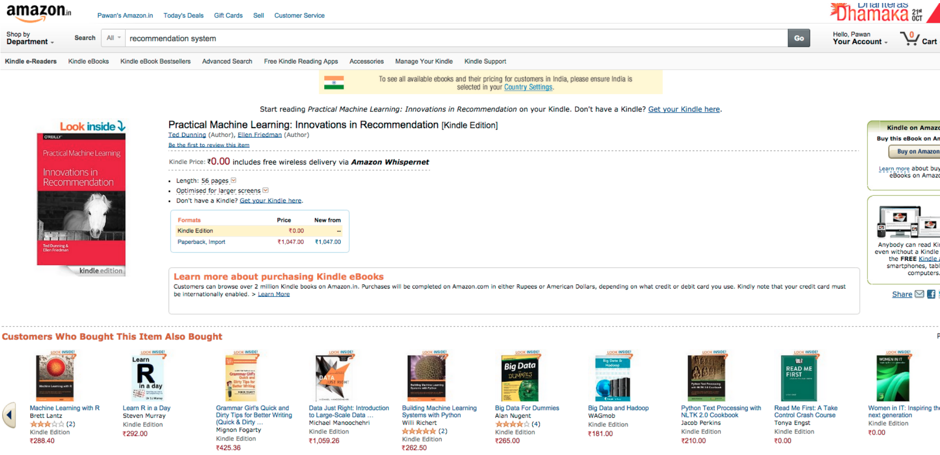
Recommendation Systems

Pawan Goyal

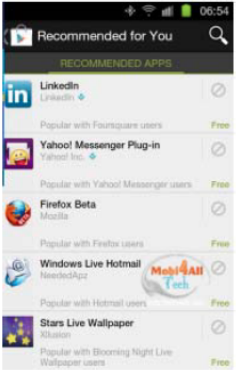
CSE, IITKGP

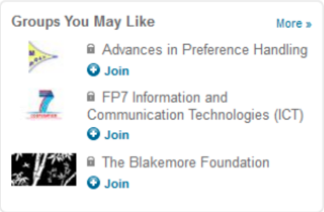
October 21, 2014

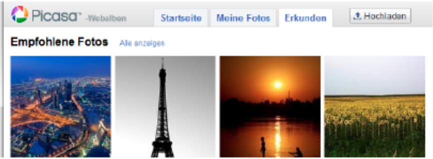
Recommendation System?



Recommendation in Social Web







Why using Recommender Systems?

Value for the customers

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

Why using Recommender Systems?

Value for the customers

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

Value for the provider

- 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

Real-world check

Myths from industry

- Amazon.com generates X percent of their sales through the recommendation lists ($X > 35\%$)
- Netflix generates X percent of their sales through the recommendation lists ($X > 30\%$)

Real-world check

Myths from industry

- Amazon.com generates X percent of their sales through the recommendation lists ($X > 35\%$)
- Netflix generates X percent of their sales through the recommendation lists ($X > 30\%$)

There must be some value in it

- See recommendation of groups, jobs or people on LinkedIn
- Friend recommendation and ad personalization on Facebook
- Song recommendation at last.fm
- News recommendation at Forbes.com ($+37\%$ CTR)

Recommender Systems as a function

What is given?

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

Recommender Systems 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

Recommender Systems 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

Recommender Systems 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

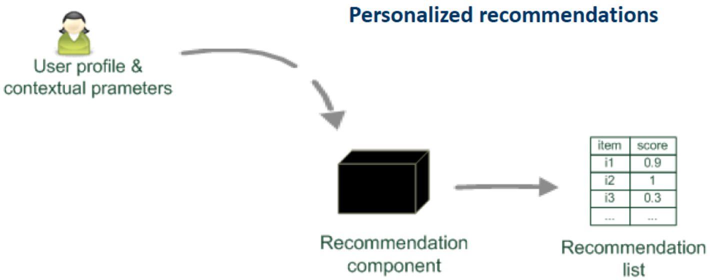
- Remember that relevance might be context-dependent
- Characteristics of the list might be important (diversity)

Recommender systems reduce information overload by estimating relevance

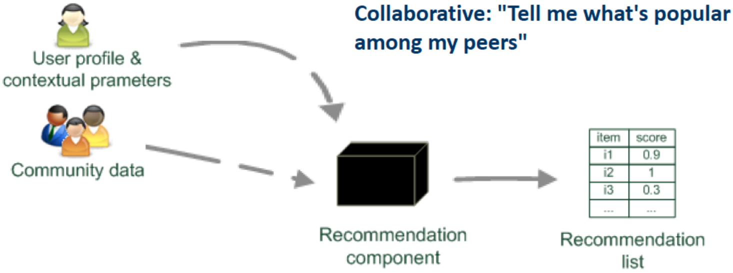
item	score
i1	0.9
i2	1
i3	0.3
...	...

Recommendation list

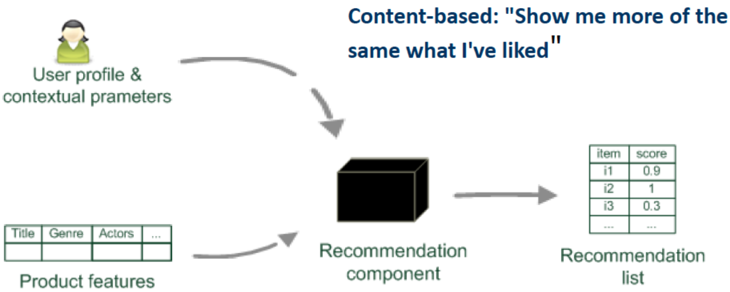
Paradigms of Recommender Systems



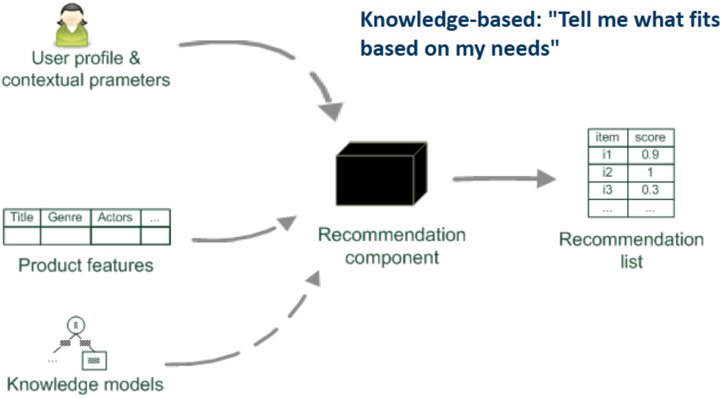
Paradigms of Recommender Systems



Paradigms of Recommender Systems

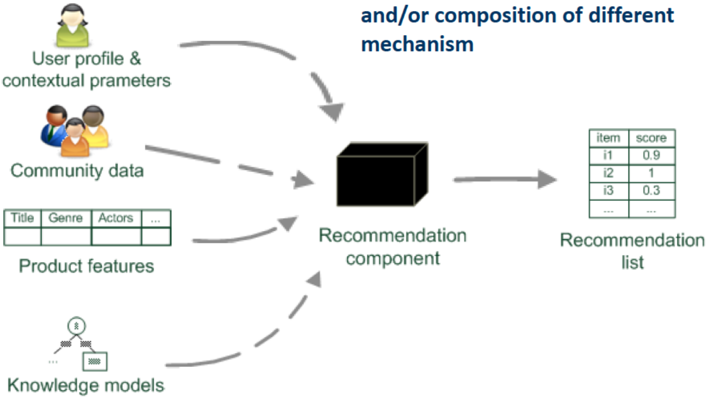


Paradigms of Recommender Systems



Paradigms of Recommender Systems

Hybrid: combinations of various inputs and/or composition of different mechanism



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
Knowledge-based	Deterministic recommendations, assured quality, no cold-start, can resemble sales dialogue	Knowledge engineering effort to bootstrap, basically static, does not react to short-term trends

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, ...)

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

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, e.g. 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

Some first 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

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

- 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

Popular similarity model

Pearson Correlation

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

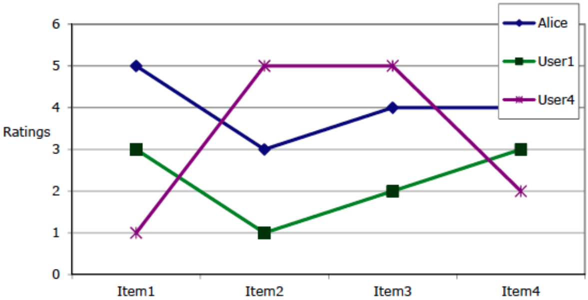
- a, b: users
- $r_{a,p}$: rating of user a for item p
- P: set of items, rated both by a and b
- $\overline{r_a}$, $\overline{r_b}$: user's average ratings
- Possible similarity values are between -1 to 1

For the example considered

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

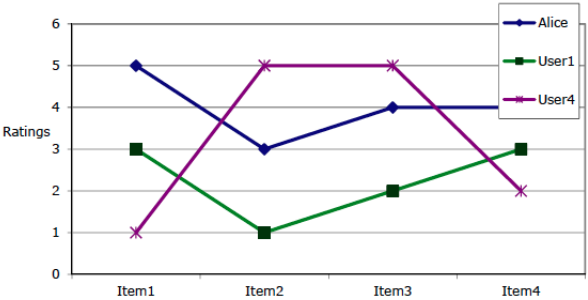
Pearson Correlation

Takes Difference in rating behavior into account



Pearson Correlation

Takes Difference in rating behavior into account



Works well in usual domains

Making Predictions

- A common prediction function:

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

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

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 taken into account 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 commonly 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?

More on ratings

Pure CF-based systems only rely on the rating matrix

Explicit ratings

- Most commonly 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?

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

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 CF

- Assume there is a very close neighbor n of u who however has not rated the target item i yet.

Example algorithms for sparse datasets

Recursive CF

- 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

Example algorithms for sparse datasets

Recursive CF

- 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

sim = 0.85

Predict rating for User1

Example algorithms for sparse datasets

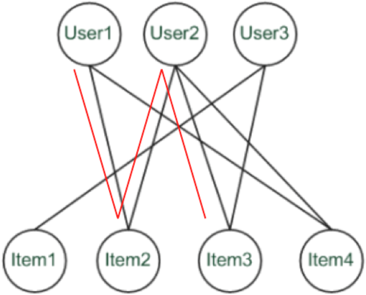
Graph-based methods: Spreading activation

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

Example algorithms for sparse datasets

Graph-based methods: Spreading activation

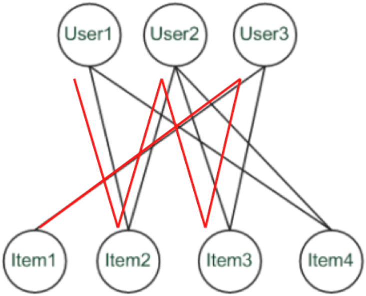
- Idea: Use paths of lengths 3 and 5 to recommend items
- Length 3: Recommend Item3 to User1
- Length 5: Item1 also recommendable



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

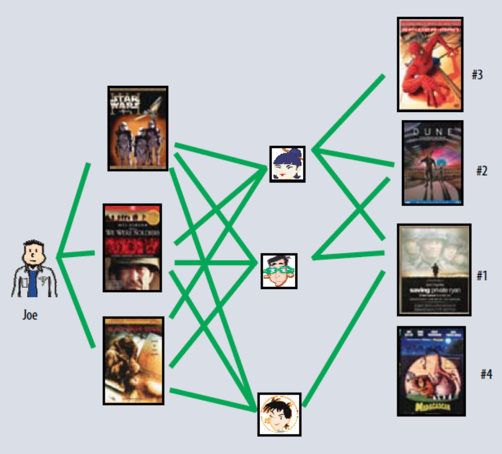


Figure 1. The user-oriented neighborhood method. Joe likes the three movies on the left. To make a prediction for him, the system finds similar users who also liked those movies, and then determines which other movies they liked. In this case, all three liked *Saving Private Ryan*, so that is the first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

Latent Factor Approach

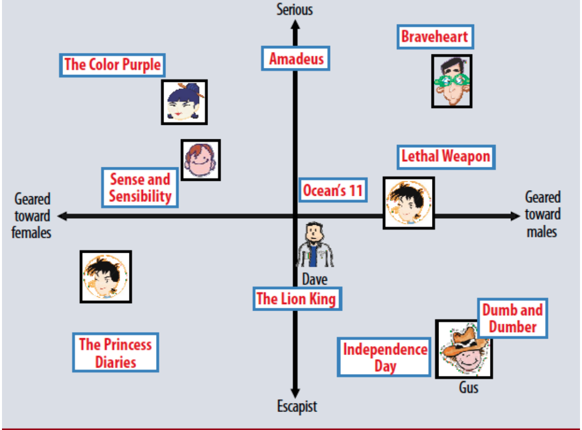


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

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

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: $\hat{r_{ui}} = \overline{r_u} + U_k(u) \times \Sigma_k \times {V_k}^T(i)$
- The problem, however, is the high portion of missing values
- Using only relatively few entries may lead to overfitting

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 associated with a vector $q_i \in R^f$, and each user u 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 $q_i, p_u \in R^f$

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.

Stochastic Gradient Descent

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

Let $e_{ui} = r_{ui} - {q_i}^T p_u$
Gradient descent can be written as

- $q_i \leftarrow q_i + \gamma(e_{ui}p_u - \lambda q_i)$
- $p_u \leftarrow p_u + \gamma(e_{ui}q_i - \lambda p_u)$

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

Adding Biases

An Example

- You want a first-order estimate for user Joe's rating of the movie *Titanic*.
- Let the average rating over all movies, μ, is 3.7 stars
- Titanic tends to be rated 0.5 stars above the average
- Joe is a critical user, who tends to rate 0.3 stars lower than the average
- Thus, the estimate for Titanic's rating by Joe would be (3.7+0.5-0.3) = 3.9 stars

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 interaction
The squared error function:

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

Additional Input Sources

- Many users may supply very few ratings
- Difficult to reach general conclusions on their taste

Additional Input Sources

- Many users may supply very few ratings
- Difficult to reach general conclusions on their taste
- Incorporate additional sources of information about the users
- E.g., gather implicit feedback, use purchases or browsing history to learn the tendencies

Modeling Implicit Feedback

Boolean Implicit Feedback

- $N(u)$: set of items for which user u expressed an implicit preference
- Let item i be associated with $x_i \in R^f$
- The user can be characterized by the vector $\sum_{i \in N(u)} x_i$
- Normalizing the sum: $\frac{\sum_{i \in N(u)} x_i}{\sqrt{|N(u)|}}$

Modeling Demographics

- Consider boolean attributes where user u corresponds to a set of attributes $A(u)$
- These attributes can describe gender, age group, Zip code, income level etc.
- Let a feature vector $y_a \in R^f$ correspond to each attribute to describe a user through this set as: $\sum_{a \in A(u)} y_a$

Integrating enhanced user representation in the matrix factorization model:

$$\hat{r_{ui}} = \mu + b_i + b_u + {q_i}^T [p_u + |N(u)|^{-0.5} \sum_{i \in N(u)} x_i + \sum_{a \in A(u)} y_a]$$

Adding Temporal Dynamics

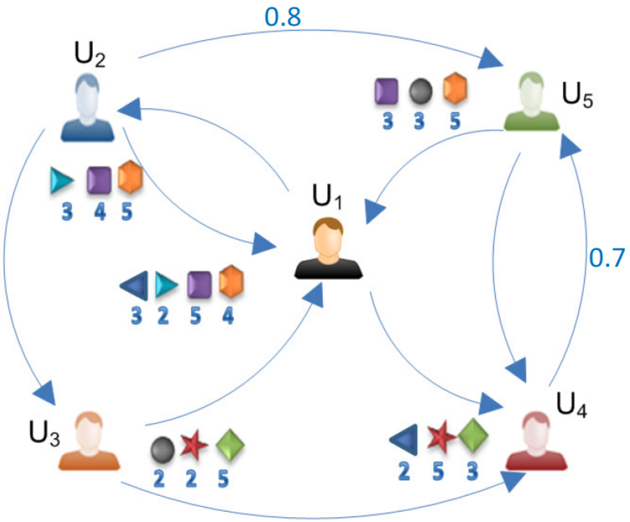
- In reality, product perception and popularity constantly change as new selections emerge
- Customers' inclinations evolve, leading them to redefine their taste
- The system should account for the temporal effects reflecting the dynamic, time-drifting nature of user-item interactions

Adding Temporal Dynamics

- In reality, product perception and popularity constantly change as new selections emerge
- Customers' inclinations evolve, leading them to redefine their taste
- The system should account for the temporal effects reflecting the dynamic, time-drifting nature of user-item interactions
- Items that can vary over time: item biases, $b_i(t)$; user biases, $b_u(t)$; user preferences, $p_u(t)$
- It can be integrated in the matrix factorization model as:

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

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

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
- Are more robust to fraud, in particular to profile attacks

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; Goldbeck (2005)

- 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

TidalTrust

Predicted Rating

$$\hat{r_{u,i}} = \frac{\sum\limits_{v \in raters} t_{u,v} r_{v,i}}{\sum\limits_{v \in raters} t_{u,v}}$$

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

TidalTrust

Predicted Rating

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

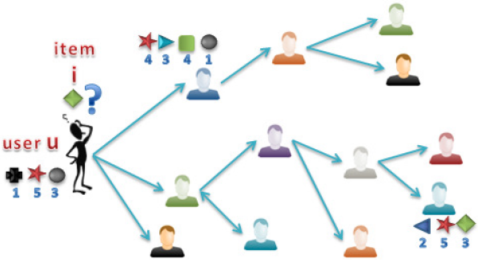
$r_{v,i}$ denotes rating of user v for item i

Shortest distance?

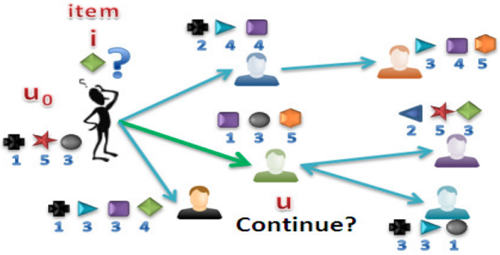
- Efficient
- Taking a short distance gives high precision and low recall
- One can consider raters up to a maximum-depth d, a trade-off between precision (and efficiency) and recall

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

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

- $\phi_{u,i,k}$ 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 $\phi_{u,i,k}$ should increase with k

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

where RI_u denotes the set of items rated by user u

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

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:

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

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

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 ?

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 $\hat{r_{u_0,i}}$