CS 175: Project in Artificial Intelligence

Slides 4: Collaborative Filtering

Topic 6: Collaborative Filtering

Some slides taken from Prof. Smyth

(with slight modifications)

Outline

- General aspects of recommender systems
- Nearest neighbor methods
- Matrix decomposition and singular value decomposition (SVD)

Recommender Systems

- Ratings or Vote data = m x n sparse binary matrix
  - n columns = “products”, e.g., books for purchase or movies for viewing
  - m rows = users
  - Interpretation:
    - Ratings: v(i,j) = user i’s rating of product j (e.g. on a scale of 1 to 5)
    - Purchases: v(i,j) = 1 if user i purchased product j
    - entry = 0 if no purchase or rating
- Automated recommender systems
  - Given ratings or votes by a user on a subset of items, recommend other items that the user may be interested in

Examples of Recommender Systems

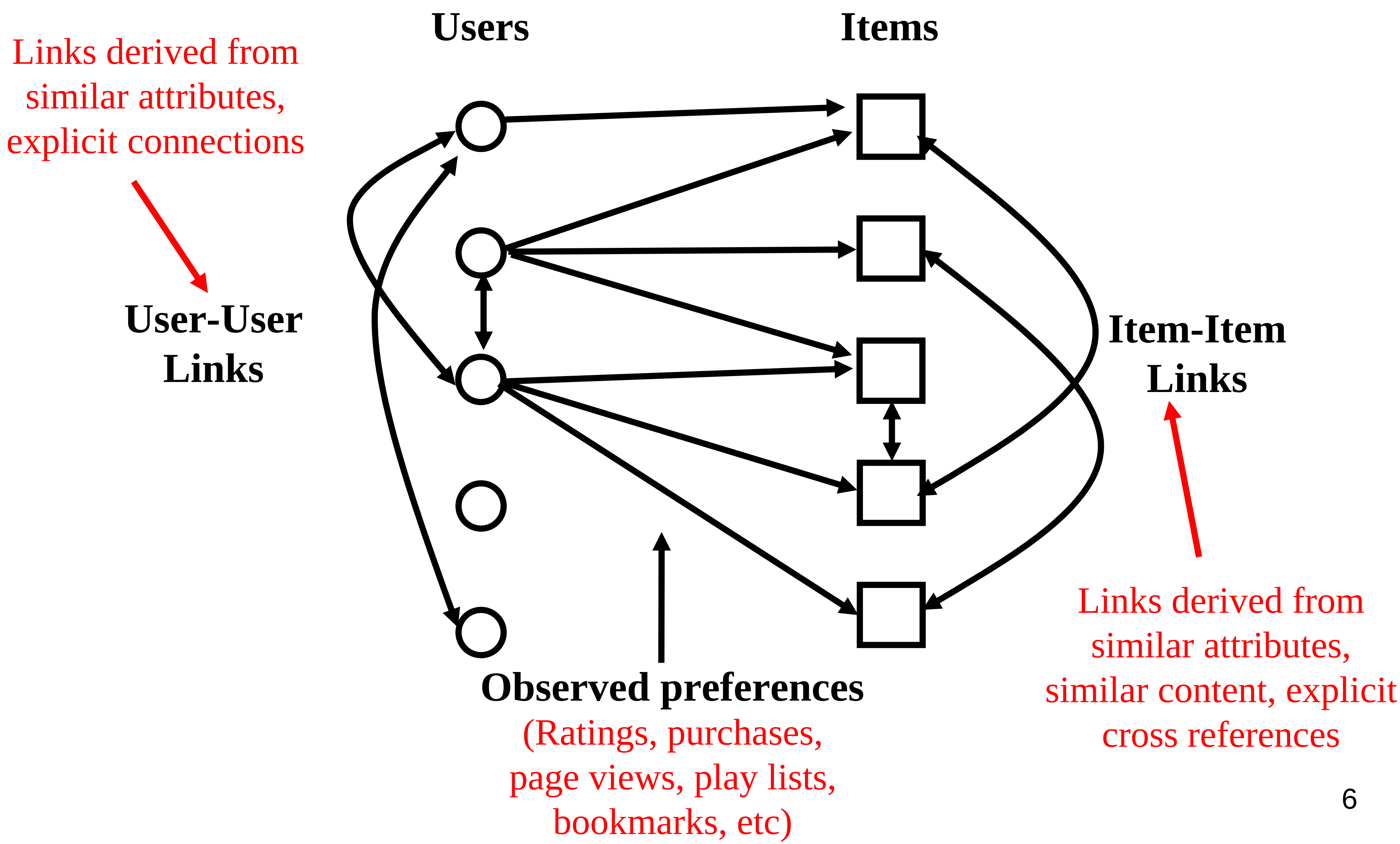
- Shopping
  - Amazon.com etc

- Movie and music recommendations:
  - Netflix
  - Last.fm

- Digital library recommendations
  - CiteSeer (Popescul et al, 2001):
    - m = 177,000 documents
    - N = 33,000 users
    - Each user accessed 18 documents on average (0.01% of the database -> very sparse!)

- Web page recommendations

The Recommender Space as a Bipartite Graph



Different types of recommender algorithms

- Nearest-neighbor/collaborative filtering algorithms
  - Widely used – simple and intuitive
- Matrix factorization (e.g., SVD)
  - Has gained popularity recent due to Netflix competition
- Less used
  - Neural networks
  - Cluster-based algorithms
  - Probabilistic models

Nearest-Neighbor Algorithms for Collaborative Filtering

$r_{i,k}$ = rating of user *i* on item *k*

I_i = items for which user *i* has generated a rating

Mean rating for user *i* is

$$\mu_i = \frac{1}{|I_i|} \sum_{j \in I_i} r_{i,j}$$

Predicted vote for user i on item j is a weighted sum

$$r_{i,j} = \mu_i + C \sum_{k=1}^{K} w_{i,k}(r_{k,j} - \mu_k)$$

Normalization constant
(e.g., total sum of weights)

weights of *K* similar users

Value of K can be optimized on a validation data set

Nearest-Neighbor Weighting

- K-nearest neighbor

$$w_{i,k} = 1 \text{ if } k \in \text{ neighbors}(i) \quad 0 \text{ otherwise}$$

- Pearson correlation coefficient (Resnick '94, Grouplens):

$$w_{i,k} = \rho_{i,k} = \frac{\sum_j (r_{k,j} - \mu_k)(r_{i,j} - \mu_i)}{\sqrt{\sum_j (r_{k,j} - \mu_k)^2 \sum_j (r_{i,j} - \mu_i)^2}}$$

Sums are over items rated by both users

- Can also scale weights by number of items in common

$$w'_{i,k} = \frac{n_{i,k}}{n_{i,k} + n_s} w_{i,k}$$

Smoothing constant, e.g., 10 or 100

Comments on Neighbor-based Methods

- Here we emphasized user-user similarity
  - Can also do this with item-item similarity, i.e.,
  - Find similar items (across users) to the item we need a rating for

- Simple and intuitive
  - Easy to provide the user with explanations of recommendations

- Computational Issues
    - In theory we need to calculate all n^2 pairwise weights
    - So scalability is an issue (e.g., real-time)
    - Significant engineering involved, many tricks

- For recent advances in neighbor-based approaches see

  Y. Koren, Factor in the neighbors: scalable and accurate collaborative filtering, ACM Transactions on Knowledge Discovery in Data, 2010

NOTES ON MATRIX DECOMPOSITION AND SVD

Matrix Decomposition

- Matrix D = m x n
  - e.g., Ratings matrix with m customers, n items
  - assume for simplicity that m > n

- Typically
  - R is sparse, e.g., less than 1% of entries have ratings
  - n is large, e.g., 18000 movies
  - So finding matches to less popular items will be difficult

Idea:

compress the columns (items) into a lower-dimensional representation

Singular Value Decomposition (SVD)

$$\underset{m \times n}{D} = \underset{m \times n}{U} \; \underset{n \times n}{\Sigma} \; \underset{n \times n}{V^t}$$

where: rows of V^t are eigenvectors of D^tD = basis functions

Σ is diagonal, with $\delta_{ii} = \mathrm{sqrt}(\lambda_i)$ (ith eigenvalue)

rows of U are coefficients for basis functions in V

(here we assumed that m > n, and rank D = n)

SVD Example

- Data [

10	20	10
2	5	2
8	17	7
9	20	10
12	22	11

SVD Example

- Data [

10	20	10
2	5	2
8	17	7
9	20	10
12	22	11

Note the pattern in the data above: the center column values are typically about twice the 1st and 3rd column values:

$\Rightarrow$ So there is redundancy in the columns, i.e., the column values are correlated

SVD Example

- Data [

10	20	10
2	5	2
8	17	7
9	20	10
12	22	11

D = U Σ V^t

where U =

0.50	0.14	-0.19
0.12	-0.35	0.07
0.41	-0.54	0.66
0.49	-0.35	-0.67
0.56	0.66	0.27

where Σ =

48.6	0	0
0	1.5	0
0	0	1.2

and V^t =

0.41	0.82	0.40
0.73	-0.56	0.41
0.55	0.12	-0.82

SVD Example

- Data [

10	20	10
2	5	2
8	17	7
9	20	10
12	22	11

D = U Σ V^t^

$$D = U \Sigma V^t$$

where U =

0.50	0.14	-0.19
0.12	-0.35	0.07
0.41	-0.54	0.66
0.49	-0.35	-0.67
0.56	0.66	0.27

where Σ =

$\boxed{48.6}$	0	0
0	1.5	0
0	0	1.2

Note that first singular value is much larger than the others

and V^t =

0.41	0.82	0.40
0.73	-0.56	0.41
0.55	0.12	-0.82

SVD Example

- Data [

$$\begin{matrix} 10 & 20 & 10 \\ 2 & 5 & 2 \\ 8 & 17 & 7 \\ 9 & 20 & 10 \\ 12 & 22 & 11 \end{matrix}$$

$D = U \Sigma V^t$

where $U = \begin{matrix} 0.50 & 0.14 & -0.19 \\ 0.12 & -0.35 & 0.07 \\ 0.41 & -0.54 & 0.66 \\ 0.49 & -0.35 & -0.67 \\ 0.56 & 0.66 & 0.27 \end{matrix}$

where $\Sigma = \begin{matrix} \boxed{48.6} & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1.2 \end{matrix}$

Note that first singular value
is much larger than the others

and $V^t = \begin{matrix} \boxed{0.41 \quad 0.82 \quad 0.40} \\ 0.73 \quad -0.56 \quad 0.41 \\ 0.55 \quad 0.12 \quad -0.82 \end{matrix}$

First basis function (or eigenvector)
carries most of the information and it "discovers"
the pattern of column dependence

Rows in D = weighted sums of basis vectors

1st row of D = [10 20 10]

Since $D = U\ S\ V^t$, then $D(1,:) = U(1,:) * \Sigma * V^t$

$= [24.5 \ \ 0.2 \ -0.22] * V^t$

$$V^t = \begin{matrix} 0.41 & 0.82 & 0.40 \\ 0.73 & -0.56 & 0.41 \\ 0.55 & 0.12 & -0.82 \end{matrix}$$

$\Rightarrow$ $D(1,:) = 24.5\ v_1 + 0.2\ v_2 + -0.22\ v_3$

where v_1, v_2, v_3 are rows of V^t and are our basis vectors

Thus, [24.5, 0.2, 0.22] are the weights that characterize row 1 in D

In general, the ith row of $U * \Sigma$ is the set of weights for the ith row in D

Summary of SVD Representation

$$D = U \Sigma V^t$$

Data matrix:

Rows = data vectors

$U^*\Sigma$ matrix:

Rows = weights for the rows of D

V^t matrix:

Rows = our basis functions

How do we compute U, Σ, and V?

- SVD decomposition is a standard eigenvector/value problem
  - The eigenvectors of D' D = the rows of V
  - The eigenvectors of D D' = the columns of U
  - The diagonal matrix elements in Σ are square roots of the eigenvalues of D' D

=> finding U,Σ,V is equivalent to finding eigenvectors of D'D

  - Solving eigenvalue problems is equivalent to solving a set of linear equations – time complexity is $O(m\, n^2 + n^3)$

In MATLAB, we can calculate this using the svd.m function, i.e.,

[u, s, v] = svd(D);

If matrix D is non-square, we can use svd(D,0)

Approximating the matrix D

- Example: we could approximate any row D just using a single weight

- Row 1:
  - Can be approximated by

$$D' = w_1 * v_1 = 24.5 * [\,0.41 \;\; 0.82 \;\; 0.40\,]$$
$$= [10.05 \;\; 20.09 \;\; 9.80]$$

  - D(1,:) = 10 20 10
  - Note that this is a close approximation of the exact D(1,:)

  (Similarly for any other row)

- Basis for data compression:
  - Sender and receiver agree on basis functions in advance
  - Sender then sends the receiver a small number of weights
  - Receiver then reconstructs the signal using the weights + the basis function
  - Results in far fewer bits being sent on average – trade-off is that there is some loss in the quality of the original signal

Matrix Approximation with SVD

$$D \approx U \quad \Sigma \quad V^t$$

$m \times n \qquad m \times f \quad f \times f \quad f \times n$

where: columns of V are first f eigenvectors of R^tR

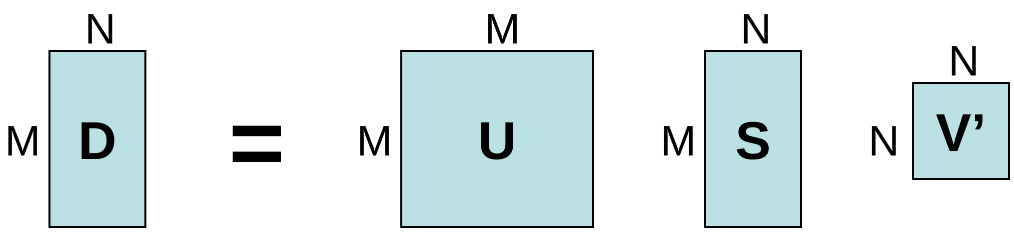
Σ is diagonal with f largest eigenvalues

rows of U are coefficients in reduced dimension V-space

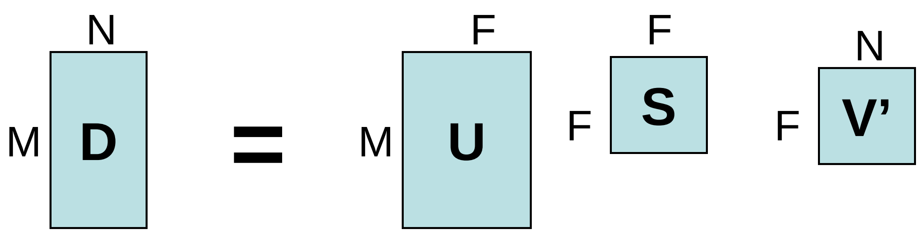
This approximation gives the best rank-f approximation to matrix R
in a least squares sense (this is also known as principal components analysis)

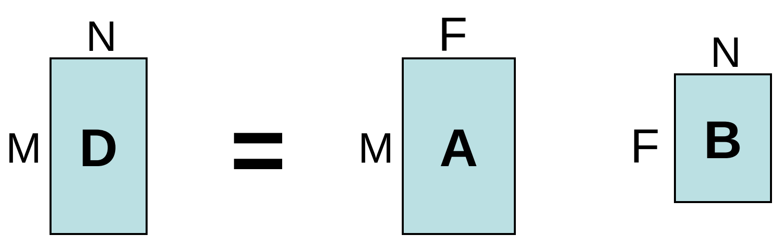
Singular Value Decomposition

- A matrix D can be decomposed: $D = USV'$



- Rank-f approximation:





Why do SVD?

- SVD provides the best f-rank approximation under the Frobenius Norm*:

$$F(D - AB) = \sum_{m=1}^{M} \sum_{n=1}^{N} (D_{mn} - (AB)_{mn})^2$$

- We often want to minimize (root) mean squared error for our ratings

* Benjamin Marlin. *Collaborative Filtering: A Machine Learning Perspective*. 2004.

Stochastic Gradient Descent

- Sometimes, matrix of ratings is too huge (i.e. Netflix is 480189 x 17770) to do full SVD
- Perform stochastic gradient descent to approximate A and B
  - Repeat until convergence:
    - Select one rating (D_{mn}) in our training set, randomly
    - Update row 'm' in A and column 'n' in B, based on update equations

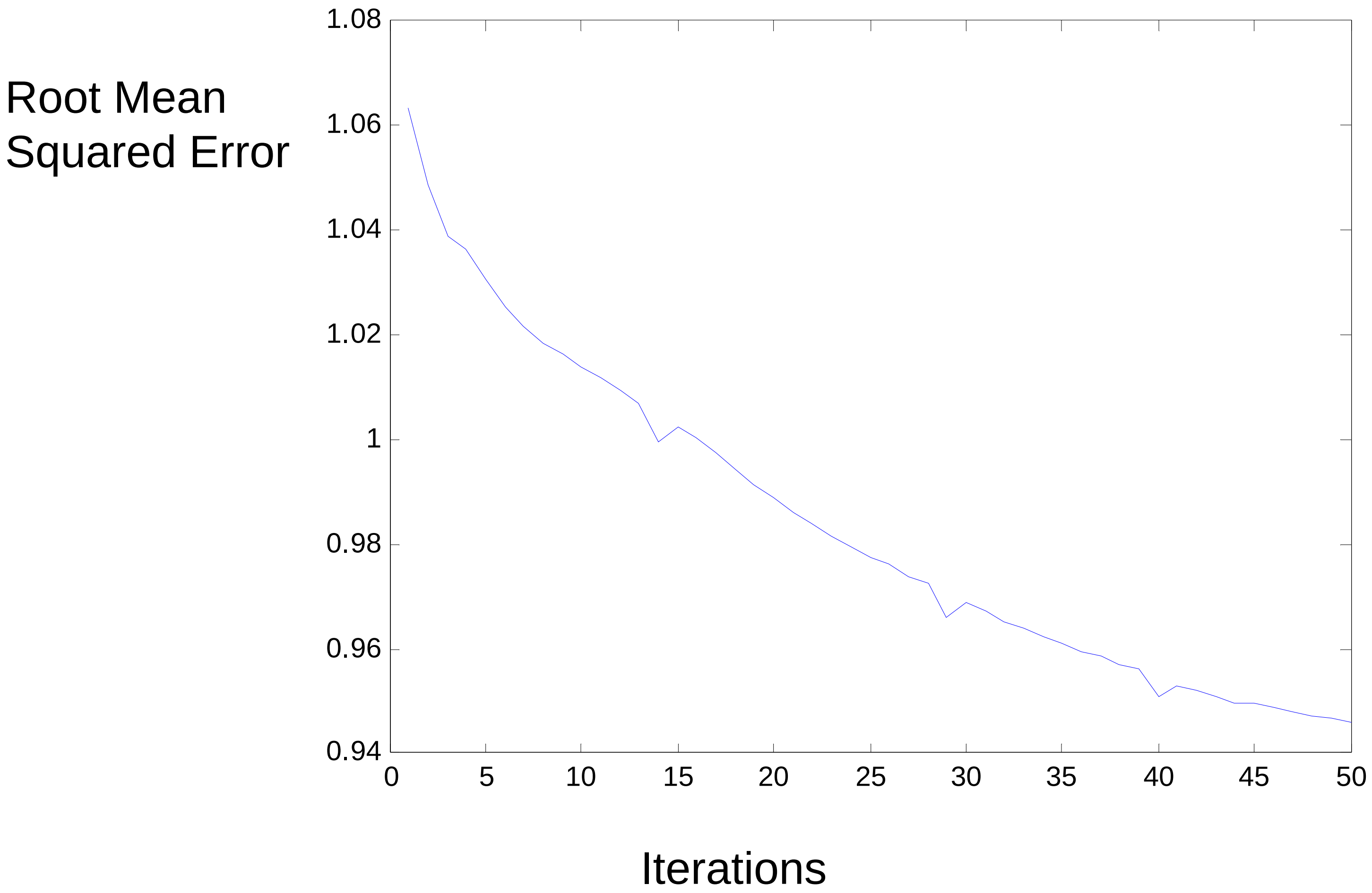
$$\frac{dF}{dA_{mf}} = -2(D_{mn} - (AB)_{mn})B_{fn}$$

$$\frac{dF}{dB_{fn}} = -2(D_{mn} - (AB)_{mn})A_{mf}$$

Exercise:
Work out these derivatives

- Can be done efficiently in Matlab, via vectorization
  - With f = 300, can do about 600,000 iterations per minute

Results of SVD on Netflix (f=600):



Example: Applying SVD to a Document-Term Matrix

	database	SQL	index	regression	likelihood	linear
d1	24	21	9	0	0	3
d2	32	10	5	0	3	0
d3	12	16	5	0	0	0
d4	6	7	2	0	0	0
d5	43	31	20	0	3	0
d6	2	0	0	18	7	16
d7	0	0	1	32	12	0
d8	3	0	0	22	4	2
d9	1	0	0	34	27	25
d10	6	0	0	17	4	23

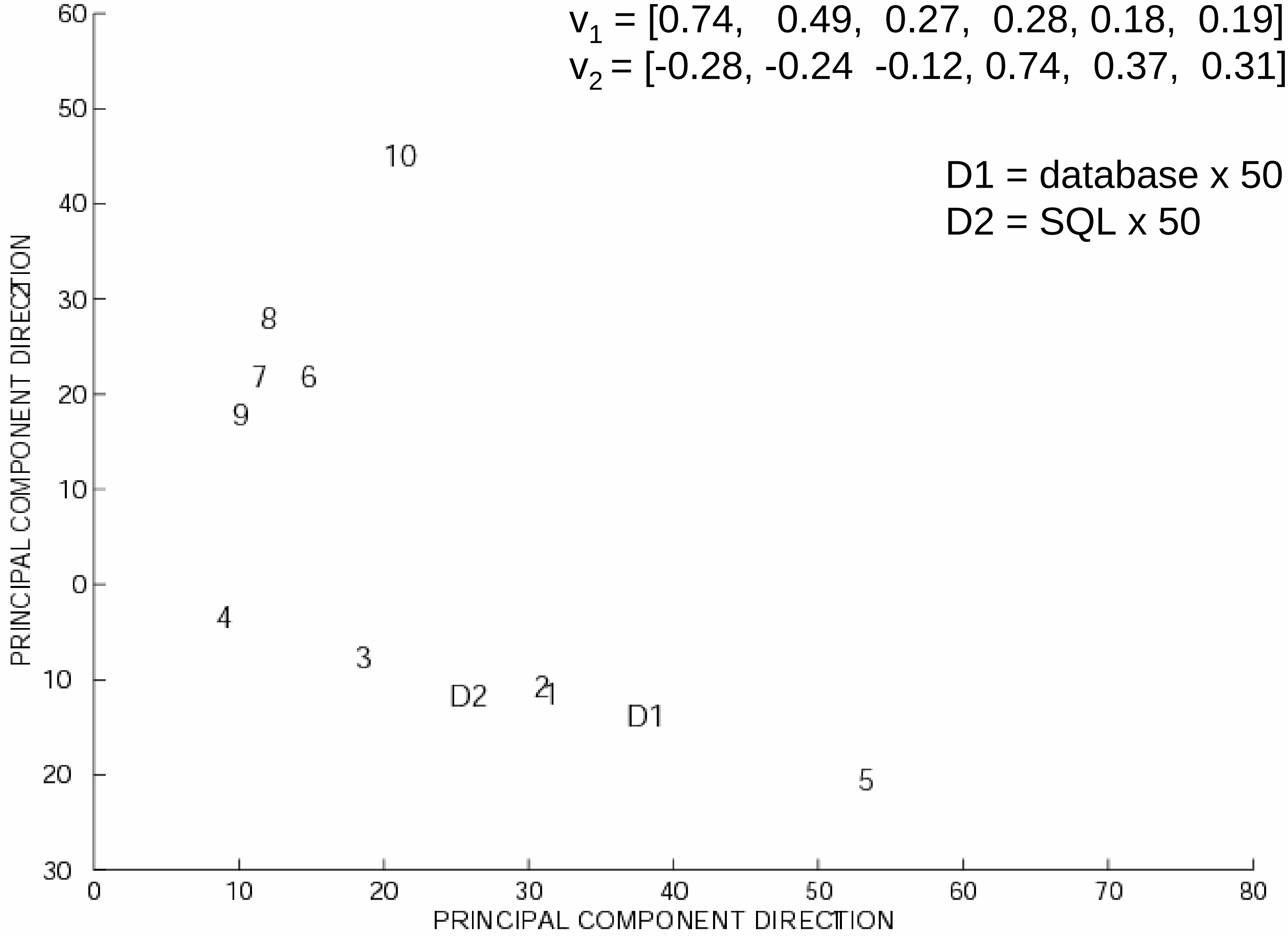
Results of SVD with 2 factors (f=2)

	database	SQL	index	regression	likelihood	linear
d1	24	21	9	0	0	3
d2	32	10	5	0	3	0
d3	12	16	5	0	0	0
d4	6	7	2	0	0	0
d5	43	31	20	0	3	0
d6	2	0	0	18	7	16
d7	0	0	1	32	12	0
d8	3	0	0	22	4	2
d9	1	0	0	34	27	25
d10	6	0	0	17	4	23

	U1	U2
d1	30.9	-11.5
d2	30.3	-10.8
d3	18.0	-7.7
d4	8.4	-3.6
d5	52.7	-20.6
d6	14.2	21.8
d7	10.8	21.9
d8	11.5	28.0
d9	9.5	17.8
d10	19.9	45.0

v_1 = [0.74, 0.49, 0.27, 0.28, 0.18, 0.19]
v_2 = [-0.28, -0.24 -0.12, 0.74, 0.37, 0.31]

D1 = database x 50
D2 = SQL x 50



Latent Semantic Indexing

- LSI = application of SVD to document-term data
- Querying
  - Project documents into f-dimensional space
  - Project each query q into f-dimensional space
  - Find documents closest to query q in f-dimensional space
  - Often works better than matching in original high-dimensional space
- Why is this useful?
  - Query contains “automobile”, document contains “vehicle”
  - can still match Q to the document since the 2 terms will be close in k-space (but not in original space), i.e., addresses synonymy problem

Related Ideas

- Topic Modeling
  - Can also be viewed as matrix factorization
    - Basis functions = topics
  - Topics tend to be more interpretable than LSI vectors (better suited to non-negative matrices)
  - May also perform better for document retrieval

- Non-negative Matrix Factorization

NETFLIX: CASE STUDY

Netflix

- Movie rentals by DVD (mail) and online (streaming)
- 100k movies, 10 million customers
- Ships 1.9 million disks to customers each day
  - 50 warehouses in the US
  - Complex logistics problem
- Employees: 2000
  - But relatively few in engineering/software
  - And only a few people working on recommender systems
- Moving towards online delivery of content
- Significant interaction of customers with Web site

The $1 Million Question

Million Dollars Awarded Sept 21st 2009

Ratings Data

17,700 movies

480,000 users

1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

Scoring

Minimize root mean square error (RMSE)

Mean square error = $1/|R| \sum_{(u,i) \varepsilon R} (r_{ui} - \hat{r}_{ui})^2$

Does not necessarily correlate well with user satisfaction

But is a widely-used well-understood quantitative measure

RMSE Baseline Scores on Test Data

1.054 - just predict the mean user rating for each movie

0.953 - Netflix’s own system (Cinematch) as of 2006

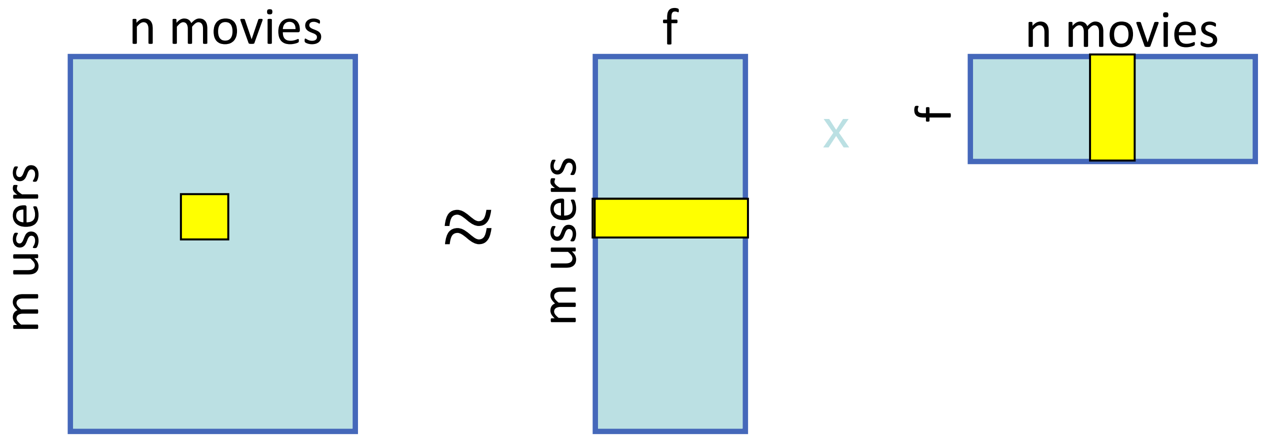
0.941 - nearest-neighbor method using correlation

0.857 - required 10% reduction to win $1 million

Why did Netflix do this?

- Customer satisfaction/retention is key to Netflix – they would really like to improve their recommender systems
- Progress with internal system (Cinematch) was slow
- Initial prize idea from CEO Reed Hastings
- $1 million would likely easily pay for itself
- Potential downsides
  - Negative publicity (e.g., privacy)
  - No-one wins the prize (conspiracy theory)
  - The prize is won within a day or 2
  - Person-hours at Netflix to run the competition
  - Algorithmic solutions are not useful operationally

Matrix Factorization of Ratings Data



$$r_{ui} \approx a^t_u \ b_i$$

$$r_{ui} \approx a^t_i \ b_u$$

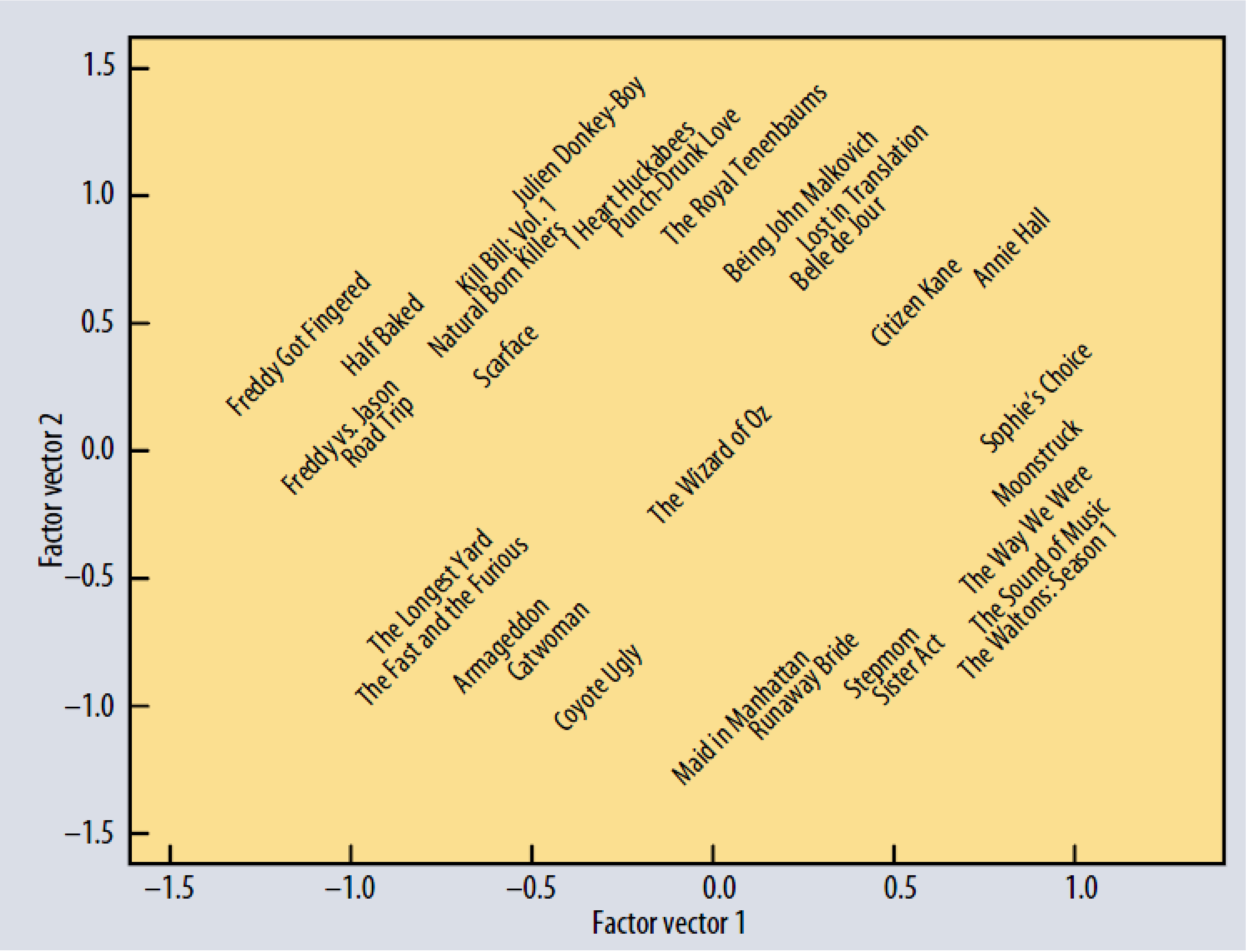


Figure from Koren, Bell, Volinksy, IEEE Computer, 2009

Dealing with Missing Data

$$r_{ui} \approx a^t_i \; b_u$$

$$\min_{a,b} \sum_{(u,i)\,\varepsilon\,R} (r_{ui} - a^t_i \; b_u)^2$$

sum is only over known ratings

Dealing with Missing Data

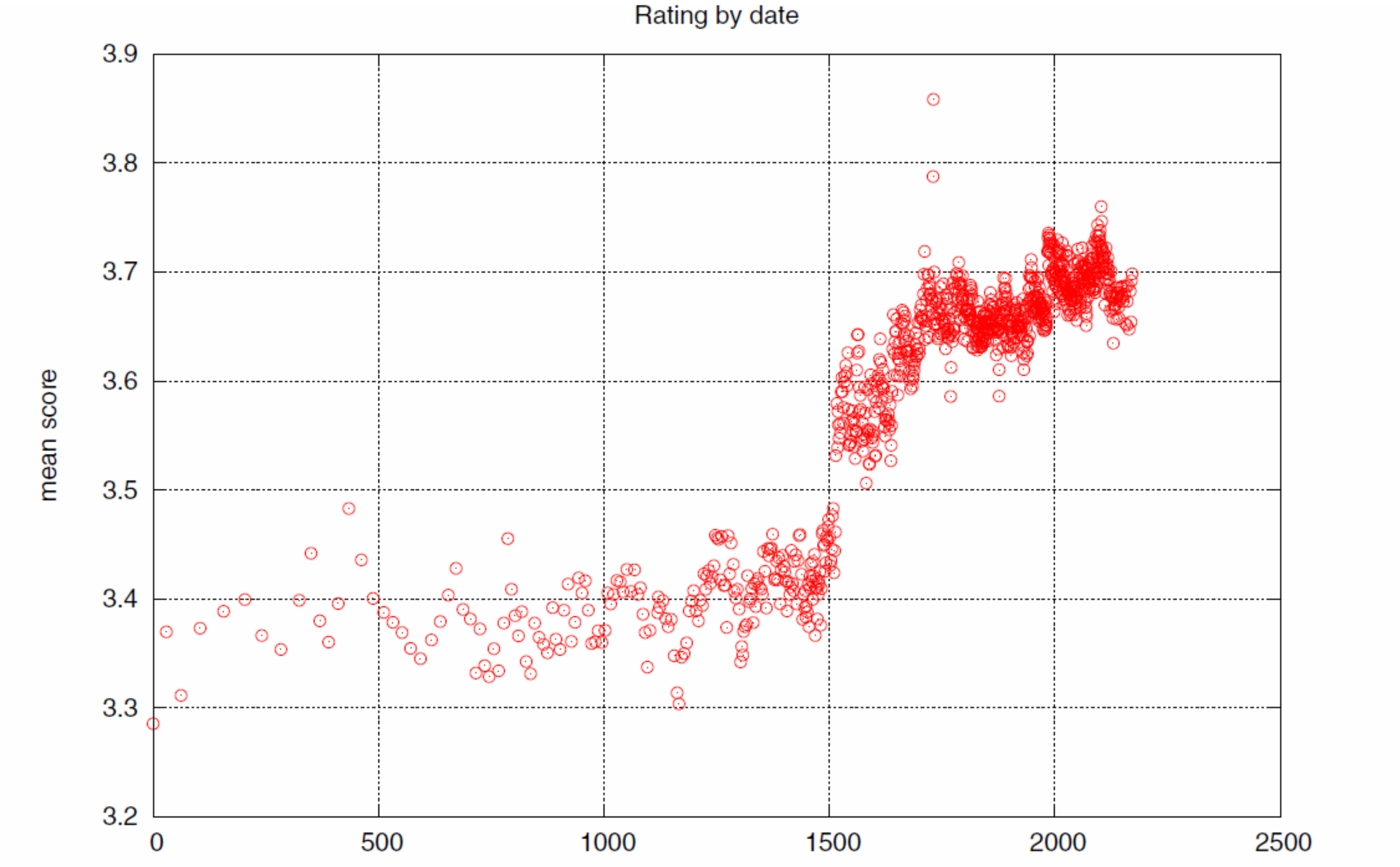
$$r_{ui} \approx a^t_i \; b_u$$

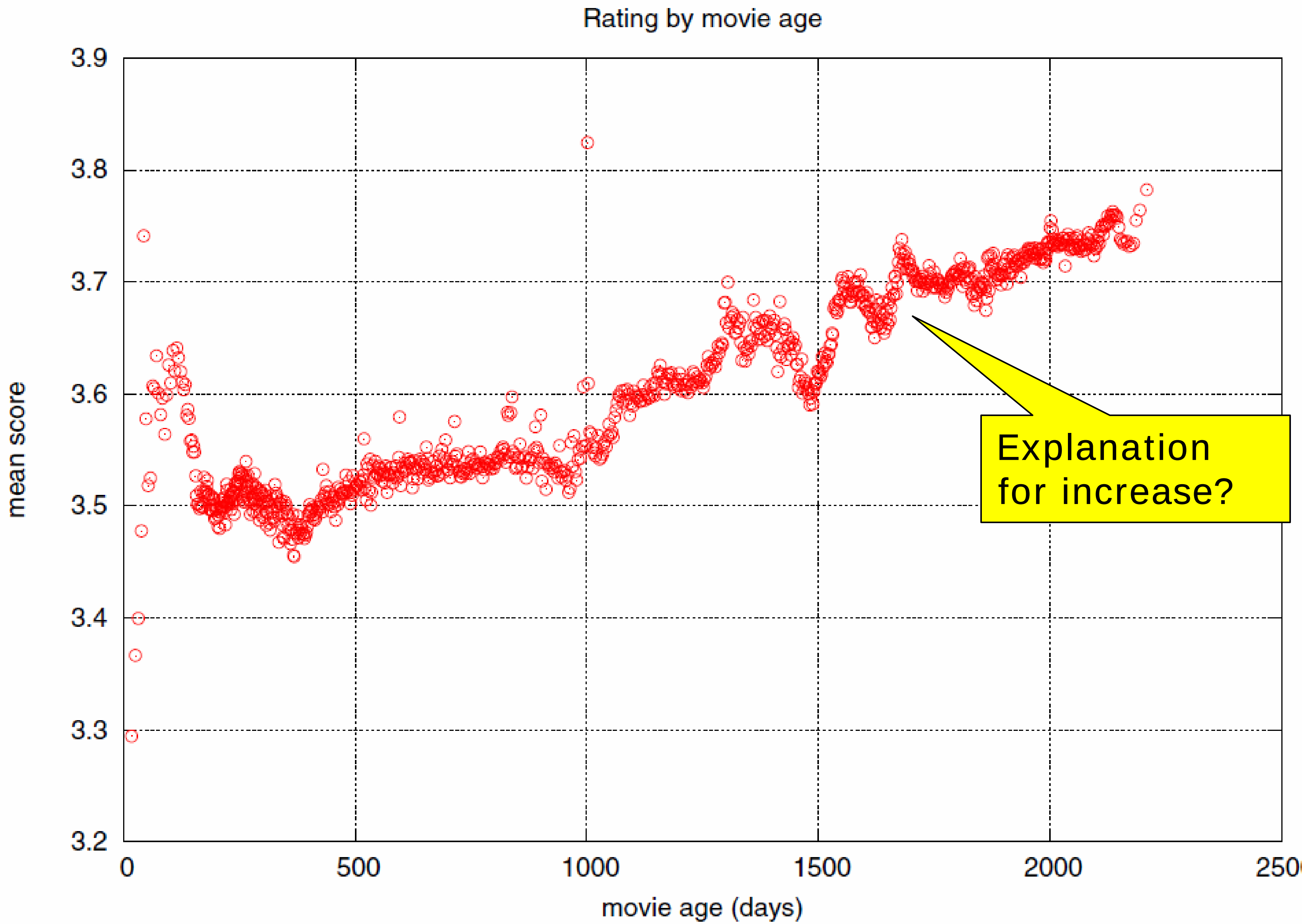
$$\min_{a,b} \sum_{(u,i)\, \varepsilon\, R} (r_{ui} - a^t_i \; b_u)^2$$

Add regularization

$$\min_{a,b} \sum_{(u,i)\, \varepsilon\, R} (r_{ui} - a^t_i \; b_u)^2 + \lambda \; (|a_i|^2 + |b_u|^2)$$

Time effects also important





Rating by movie age
3.9
3.8
3.7
3.6
3.5
3.4
3.3
3.2
mean score
0
500
1000
1500
2000
2500
movie age (days)
Explanation for increase?

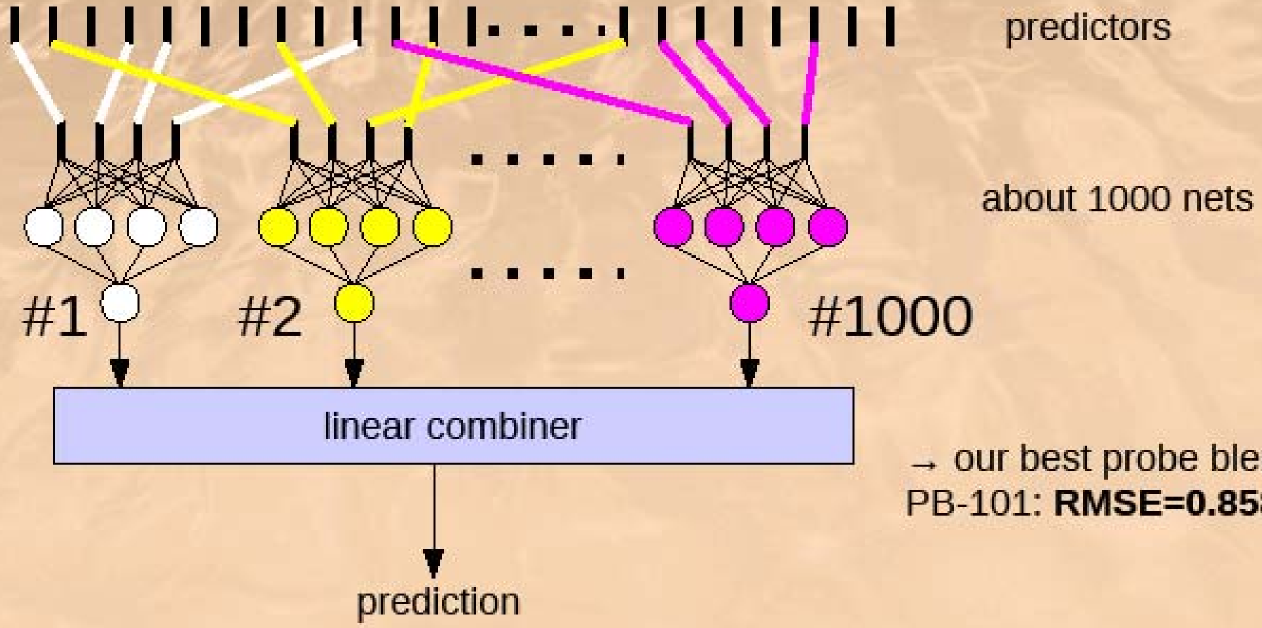
The Kitchen Sink Approach….

- Many options for modeling
  - Variants of the ideas we have seen so far
    - Different ways to model time
    - Different ways to handle implicit information
    - Different numbers of factors
    - ….
  - Other models
    - Nearest-neighbor models
    - Restricted Boltzmann machines

- Model averaging was useful….
  - Linear model combining
  - Neural network combining
  - Gradient boosted decision tree combining
  - Note: combining weights learned on validation set (“stacking”)

Ensemble NNBlend

- Train many small NN's (>1000) on a random subset
  - Per net: 20..40 weights
- Combine them linearly



→ our best probe blend:
PB-101: **RMSE=0.8584**

Other Aspects of Model Building

- Automated parameter tuning
  - Using a validation set, and grid search, various parameters such as learning rates, regularization parameters, etc., can be optimized

- Memory requirements
  - Memory: can fit within roughly 1 Gbyte of RAM

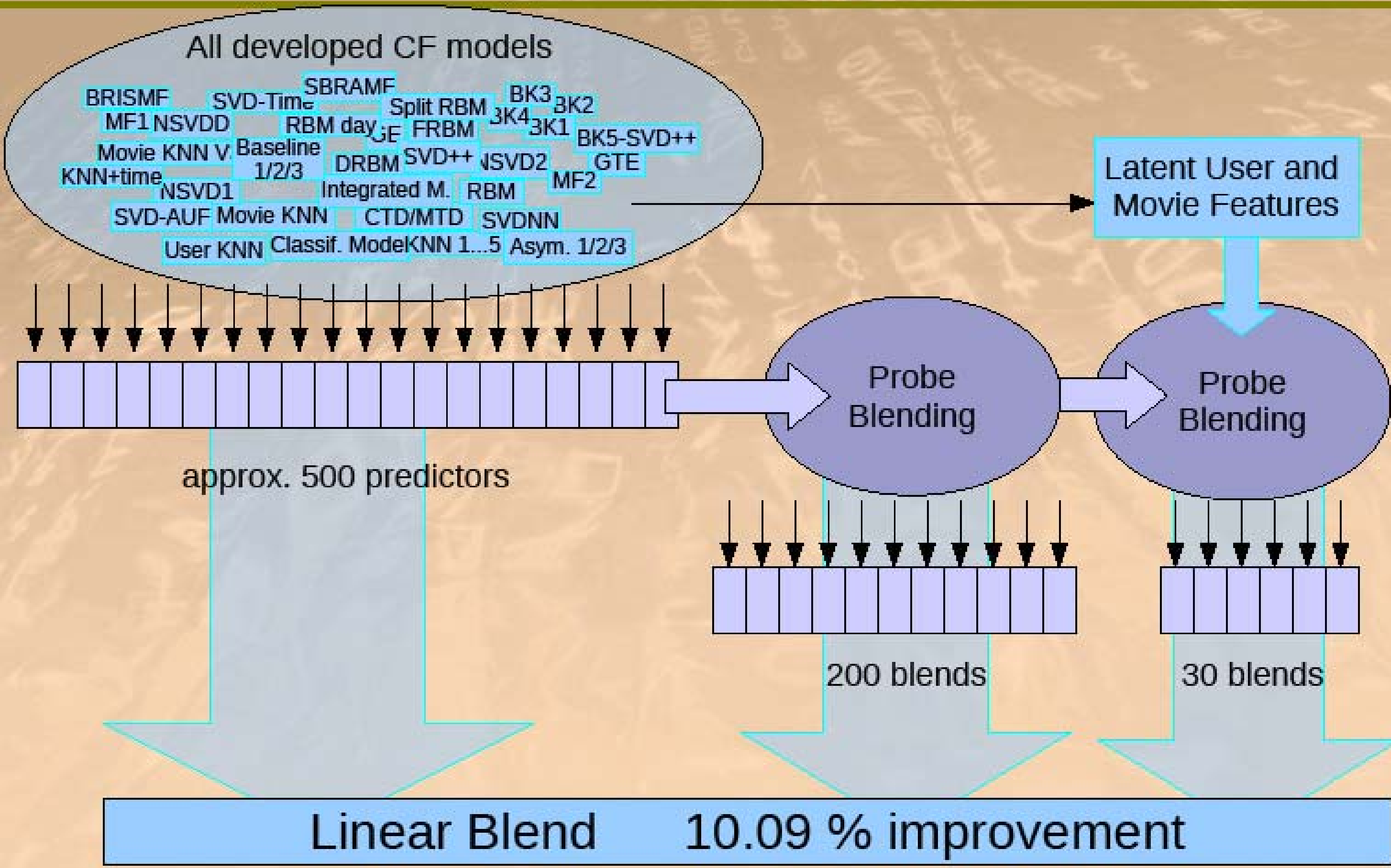
- Training time
  - Order of days: but achievable on commodity hardware rather than a supercomputer
  - Some parallelization used

Matrix factorization vs Near Neighbor?

From Koren, ACM Transactions on Knowledge Discovery, 2010

“Latent factor models such as SVD face real difficulties when needed to explain predictions. …Thus, we believe that for practical applications neighborhood models are still expected to be a common choice.”

The big picture
Solution of BellKor's Pragmatic Chaos



June 26th 2009: after 1000 Days and nights…

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	xlvector	0.8639	9.20	2009-06-26 13:49:04
13	xiangliang	0.8639	9.20	2009-06-26 07:47:34
14	Feeds2	0.8641	9.18	2009-06-26 22:51:55
15	Ces	0.8642	9.17	2009-06-24 14:34:14

The Leading Team

- BellKorPragmaticChaos
  - BellKor:
    - Yehuda Koren (now Yahoo!), Bob Bell, Chris Volinsky, AT&T
  - BigChaos:
    - Michael Jahrer, Andreas Toscher, 2 grad students from Austria
  - Pragmatic Theory
    - Martin Chabert, Martin Piotte, 2 engineers from Montreal (Quebec)

- June 26th submission triggers 30-day “last call”

- Submission timed purposely to coincide with vacation schedules

The Last 30 Days

- Ensemble team formed
  - Group of other teams on leaderboard forms a new team
  - Relies on combining their models
  - Quickly also get a qualifying score over 10%

- BellKor
  - Continue to eke out small improvements in their scores
  - Realize that they are in direct competition with Ensemble

- Strategy
  - Both teams carefully monitoring the leaderboard
  - Only sure way to check for improvement is to submit a set of predictions
    - This alerts the other team of your latest score

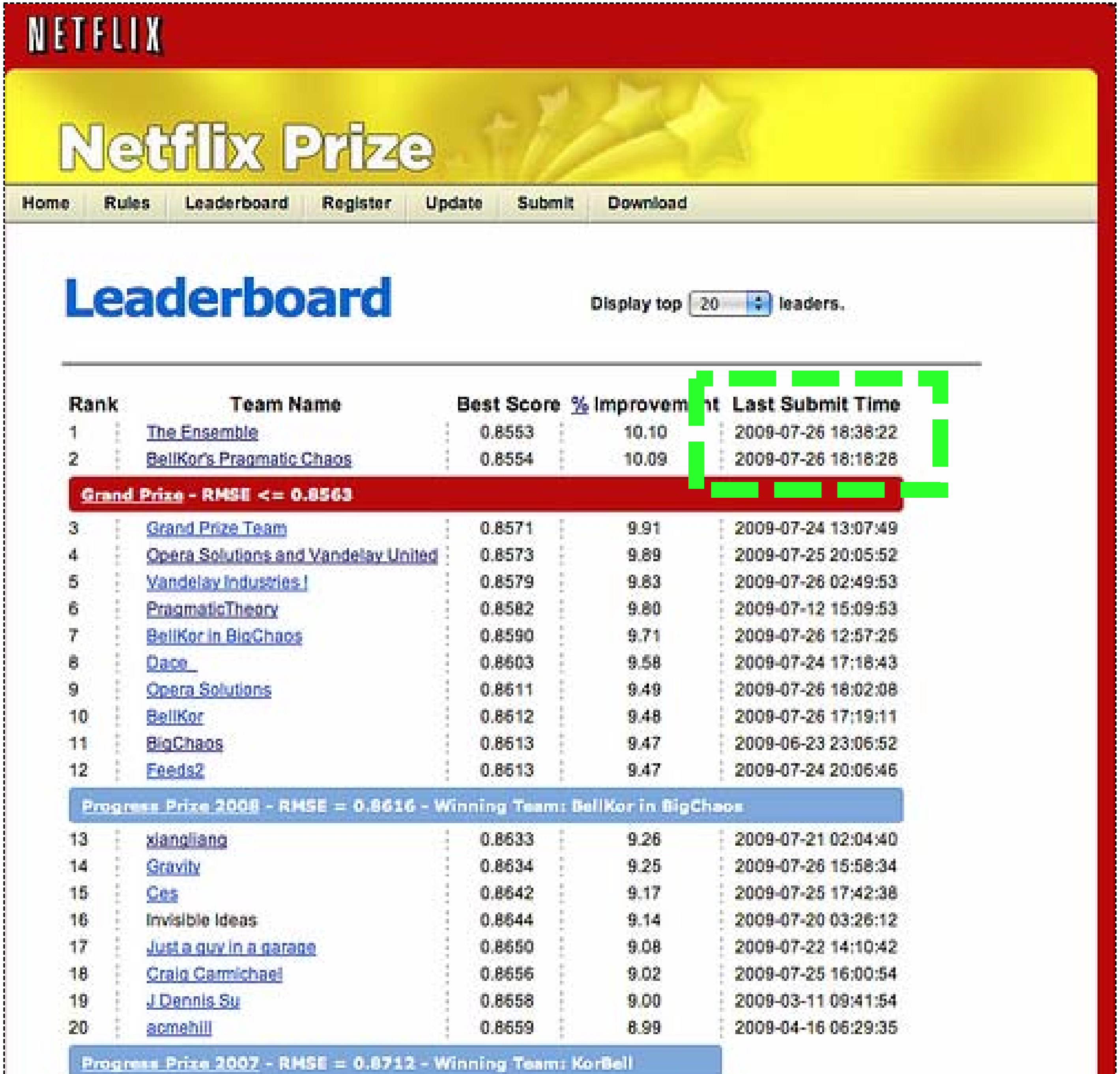
24 Hours from the Deadline

- Submissions limited to 1 a day
  - So only 1 final submission could be made by either in the last 24 hours

  - team 24 hours before deadline…
  - BellKor team member in Austria notices (by chance) that Ensemble posts a score that is slightly better than BellKor's
  - Leaderboard score disappears after a few minutes (rule loophole)

- Frantic last 24 hours for both teams
  - Much computer time on final optimization
  - run times carefully calibrated to end about an hour before deadline

- Final submissions
  - BellKor submits a little early (on purpose), 40 mins before deadline
  - Ensemble submits their final entry 20 mins later
  - ….and everyone waits….


NETFLIX

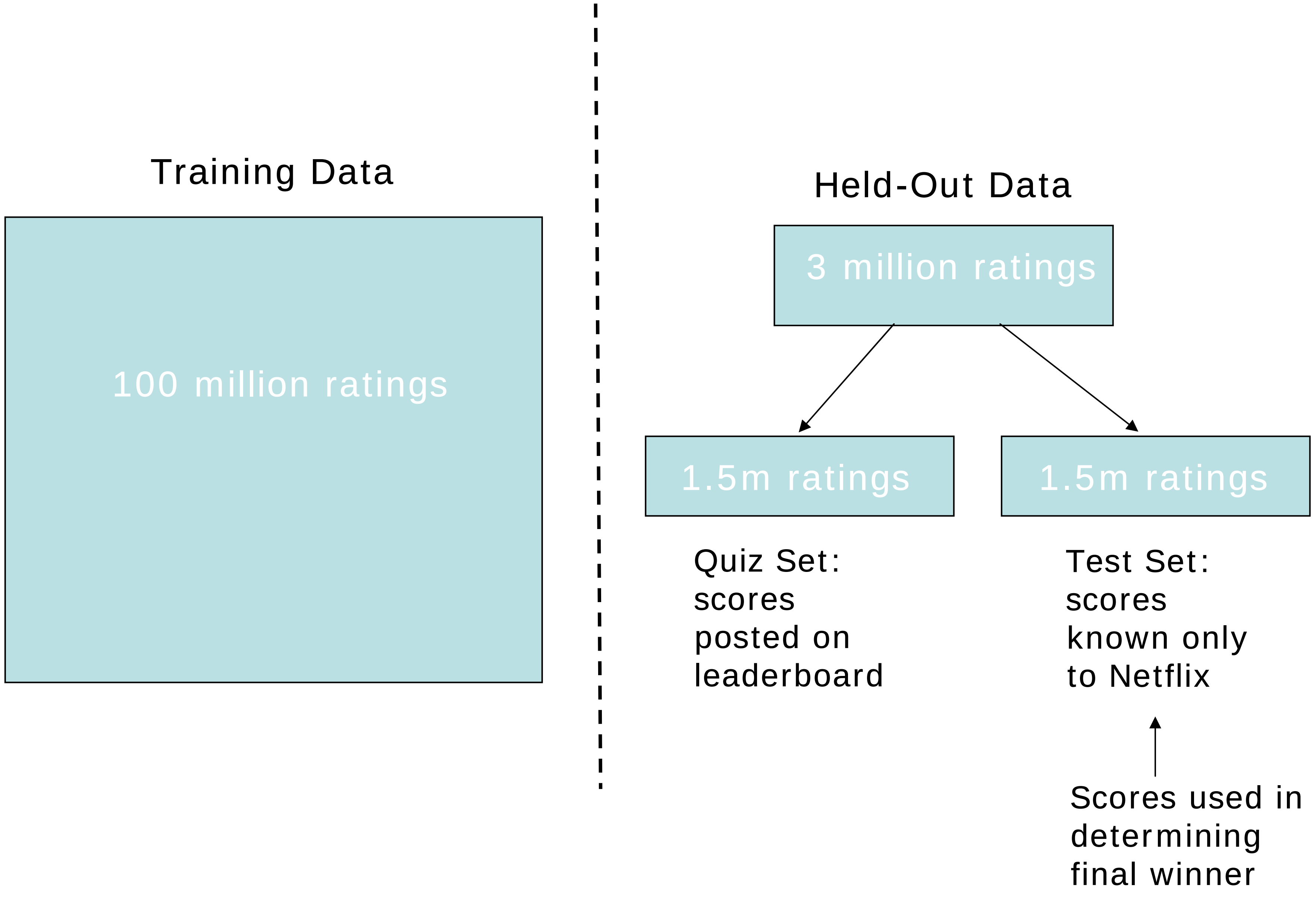
Netflix Prize

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

Leaderboard

Display top 20 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	The Ensemble	0.8553	10.10	2009-07-26 18:38:22
2	BellKor's Pragmatic Chaos	0.8554	10.09	2009-07-26 18:18:28
Grand Prize - RMSE <= 0.8563				
3	Grand Prize Team	0.8571	9.91	2009-07-24 13:07:49
4	Opera Solutions and Vandelay United	0.8573	9.89	2009-07-25 20:05:52
5	Vandelay Industries !	0.8579	9.83	2009-07-26 02:49:53
6	PragmaticTheory	0.8582	9.80	2009-07-12 15:09:53
7	BellKor in BigChaos	0.8590	9.71	2009-07-26 12:57:25
8	Dace_	0.8603	9.58	2009-07-24 17:18:43
9	Opera Solutions	0.8611	9.49	2009-07-26 18:02:08
10	BellKor	0.8612	9.48	2009-07-26 17:19:11
11	BigChaos	0.8613	9.47	2009-06-23 23:06:52
12	Feeds2	0.8613	9.47	2009-07-24 20:06:46
Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8633	9.26	2009-07-21 02:04:40
14	Gravity	0.8634	9.25	2009-07-26 15:58:34
15	Ces	0.8642	9.17	2009-07-25 17:42:38
16	Invisible Ideas	0.8644	9.14	2009-07-20 03:26:12
17	Just a guy in a garage	0.8650	9.08	2009-07-22 14:10:42
18	Craig Carmichael	0.8656	9.02	2009-07-25 16:00:54
19	J Dennis Su	0.8658	9.00	2009-03-11 09:41:54
20	acmehill	0.8659	8.99	2009-04-16 06:29:35
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell				


Training Data
100 million ratings
Held-Out Data
3 million ratings
1.5m ratings
1.5m ratings
Quiz Set:
scores
posted on
leaderboard
Test Set:
scores
known only
to Netflix
Scores used in
determining
final winner

NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update | Download

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	[illegible]	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8642	9.27	2009-07-15 14:53:22
14	Gravity	0.8643	9.26	2009-04-22 18:31:32
15	Ces	0.8651	9.18	2009-06-21 19:24:53
16	Invisible Ideas	0.8653	9.15	2009-07-15 15:53:04
17	Just a guy in a garage	0.8662	9.06	2009-05-24 10:02:54
18	J Dennis Su	0.8666	9.02	2009-03-07 17:16:17
19	Craig Carmichael	0.8666	9.02	2009-07-25 16:00:54
20	acmehill	0.8668	9.00	2009-03-21 16:20:50
Progress Prize 2007 - RMSE = 0.8723 - Winning Team: KorBell				

NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update | Download

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	[illegible]	[illegible]	[illegible]
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8642	9.27	2009-07-15 14:53:22
14	Gravity	0.8643	9.26	2009-04-22 18:31:32
15	Ces	0.8651	9.18	2009-06-21 19:24:53
16	Invisible Ideas	0.8653	9.15	2009-07-15 15:53:04
17	Just a guy in a garage	0.8662	9.06	2009-05-24 10:02:54
18	J Dennis Su	0.8666	9.02	2009-03-07 17:16:17
19	Craig Carmichael	0.8666	9.02	2009-07-25 16:00:54
20	acmehill	0.8668	9.00	2009-03-21 16:20:50
Progress Prize 2007 - RMSE = 0.8723 - Winning Team: KorBell				

Million Dollars Awarded Sept 21^{st} 2009

Lessons Learned

- Scalability is important
  - e.g., stochastic gradient descent on sparse matrices
- Latent factor models work well on this problem
  - Previously had not been explored for recommender systems
- Understanding your data is important, e.g., time-effects
- Combining models works surprisingly well
  - But final 10% improvement can probably be achieved by judiciously combining about 10 models rather than 1000's
  - This is likely what Netflix will do in practice
- Surprising amount of collaboration among participants

The New York Times

Netflix Competitors Learn the Power of Teamwork

By STEVE LOHR
Published: July 27, 2009

A contest set up by Netflix, which offered a $1 million prize to anyone who could significantly improve its movie recommendation system, ended on Sunday with two teams in a virtual dead heat, and no winner to be declared until September.

Enlarge This Image

Ozier Muhammad/The New York Times

Chris Volinsky, a scientist at AT&T Research, left, is on a high-ranking team in a Netflix contest. With him is Robert Bell.

Related

The Screens Issue: If You Liked This, You're Sure to Love That (November 23, 2008)

Times Topics: Netflix Inc.

But the contest, which began in October 2006, has already produced an impressive legacy. It has shaped careers, spawned at least one start-up company and inspired research papers. It has also changed conventional wisdom about the best way to build the automated systems that increasingly help people make online choices about movies, books, clothing, restaurants, news and other goods and services.

These so-called recommendation engines are computing models that predict what a person might enjoy based on statistical scoring of that person's stated preferences, past consumption patterns and similar choices made by many others — all made possible by the ease of data collection and tracking on the Web.

Why Collaboration?

Openness of competition structure

- Rules stated that winning solutions would be published
  - Non-exclusive license of winning software to Netflix
  - “Description of algorithm to be posted on site”
- Research workshops sponsored by Netflix
- Leaderboard was publicly visible: “it was addictive….”

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update | Download

Netflix Prize: Forum

Forum for discussion about the Netflix Prize and dataset.

Index | User list | Rules | Search | Register | Login

You are not logged in.

Announcement

Congratulations to team "BellKor's Pragmatic Chaos" for being awarded the $1M Grand Prize on September 21, 2009. Stay tuned for details of the next contest, Netflix Prize 2.

Administrivia

Forum	Topics	Posts	Last post
Important Announcements	5	151	Today 04:29:38 by YehudaKoren
Registration Problems	1	1	2006-10-05 08:37:53 by prizemaster
Administrivia Administrative notes from the maintainers	3	43	2009-06-22 09:23:04 by dale5351
Prize and Forum FAQ	15	18	2009-03-24 10:18:36 by prizemaster
Request for new Category or Forum Want to add a new high-level Category or Forum? This is the place to ask or comment.	18	40	2008-04-29 20:50:19 by filmmakershelp

Awarded Prizes

Forum	Topics	Posts	Last post
Grand Prize	1	14	2009-10-09 12:18:23 by statistician
Progress Prize 2008	2	17	2009-03-18 02:40:53 by CS1
Progress Prize 2007	5	29	2008-10-06 06:51:51 by dinc3r

Questions (and answers)

Forum	Topics	Posts	Last post

Why Collaboration?

Development of Online Community

- Active Netflix prize forum + other blogs
- Quickly acquired “buzz”
- Forum was well-moderated by Netflix
- Attracted discussion from novices and experts alike
- Early posting of code and solutions
- Early self-identification (links via leaderboard)

Why Collaboration?

Academic/Research Culture

- Nature of competition was technical/mathematical
- Attracted students, hobbyists, researchers
- Many motivated by fundamental interest in producing better algorithms - $1 million would be a nice bonus
- History in academic circles of being open, publishing, sharing

Questions

- Does reduction in squared error metric correlate with real improvements in user satisfaction?

- Are these competitions good for scientific research?
  - Should researchers be solving other more important problems?

- Are competitions a good strategy for companies?

Evaluation Methods

- Research papers use historical data to evaluate and compare different recommender algorithms
  - predictions typically made on items whose ratings are known
  - e.g., leave-1-out method,
    - each positive vote for each user in a test data set is in turn “left out”
    - predictions on left-out items made given rated items
  - e.g., predict-given-k method
    - Make predictions on rated items given k=1, k=5, k=20 ratings
  - See Herlocker et al (2004) for detailed discussion of evaluation

- Approach 1: measure quality of rankings
    - Score = weighted sum of true votes in top 10 predicted items

- Approach 2: directly measure prediction accuracy
    - Mean-absolute-error (MAE) between predictions and actual votes
    - Typical MAE on large data sets ~ 20% (normalized)
      - E.g., on a 5-point scale predictions are within 1 point on average

Evaluation with (Implicit) Binary Purchase Data

- Cautionary note:
  - It is not clear that prediction on historical data is a meaningful way to evaluate recommender algorithms, especially for purchasing
  - Consider:
    - User purchases products A, B, C
    - Algorithm ranks C highly given A and B, gets a good score
    - However, what if the user would have purchased C anyway, i.e., making this recommendation would have had no impact? (or possibly a negative impact!)
  - What we would really like to do is reward recommender algorithms that lead the user to purchase products that they would not have purchased without the recommendation
    - This can't be done based on historical data alone
  - Requires direct "live" experiments (which is often how companies evaluate recommender algorithms)