Machine Learning
CS 4780/5780 - Fall 2012
Cornell University
Department of Computer Science

Recommendation Systems

Antonio Bahamonde

Centro de Inteligencia Artificial.
Universidad de Oviedo at Gijón. Spain

Spanish Association for Artificial Intelligence

From 9/2012 till 7/2013
Visiting Professor in Cornell University

Contents

- Definitions
- Netflix Prize
- Similarity based methods
- Matrix factorization
- References

RS: Definitions

RS help to match users with items

- Ease information overload
- *Sales* assistance (guidance, advisory, persuasion,...)

RS are software agents that elicit the interests and preferences of individual consumers [...] and make recommendations accordingly.

Different system designs / paradigms

- Based on availability of exploitable data
- Implicit and explicit user feedback
- Domain characteristics

Purpose and success criteria

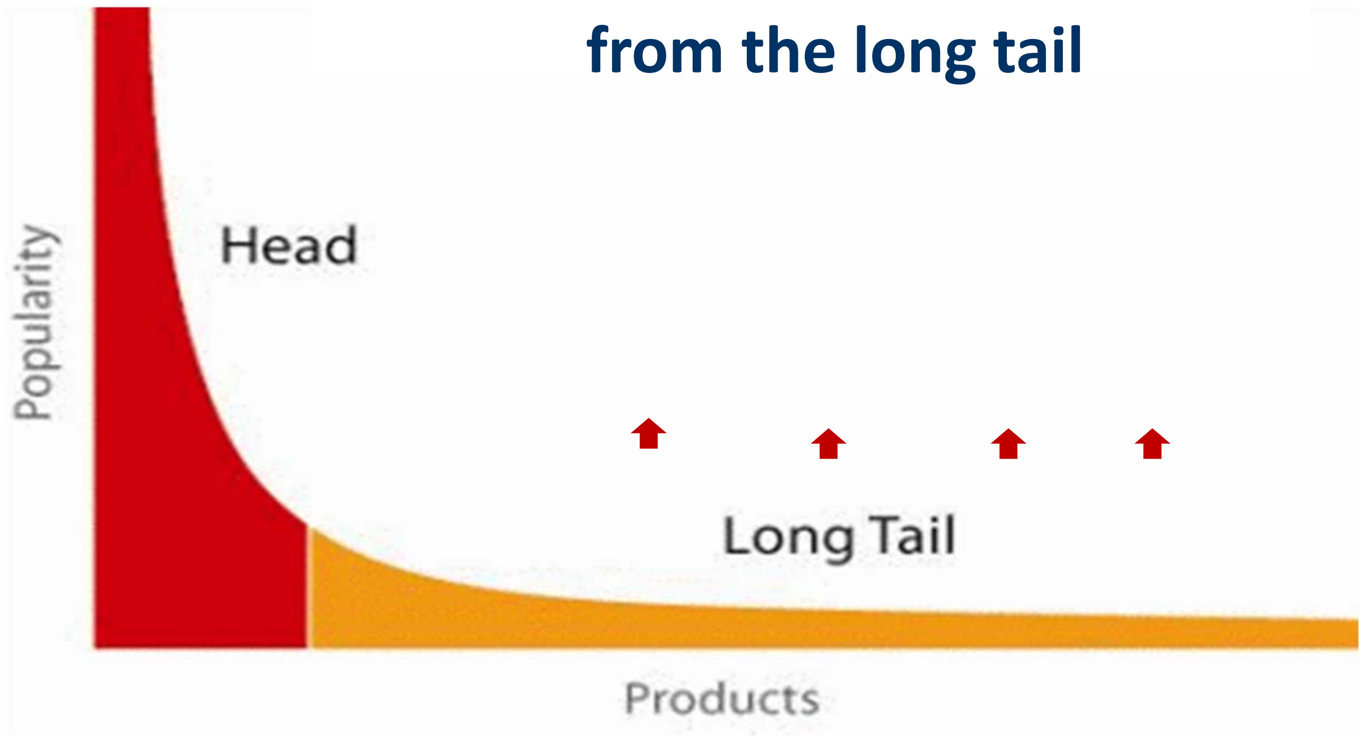
Retrieval perspective

- Reduce search costs
- Provide "correct" proposals – Users know in advance what they want

Recommendation perspective

- *Serendipity*– identify items from the Long Tail –Users did not know about existence

Recommend items from the long tail



Purpose and success criteria

Prediction perspective

- Predict to what degree users like an item
- Most popular evaluation scenario in research

Interaction perspective

- Give users a "good feeling" – Educate users about the product domain
- Convince/persuade users - explain

Finally, commercial perspective

- Increase "hit", "clickthrough", "lookers to bookers" rates
- Optimize sales margins and profit

Popular RS

- Google
- Genius (Apple)
- last.fm
- Amazon
- Netflix
- TiVo

Popular RS: Waze

Now in the first line of actuality since Apple suggested its use instead of the *failed* new maps in iOS6.

Popular RS
(in everyday life)

- Bestseller lists
- Top 40 music lists
- The "recent returns" shelf at the library
- Unmarked but well-used paths thru the woods
- Most popular lists in newspapers
- Many weblogs
- "Read any good books lately?"

Collaborative Filtering

The most prominent approach to RS

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

Approach

- use the "wisdom of the crowd" to recommend items (**crowdsourcing**)

Basic assumption and idea

- users give ratings to catalog items (implicitly or explicitly)
- customers who had similar tastes in the past, will have similar tastes in the future

Collaborative Filtering

- Relate two fundamentally different entities: users and items

  explicit feedback (ratings)
  implicit (purchase or browsing history, search patterns, ...)
  sometimes items descriptions by feature (content based)

- Approaches:

  neighborhood
  latent factor

Naïve Neighborhood approach

- item-item
- user-user

	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

Compute similarity and then prediction

Naïve Neighborhood approach

- user-user

	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

Naïve Neighborhood approach

- item-item

	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

Naïve Neighborhood approach

- How to measure similarities?
    - correlation $\rho(\boldsymbol{a}, \boldsymbol{b})$
    - cosine $cos(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a}, \boldsymbol{b}}{\|\boldsymbol{a}\| \|\boldsymbol{b}\|}$

$$b(u, i) = \overline{r(u, \cdot)} + \frac{\sum_{u' \neq u} sim(u, u')(r(u', i) - \overline{r(u', \cdot)})}{\sum_{u' \neq u} sim(u, u')}$$

similar formula for item-item

Naïve Neighborhood approach

user-user using 2 neighbors

$\overline{r(u,\cdot)}$	$\rho(alice,u)$		item1	item2	item3	item4	item5
4,00		alice	5	3	4	4	?
2,25	0,853	user1	3	1	2	3	3
3,50	0,707	user2	4	3	4	3	5
3,00	0	user3	3	3	1	5	4
3,25	-0,792	user4	1	5	5	2	1

$$b(alice, item5) = 4 + \frac{(3 - 2.25) * 0.853 + (5 - 3.5) * 0.707}{0.853 + 0.707} = 5.09$$

Naïve Neighborhood approach

item-item
using 2 neighbors

	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
$\rho(item5, i)$	0,969	-0,478	-0,428	0,582	
$r(\cdot, i)$	3,2	3,0	3,2	3,4	3,25

$$b(alice, item5) = 3.25 + \frac{(5-3.2)*0.969 + (4-3.4)*0.582}{0.969+0.582} = 4.6$$

Naïve Neighborhood approach

- Scalability
    - user-user is a memory based method
    - millions of users
    - does not scale for most real-world scenarios

Naïve Neighborhood approach

- item-item
    - is model based
    - models learned offline are stored for predictions at run-time
    - allows explanations
    - no cold start problem

Naïve Neighborhood approach

- However in all cases
  - not all neighbors should be taken into account (similarity thresholds)
  - not all item are rated (co-rated)
  - not involved the loss function

Netflix Prize

The New York Times (Sep, 21, 2009):

Netflix Awards $1 Million Prize and Starts a New Contest

[...]try to predict what movies particular customers would prefer

"Accurately predicting the movies Netflix members will love is a key component of our service," said Neil Hunt, chief product officer (Netflix)

Netflix Prize

The Netflix dataset

- more than 100 million movie ratings (1-5 stars)
- Nov 11, 1999 and Dec 31, 2005
- about 480, 189 users and n = 17, 770 movies
- 99% of possible rating are missing
  - movie average 5600 ratings
  - user rates average 208 movies
- training and quiz (test-prize) data

Netflix Prize

The loss function: root mean squared error (RMSE)

$$RMSE = \sqrt{\frac{1}{|Quiz|} \sum_{(u,i) \in Quiz} (r(u,i) - b(u,i))^2}$$

Netflix had its own system, Cinematch, which achieved 0.9514.

The prize was awarded to a system that reach RMSE below 0.8563 (10% improvement)

Netflix Prize

- Leaderboard

	Team	RMSE	Date	Hour
1	BellKor's Pragmatic Chaos	0,8567	26/07/09	18:18:28
2	The Emsemble	0,8567	26/07/09	18:38:22
3	Grand Prize Team	0,8582	10/07/09	21:24:40
4	Opera Solutions and Vandelay United	0,8588	10/07/09	01:12:31

Netfilx prize winners

Yehuda Koren, Robert M. Bell: Advances in Collaborative Filtering. Recommender Systems Handbook 2011: 145-186

Yehuda Koren,	Yahoo! Research
Robert Bell,	AT&T Labs – Research

Discuss similarity and matrix factorization approaches

Similarity approach revisited

3 major components:

- data normalization
- neighbor selection
- determination of interpolation weights

Baseline approach

Example: Titanic and Joe

- Average in Netflix: 3.7
- Joe critical: 0.3 less than average
- Titanic: 0.5 more than average (all users)

b(Joe,Titanic) = 3.7 - 0.3 + 0.5 = 3.9

$$b(u,i) = \mu + \overline{b(u,\cdot)} + \overline{b(\cdot,i)}$$
$$b_{u,i} = \mu + b_u + b_i$$

Baseline approach

The equations sound appealing, but Koren and Bell propose to learn it using a least square approach:

$$\min_{b_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 (\sum_u b_u^2 + \sum_i b_i^2)$$

where

$$\mathcal{K} = \{(u,i) \mid r_{ui} \text{ is known}\}$$

and the last term is a regularization term to avoid overfitting

Baseline approach

In the Netflix data

The average rating (μ) = 3.6

Learned bias

user (b_u) has average 0.044

their absolute value ($|b_u|$) is 0.32

item (movie) (b_i) has average -0.26 (s.d. 0.48),

($|b_i|$) is 0.43

	baseline	Cinematch	Prize
RMSE	0,9799	0,9514	0,8567

Improvements

- Koren & Bell report significant improvements adding
  - temporal dynamics (temporal drift of preferences)
  - implicit feedback (movie rental history (not available; rated movies!),

	baseline	+ temporal	+tem (spline)	Cinematch	Prize
RMSE	0,9799	0,9771	0,9603	0,9514	0,8567

Matrix factorization

- Tries to capture users and items relationships
- Based on well-known algebraic decomposition of matrices used before in Information Retrieval (LSI)
- Intended idea: consider latent variables
- As implemented by Koren and Bell, this approach won the Netflix Prize

Matrix factorization

- Transform both items and users into a feature space of lower dimensionality (k), the latent space
- Tries to explain ratings by characterizing both items and users on factors automatically learned from data.
- Factors might measure aspects as comedy, drama, amount of action, ...
- Efficient implementation offline
- Admit improvements in temporal drift and implicit feedback

Matrix factorization (SVD)

SVD (singular value decomposition). Matrix M with rows users and columns items

$$M = U * S * (Items)^T$$

where

$$U^T U = (Items)^T (Items) = I$$

$$S = \mathrm{diag}(s_1, \ldots, s_n), \quad s_1 \geq s_2, \ldots s_{n-1} \geq s_n \geq 0$$

$$\sqrt{S} = \mathrm{diag}(\sqrt{s_1}, \ldots, \sqrt{s_n})$$

and then

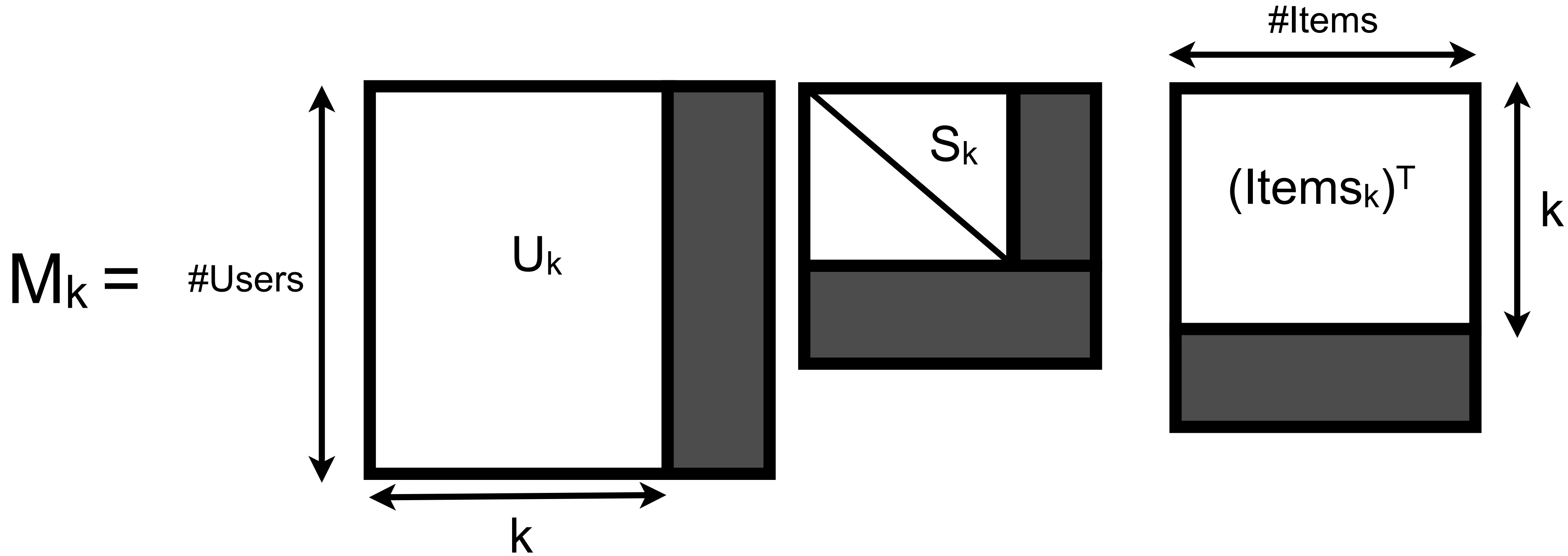
$$M = (U * \sqrt{S}) * (\sqrt{S} * Items^T)$$

Matrix factorization (SVD)

If we use only k dimensions

$$M \cong M_k = U_k * S_k * (Items_k)^T$$

This is a matrix of rank k. The most similar to M with this rank.



$$M_k = (U_k * \sqrt{S_k}) * (\sqrt{S_k} * Items_k^T)$$

Matrix factorization

Algorithm

Let M = r_{ui} - (μ) - b_u - b_i; %fill missing values with 0

[U S Items] = svd(M); % fix k <= rank(M);

U_rep = U_k *sqrt(S_k); % call p_u row u-th of U_rep

Items_rep = $Items_k$ * sqrt(S_k); %call q_i row i-th of Items_rep

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

$$M_k = (U_k * \sqrt{S_k}) * (\sqrt{S_k} * Items_k^T)$$

Matrix factorization

In this case M

	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

b_u
0,80
-0,80
0,60
0,00
-0,40

b_i	0,00	-0,20	0,00	0,20	0,00

μ	3,20

Matrix factorization

$M = r_{ui} - (\mu) - b_u - b_i$;   %fill missing values with 0

	item1	item2	item3	item4	item5
alice	1,0	-0,8	0,0	-0,2	0,0
user1	0,6	-1,2	-0,4	0,4	0,6
user2	0,2	-0,6	0,2	-1,0	1,2
user3	-0,2	0,0	-2,2	1,6	0,8
user4	-1,8	2,4	2,2	-1,0	-1,8

Matrix factorization

M [U S Items] = svd(M);

U =

-0,143	0,348	0,445	-0,500	0,640
-0,290	0,185	0,148	0,842	0,389
-0,097	0,478	-0,838	-0,096	0,226
-0,406	-0,762	-0,263	-0,118	0,414
0,849	-0,188	-0,096	0,136	0,465

S =

4,971	0,000	0,000	0,000	0,000
0,000	2,591	0,000	0,000	0,000
0,000	0,000	1,258	0,000	0,000
0,000	0,000	0,000	0,440	0,000
0,000	0,000	0,000	0,000	0,000

Matrix factorization

[U S Items] = svd(M);

Items =

-0,359	0,403	0,471	-0,536	-0,447
0,515	-0,478	-0,208	-0,513	-0,447
0,575	0,496	0,111	0,459	-0,447
-0,300	-0,581	0,385	0,474	-0,447
-0,431	0,159	-0,758	0,116	-0,447

Matrix factorization

For k = 2

U_k =

-0,143	0,348
-0,290	0,185
-0,097	0,478
-0,406	-0,762
0,849	-0,188

S_k =

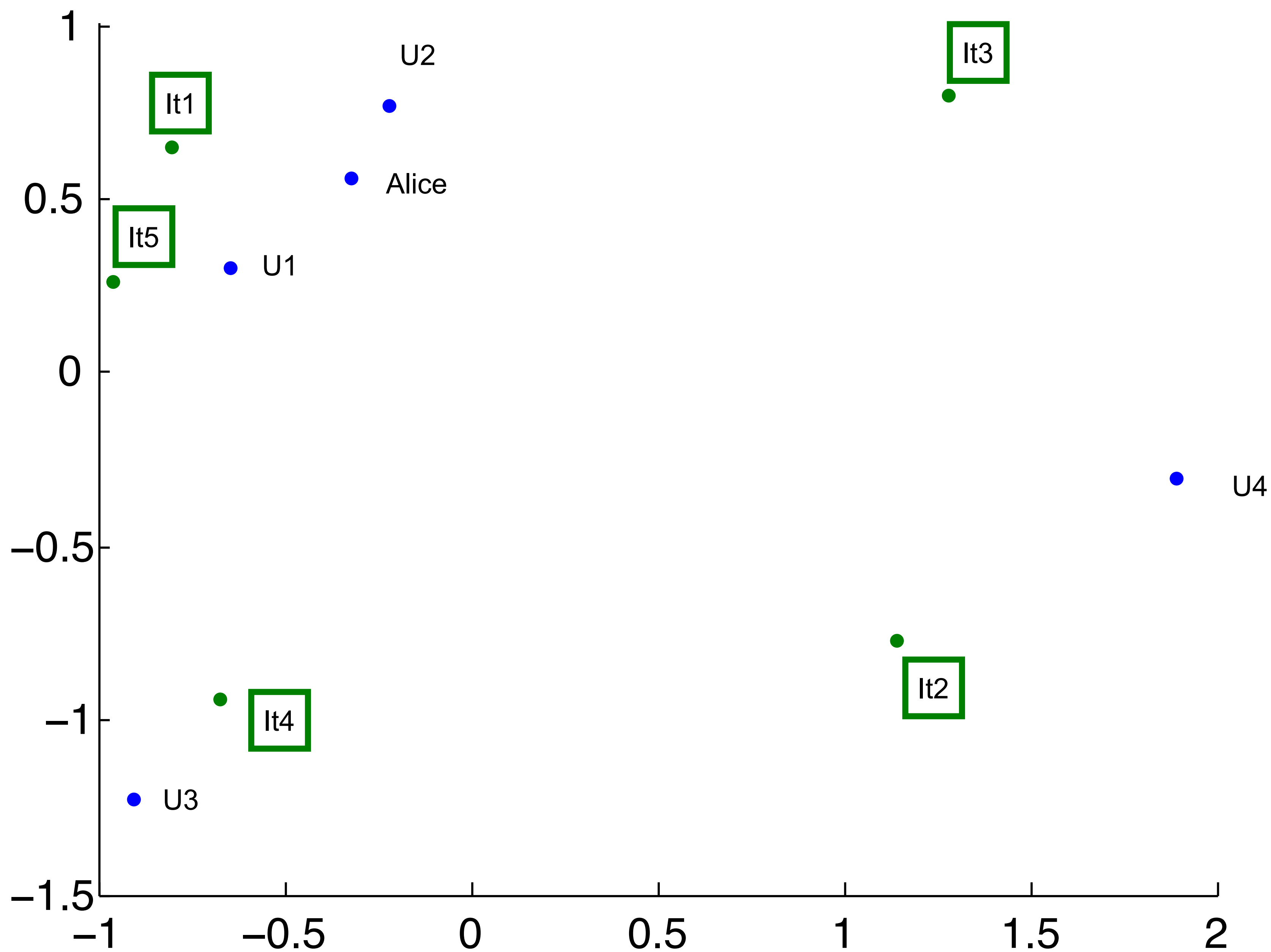
4,971	0,000
0,000	2,591
0,000	0,000

$Items_k$ =

-0,359	0,403
0,515	-0,478
0,575	0,496
-0,300	-0,581
-0,431	0,159

Matrix factorization

k=2



Matrix factorization

For k = 2;

M2 = U2*S2*(Items2)'

prediction2 = $(\mu) - b_u - b_i + M2(1,5) =$ 4.4602

mean_error= mean (mean(abs(M-M2))) = 0.1933

For k = 3;

M3 = U3*S3*(Items3)'

prediction3 = $(\mu) - b_u - b_i + M3(1,5) =$ 4.0356

mean_error= mean (mean(abs(M-M3))) = 0.0624

Matrix factorization

However,

svd needs full matrices.

Earlier works relied on imputation:

- increases enormously the amount of data to be handled
- data is distorted due to inaccurate imputations

Matrix factorization

To compute all estimators, Koren and Bell, set an optimization problem that admits an efficient solution and avoids the problem of missing values

$$\min_{b_*,q_*,p_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_i - b_u - q_i^T p_u)^2 + \lambda_4 (b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2)$$

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

q_i and p_u are vectors of k components inspired in rows and columns of the svd full matrix approach

Matrix factorization

The optimization problem can be solved

Alternating least squares technique rotate between

fixing the p_u's to solve for the q_i's, and

fixing the q_i's to solve for the p_u's

(each are quadratic problems that can be optimally solved)

Stochastic Gradient Descent

Matrix factorization

The optimization problem can be solved

Stochastic Gradient Descent

$$
\begin{aligned}
&\gamma = 0.005; \lambda_4 = 0.02;\\
&\textbf{for } r_{ui} \in \mathscr{K} \textbf{ do}\\
&\quad \hat{r}_{ui} = \mu + b_i = b_u + q_i^T p_u;\\
&\quad e_{ui} = r_{ui} - \hat{r}_{ui};\\
&\quad b_u \leftarrow b_u + \gamma \cdot (e_{ui} - \lambda_4 \cdot b_u);\\
&\quad b_i \leftarrow b_i + \gamma \cdot (e_{ui} - \lambda_4 \cdot b_i);\\
&\quad q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda_4 \cdot q_i);\\
&\quad p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda_4 \cdot p_u);\\
&\textbf{end for}
\end{aligned}
$$

Matrix factorization with implicit feedback

Considering the set R(u) of items that each user u has rated as an implicit feedback

---

$\gamma = 0.007; \lambda_5 = 0.005; \lambda_6 = 0.015;$
repeat
  for $r_{ui} \in \mathscr{K}$ **do**
    $\hat{r}_{ui} = \mu + b_i = b_u + q_i^T p_u;$
    $e_{ui} = r_{ui} - \hat{r}_{ui};$
    $b_u \leftarrow b_u + \gamma \cdot (e_{ui} - \lambda_5 \cdot b_u);$
    $b_i \leftarrow b_i + \gamma \cdot (e_{ui} - \lambda_5 \cdot b_i);$
    $q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot (p_u + |R(u)|^{-1/2} \sum_{j \in R(u)} y_j) - \lambda_6 \cdot q_i);$
    $p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda_6 \cdot p_u);$
    for $j \in R(u)$ **do**
      $j_j \leftarrow y_j + \gamma \cdot (e_{ui} \cdot |R(u)|^{-1/2} q_i - \lambda_6 \cdot y_j);$
    end for
  end for
  $\gamma = \gamma \cdot 0.9$
until convergence %around 30 iterations

Matrix factorization scores

SVD method, with improvements in temporal drift and implicit feedback increases its performance. The value of the rank k is also significant

	k=10	k=20	k=50	k=100	k=200
SVD	0,9140	0,9074	0,9046	0,9025	0,9009
SVD++	0,9131	0,9032	0,8952	0,8924	0,8911
times SVD++	0,8971	0,8891	0,8824	0,8805	0,8799

Matrix factorization scores

Finally, with some extra improvements in the algorithms to solve the optimization problems, the team of Koren and Bell won the Netflix Prize with a

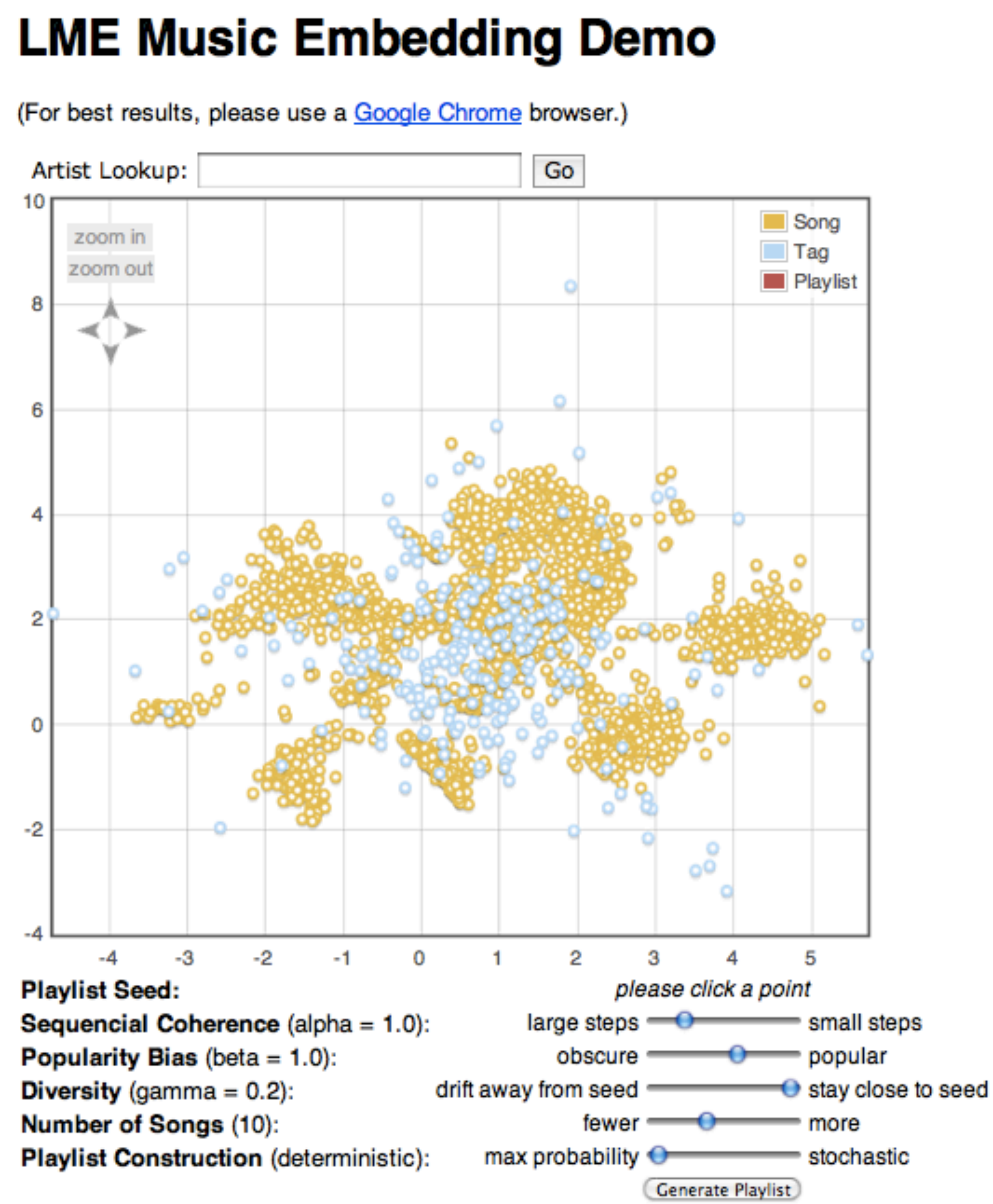
$$RMSE = 0.8567$$

Remember that the second team (The Ensemble) reached the same score. The victory was awarded to Koren and Bell since their results were submitted 20 minutes before.

Other Recommender Systems

- Playlists:

  Shuo Chen, Joshua Moore, Douglas Turnbull, Thorsten Joachims, *Playlist Prediction via Metric Embedding*, ACM Conference on Knowledge Discovery and Data Mining (KDD), 2012.



References

- Yehuda Koren, Robert M. Bell: Advances in Collaborative Filtering. Recommender Systems Handbook 2011: 145-186
- William W. Cohen: Collaborative Filtering: A Tutorial. Carnegie Mellon University
- Dietmar Jannach, Gerhard Friedrich: Tutorial: Recommender Systems. International Joint Conference on Artificial Intelligence (IJCAI). Barcelona, July 17, 2011. TU Dortmund, Alpen-Adria Universität Klagenfurt)