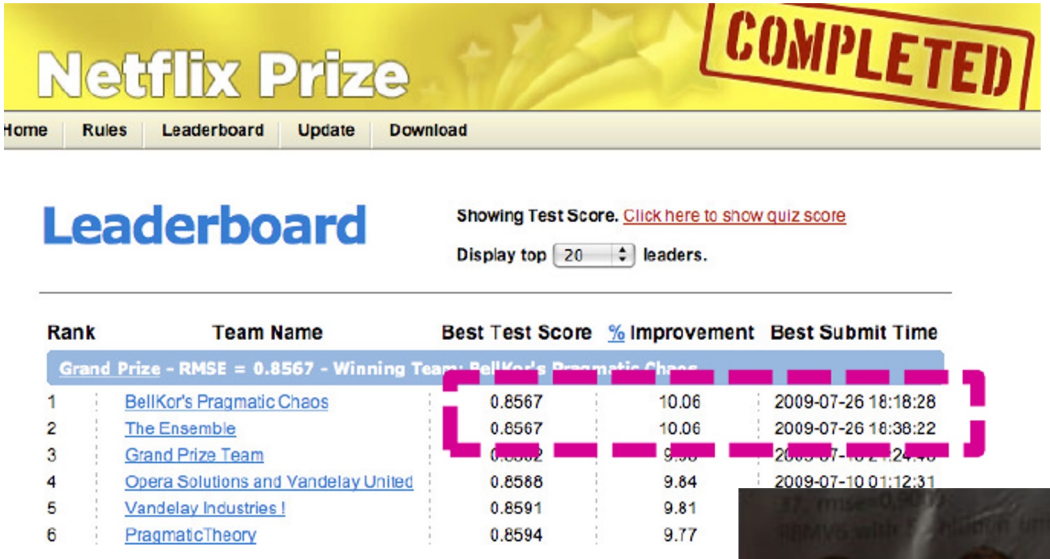
Shortly before Deadline …





https://www.asc.ohio-state.edu/statistics/statgen/joul_aut2009/BigChaos.pdf

The Grand Finale



26 July 2009.- Bellkor team submits 40 minutes before the deadline, "The Ensemble" team made of a mix of other teams submitted 20 minutes before the deadline.

Bellkor team wins one million dollars

The Netflix Prize revolutionized recommendation system research

Thank You

Slide Acknowledgements:
Prof. Tat-Seng Chua, National University of Singapore
Prof. Xiangnan He, University of Science and Technology of China
Prof. Saptarshi Ghosh, IIT Kharagpur
Prof. Pawan Goyal, IIT Kharagpur
Prof. Carlos Castillo, Universitat Pompeu Fabra