Collaborative Filtering

Radek Pelánek

Movie Recommendations

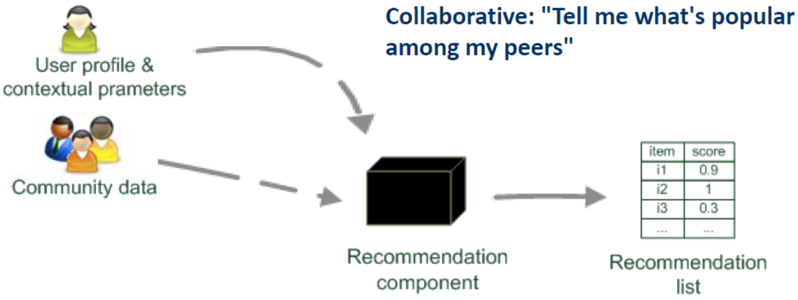
How would you do it?

- data
- computation, algorithms
- presentation

Notes on Lecture

- the most technical lecture of the course
- includes some “math with many indices”, but typically with intuitive interpretation
- use of standard machine learning techniques, which are briefly described
- projects: at least basic versions of the presented algorithms (when relevant)

Collaborative Filtering: Basic Idea



Collaborative Filtering

- simplifying assumption: users with similar taste in past will have similar taste in future
- requires only matrix of ratings
  $\Rightarrow$ applicable in many domains
- widely used in practice

Basic CF Approach

- input: matrix of user-item ratings (with missing values, often very sparse)
- output: predictions for missing values

Netflix Prize

- Netflix – video rental company
- contest: 10% improvement of the quality of recommendations
- prize: 1 million dollars
- data: user ID, movie ID, time, rating

Non-personalized Recommendations

Let us start with something simple:

non-personalized recommendations

How?

Non-personalized Recommendations

"most popular items"

- compute average rating for each item
- recommend items with highest averages
- (filter those already known to user)

problems?

Non-personalized Predictions

“averages”, issues:

- uncertainty, size of data
  - average 5 from 3 ratings
  - average 4.9 from 100 ratings
- bias, normalization
  - some users give systematically higher ratings
  - ratings not distributed uniformly
  - (specific example in later lecture)

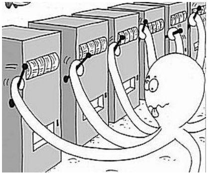
⇒ even a simple idea like “most popular items” is not that simple to realize properly

Exploitation vs Exploration

- “pure exploitation” – always recommend “top items”
- what if some other item is actually better, rating is poorer just due to noise?
- “exploration” – presenting items to get more data
- how to balance exploration and exploitation?
  - too much exploitation: we may omit some very good items
  - too much exploration: we present poor items needlessly

Multi-armed Bandit

- standard model for *exploitation vs exploration*
- arm $\Rightarrow$ (unknown) probabilistic reward
- how to choose arms to maximize reward?
- well-studied, many algorithms (e.g., *upper confidence bounds*)
- related to *reinforcement learning*
- typical application: online advertisements



Core Idea

- do not use just "averages"
- quantify uncertainty (e.g., standard deviation)
- combine average & uncertainty for decisions
- example: TrueSkill, ranking of players (leaderboard)

- systematic approach: Bayesian statistics
- pragmatic approach: $U(n) \sim \frac{1}{n}$, roulette wheel selection, ...

Personalized Techniques

now we want to make recommendations:

- personalized: based on the user's previous rankings
- collaborative filtering: using other user's rankings

Main CF Techniques

- memory based
  - *find "similar" users/items, use them for prediction*
  - nearest neighbors (user, item)
- model based
  - *model "taste" of users and "features" of items*
  - latent factors
  - matrix factorization

Neighborhood Methods: Illustration

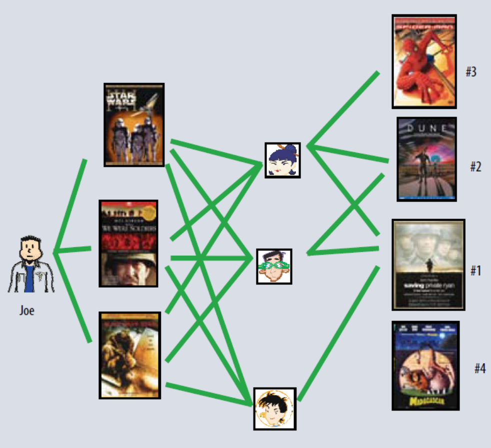


Figure 1. The user-oriented neighborhood method. Joe likes the three movies on the left. To make a prediction for him, the system finds similar users who also liked those movies, and then determines which other movies they liked. In this case, all three liked *Saving Private Ryan*, so that is the first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

Matrix factorization techniques for recommender systems

Latent Factors: Illustration

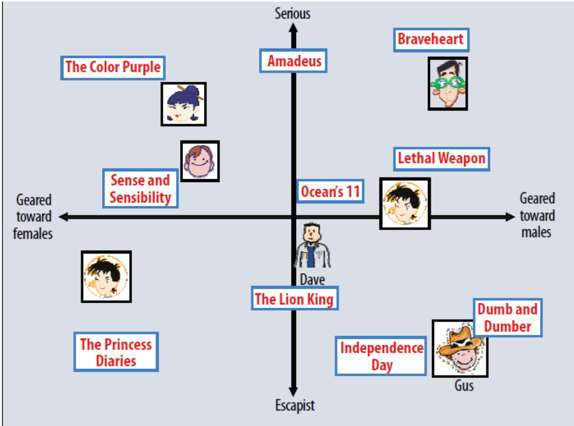


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

Matrix factorization techniques for recommender systems

Latent Factors: Netflix Data

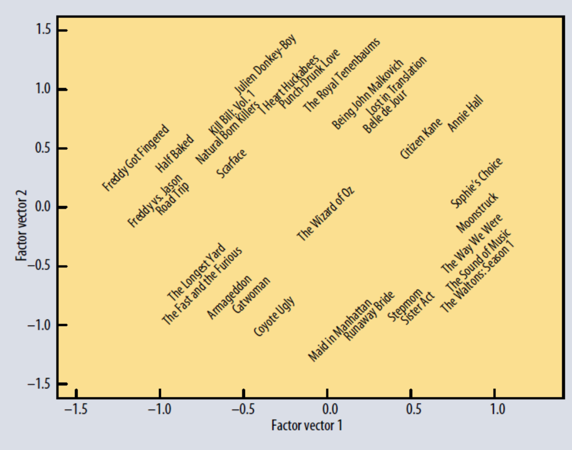


Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films.

Matrix factorization techniques for recommender systems

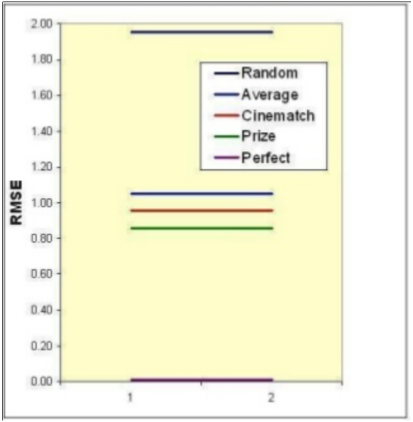
Ratings

- explicit
  - e.g., "stars" (1 to 5 Likert scale)
  - issues: users may not be willing to rate $\Rightarrow$ data sparsity
- implicit
  - "proxy" data for quality rating
  - clicks, page views, time on page

the following applies directly to explicit ratings, modifications may be needed for implicit (or their combination)

Note on Improving Performance

- simple predictors often provide reasonable performance
- further improvements often small
- but can have significant impact on behavior (not easy to evaluate)
- $\Rightarrow$ evaluation lecture



Introduction to Recommender Systems, Xavier Amatriain

User-based Nearest Neighbor CF

user Alice:

- item i not rated by Alice:
  - find "similar" users to Alice who have rated i
  - compute average to predict rating by Alice
- recommend items with highest predicted rating

User-based Nearest Neighbor CF

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

Recommender Systems: An Introduction (slides)

User Similarity

Pearson correlation coefficient (alternatives: Spearman cor. coef., cosine similarity, ...)

	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

sim = 0,85

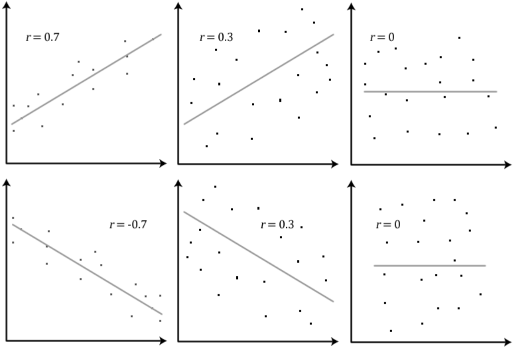
sim = 0,00

sim = 0,70

sim = -0,79

Recommender Systems: An Introduction (slides)

Pearson Correlation Coefficient: Reminder



$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

Test your intuition: `https://www.umimematiku.cz/rozhodovacka-korelacni-koeficient-2`

Making Predictions: Naive

r_{ai} – rating of user a, item i

neighbors $N = k$ most similar users
prediction $=$ average of neighbors' ratings

$$pred(a, i) = \frac{\sum_{b \in N} r_{bi}}{|N|}$$

improvements?

Making Predictions: Naive

r_{ai} – rating of user a, item i

neighbors $N = k$ most similar users
prediction $=$ average of neighbors' ratings

$$pred(a, i) = \frac{\sum_{b \in N} r_{bi}}{|N|}$$

improvements?

- user bias: consider difference from average rating $(r_{bi} - \overline{r_b})$
- user similarities: weighted average, weight $sim(a, b)$

Making Predictions

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

- r_{ai} – rating of user a, item i
- $\overline{r_a}, \overline{r_b}$ – user averages

Improvements

- number of co-rated items
- agreement on more “exotic” items more important
- case amplification – more weight to very similar neighbors
- neighbor selection

Item-based Collaborative Filtering

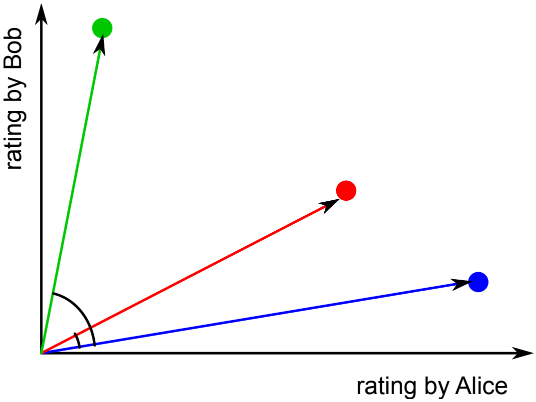
- compute similarity between **items**
- use this similarity to predict ratings
- more computationally efficient, often:
  number of items $<<$ number of users
- practical advantage (over user-based filtering): feasible to check results using intuition

Item-based Nearest Neighbor CF

	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

Recommender Systems: An Introduction (slides)

Cosine Similarity



$$\cos(\alpha) = \frac{A \cdot B}{\|A\| \|B\|}$$

Similarity, Predictions

- (adjusted) cosine similarity – similar to Pearson's r, works slightly better
- $$pred(u,p) = \frac{\sum_{i \in R} sim(i,p) r_{ui}}{\sum_{i \in R} sim(i,p)}$$
- neighborhood size limited (20 to 50)

Notes on Similarity Measures

- Pearson's r? (adjusted) cosine similarity? other?
- no fundamental reason for choice of one metric
- mostly based on practical experiences
- may depend on application

Preprocessing

- $O(N^2)$ calculations – still large
- original article: *Item-item recommendations by Amazon* (2003)
- calculate similarities in advance (periodical update)
- supposed to be stable, item relations not expected to change quickly
- reductions (min. number of co-ratings etc)

Practical Note: NaNs and zeros

common student mistake:

- creating matrix of ratings
- replacing missing values (NaN) by zeros
- computing similarity measures (e.g., Pearson r)
- using similarity measures for recommendations

Where is the mistake? Why is it a problem?

Matrix Factorization CF

- main idea: latent factors of users/items
- use these to predict ratings
- related to singular value decomposition

Notes

- singular value decomposition (SVD) – theorem in linear algebra
- in CF context the name “SVD” usually used for an approach only slightly related to SVD theorem
- related to “latent semantic analysis” and “embeddings”
- introduced during the Netflix prize, in a blog post (Simon Funk)

http://sifter.org/~simon/journal/20061211.html

Singular Value Decomposition (Linear Algebra)

$$X = USV^T$$

- U, V orthogonal matrices
- S diagonal matrix, diagonal entries $\sim$ singular values

low-rank matrix approximation (use only top k singular values)

$$\underset{m \times n}{\overset{X}{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}}} = \underset{m \times r}{\overset{U}{\begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}}} \underset{r \times r}{\overset{S}{\begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix}}} \underset{r \times n}{\overset{V^{\mathsf{T}}}{\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}}}$$

http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/svd.html

SVD – CF Interpretation

$$X = USV^T$$

- X – matrix of ratings
- U – user-factors strengths
- V – item-factors strengths
- S – importance of factors

Latent Factors

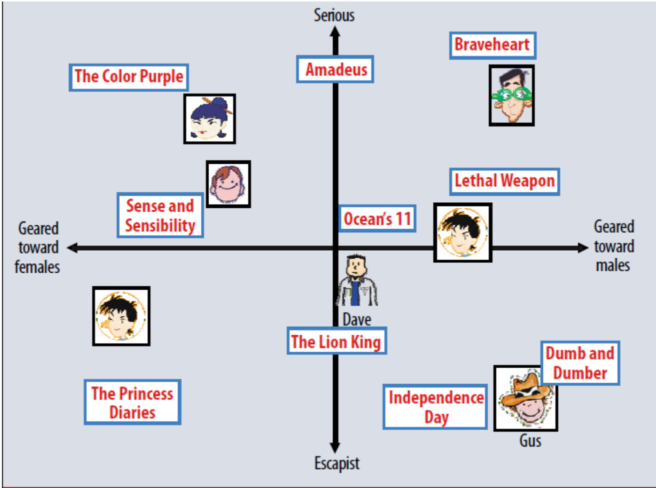


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

Latent Factors

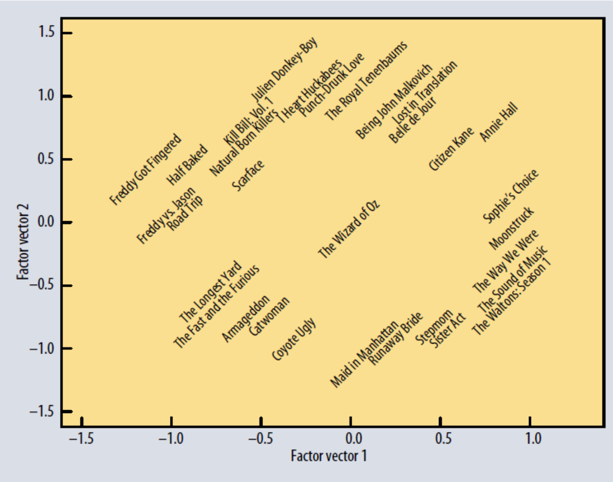
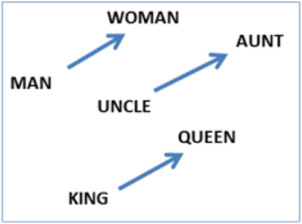
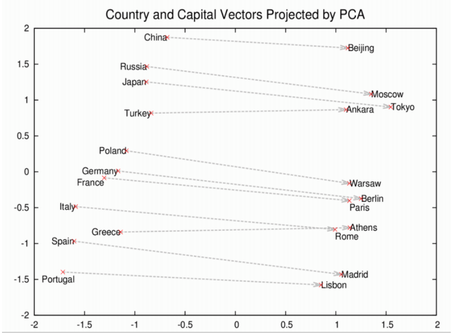


Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films.

Matrix factorization techniques for recommender systems

Sidenote: Embeddings, Word2vec





Missing Values

- matrix factorization techniques (SVD) work with full matrix
- ratings – sparse matrix
- solutions:
  - value imputation – expensive, imprecise
  - alternative algorithms (greedy, heuristic): gradient descent, alternating least squares

Notation

- u – user, i – item
- r_{ui} – rating
- $\hat{r}_{ui}$ – predicted rating
- b, b_u, b_i – bias
- q_i, p_u – latent factor vectors (length k)

Simple Baseline Predictors

[note: always use baseline methods in your experiments]

- naive: $\hat{r}_{ui} = \mu$, μ is global mean
- biases: $\hat{r}_{ui} = \mu + b_u + b_i$
  - b_u, b_i – biases, average deviations
  - some users/items – systematically larger/lower ratings

Latent Factors

(for a while assume centered data without bias)

$$\hat{r}_{ui} = q_i^T p_u$$

- vector multiplication
- user-item interaction via latent factors

illustration (3 factors):

- user (p_u): $(0.5, 0.8, -0.3)$
- item (q_i): $(0.4, -0.1, -0.8)$

Latent Factors

$$\hat{r}_{ui} = q_i^T p_u$$

- vector multiplication
- user-item interaction via latent factors

we need to find q_i, p_u from the data (cf content-based techniques)

note: finding q_i, p_u at the same time

Learning Factor Vectors

- we want to minimize "squared errors" (related to RMSE, more details leater)

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

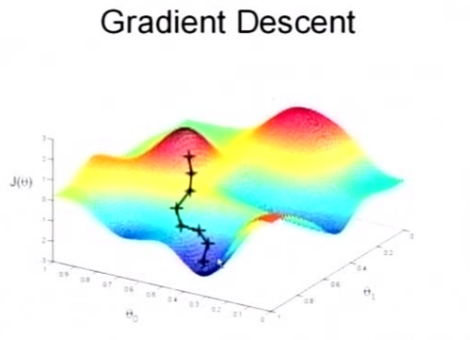
- regularization to avoid overfitting (standard machine learning approach)

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

How to find the minimum?

Stochastic Gradient Descent

- standard technique in machine learning
- greedy, may find local minimum



Gradient Descent for CF

- prediction error $e_{ui} = r_{ui} - q_i^T p_u$
- update (in parallel):
  - $q_i := q_i + \gamma(e_{ui} p_u - \lambda q_i)$
  - $p_i := p_u + \gamma(e_{ui} q_i - \lambda p_u)$
- math behind equations – gradient = partial derivatives
- γ, λ – constants, set "pragmatically"
  - learning rate γ (0.005 for Netflix)
  - regularization λ (0.02 for Netflix)

Advice

if you want to learn/understand gradient descent (and also many other machine learning notions) experiment with linear regression

- can be (simply) approached in many ways: analytic solution, gradient descent, brute force search
- easy to visualize
- good for intuitive understanding
- relatively easy to derive the equations

Advice II

recommended sources:

- Koren, Yehuda, Robert Bell, and Chris Volinsky. *Matrix factorization techniques for recommender systems.* Computer 42.8 (2009): 30-37.
- Koren, Yehuda, and Robert Bell. *Advances in collaborative filtering.* Recommender Systems Handbook. Springer US, 2011. 145-186.

Adding Bias

predictions:

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

function to minimize:

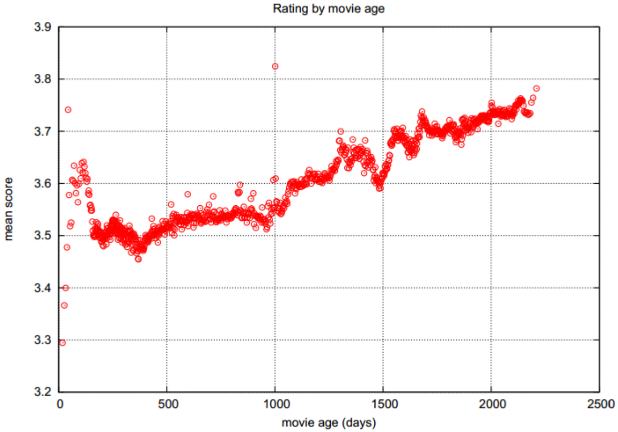
$$\min_{q,p} \sum_{(u,i) \in T} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2 + b_u^2 + b_i^2)]$$

Improvements

- additional data sources (implicit ratings)
- varying confidence level
- temporal dynamics

Temporal Dynamics

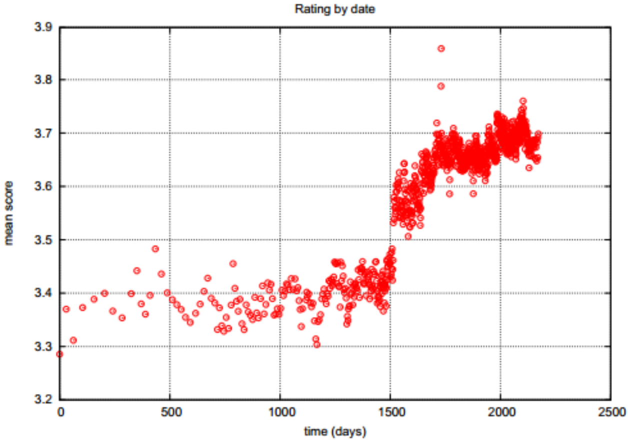
Netflix data



Y. Koren, Collaborative Filtering with Temporal Dynamics

Temporal Dynamics

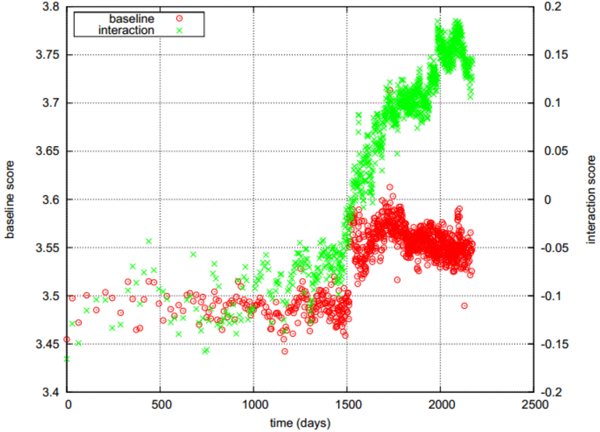
Netflix data, jump early in 2004



Y. Koren, Collaborative Filtering with Temporal Dynamics

Temporal Dynamics

baseline = behaviour influenced by exterior considerations
interaction = behaviour explained by match between users and items



Y. Koren, Collaborative Filtering with Temporal Dynamics

Results for Netflix Data

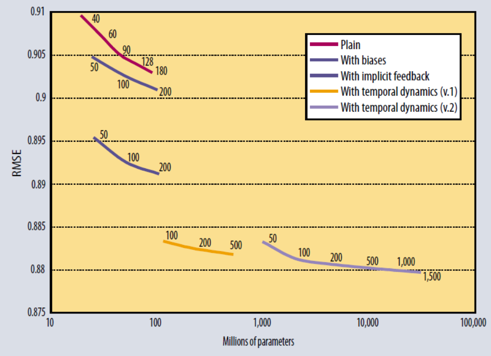
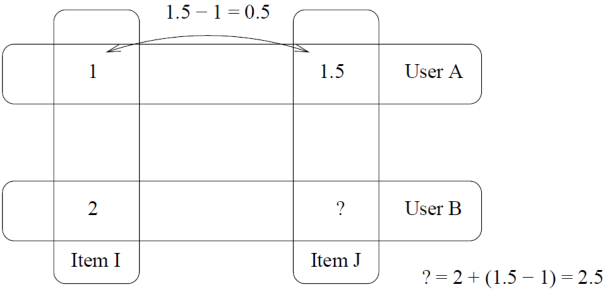


Figure 4. Matrix factorization models' accuracy. The plots show the root-mean-square error of each of four individual factor models (lower is better). Accuracy improves when the factor model's dimensionality (denoted by numbers on the charts) increases. In addition, the more refined factor models, whose descriptions involve more distinct sets of parameters, are more accurate. For comparison, the Netflix system achieves RMSE = 0.9514 on the same dataset, while the grand prize's required accuracy is RMSE = 0.8563.

Matrix factorization techniques for recommender systems

Slope One

Slope One Predictors for Online Rating-Based Collaborative Filtering



average over such simple prediction

Slope One

- accurate within reason
- easy to implement
- updateable on the fly
- efficient at query time
- expect little from first visitors

Other CF Techniques

- clustering
- association rules
- classifiers

Clustering

main idea:

- cluster similar users
- non-personalized predictions ("popularity") for each cluster

Clustering

	Book1	Book2	Book3	Book4	Book5	Book6
Customer A	X			X		
Customer B		X	X		X	
Customer C		X	X			
Customer D		X				X
Customer E	X				X	

Customers B, C and D are « clustered » together. Customers A and E are clustered into another separate group

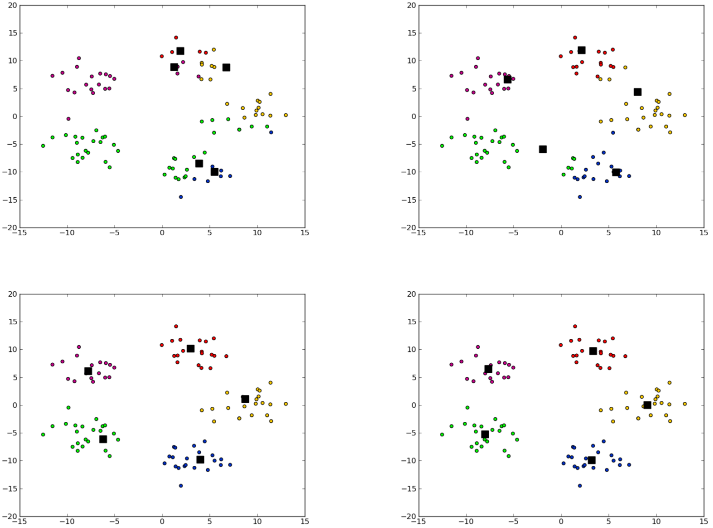
- « Typical » preferences for **CLUSTER** are:
  - Book 2, very high
  - Book 3, high
  - Books 5 and 6, may be recommended
  - Books 1 and 4, not recommended at all

Introduction to Recommender Systems, Xavier Amatriain

Clustering

- unsupervised machine learning
- many algorithms – k-means, EM algorithm, . . .

Clustering: K-means



Association Rules

- relationships among items, e.g., common purchases
- famous example (google it for more details): “beer and diapers”
- general machine learning algorithms
- “Customers Who Bought This Item Also Bought...”
  - advantage: provides explanation, useful for building trust
- closely related to item-based collaborative filtering

Classifiers

- general machine learning techniques
- positive / negative classification
- train, test set
- logistic regression, support vector machines, decision trees, Bayesian techniques, ...

Limitations of CF

Limitations of CF

- cold start problem
- impact of noise (e.g., one account used by different people)
- possibility of attacks
- popularity bias – difficult to recommend items from the long tail

Cold Start Problem

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

Cold Start Problem

- use another method (non-personalized, content-based ...) in the initial phase
- ask/force user to rate items
- use defaults (means)
- better algorithms (e.g., recursive CF)

Collaborative Filtering: Summary

- requires only ratings, widely applicable
- neighborhood methods, latent factors
- use of machine learning techniques