Recommendation Systems: k-NN, SVD & Netflix challenge

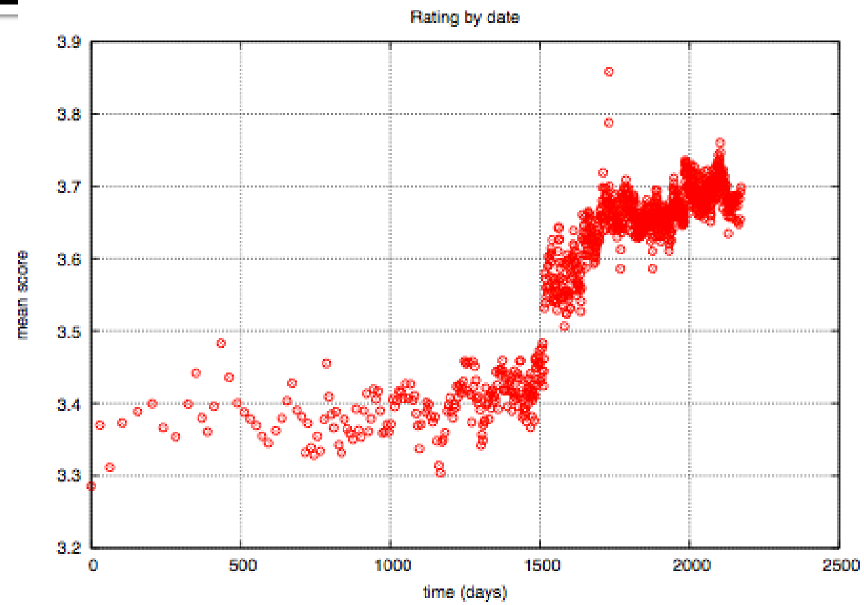
CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
http://cs246.stanford.edu

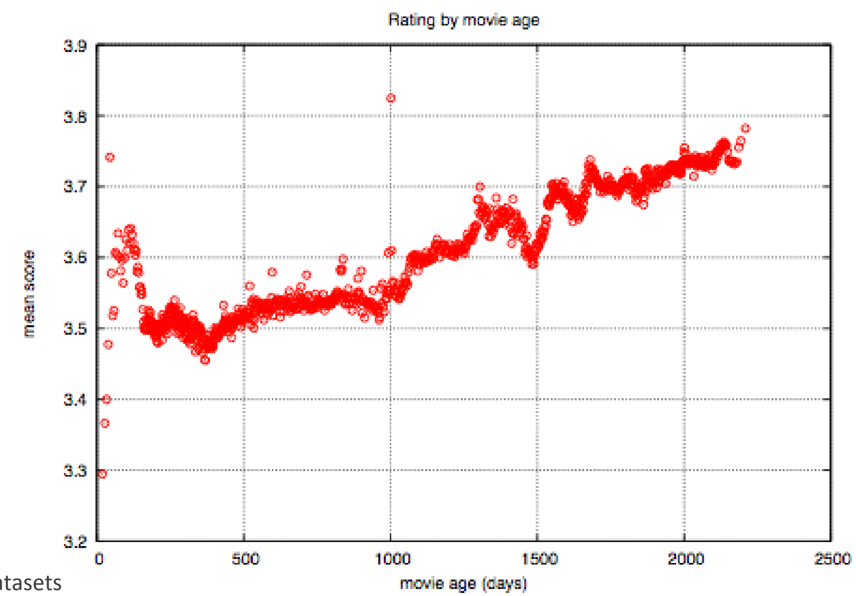
The Netflix Prize

- Training data
  - 100 million ratings, 480,000 users, 17,770 movies
  - 6 years of data: 2000-2005
- Test data
  - Last few ratings of each user (2.8 million)
  - Evaluation criterion: root mean squared error (RMSE)
  - Netflix Cinematch RMSE: 0.9514
- Competition
  - 2700+ teams
  - $1 million prize for 10% improvement on Cinematch
  - $50,000 progress prize for 8.43% improvement

Data: An Exploratory Study

- Sudden rise in the avg. rating (early 2004):
  - Improvements in Netflix
  - GUI improvements
  - Meaning of rating changed?
- Ratings increase with the movie age at the time of the rating





[Bellkor Team]

Data about the Movies

Most Loved Movies	Avg rating	Count
The Shawshank Redemption	4.593	137812
Lord of the Rings :The Return of the King	4.545	133597
The Green Mile	4.306	180883
Lord of the Rings :The Two Towers	4.460	150676
Finding Nemo	4.415	139050
Raiders of the Lost Ark	4.504	117456

Most Rated Movies
Miss Congeniality
Independence Day
The Patriot
The Day After Tomorrow
Pretty Woman
Pirates of the Caribbean

Highest Variance
The Royal Tenenbaums
Lost In Translation
Pearl Harbor
Miss Congeniality
Napolean Dynamite
Fahrenheit 9/11

Most Active Users

[Bellkor Team]

User ID	# Ratings	Mean Rating
305344	17,651	1.90
387418	17,432	1.81
2439493	16,560	1.22
1664010	15,811	4.26
2118461	14,829	4.08
1461435	9,820	1.37
1639792	9,764	1.33
1314869	9,739	2.95

5

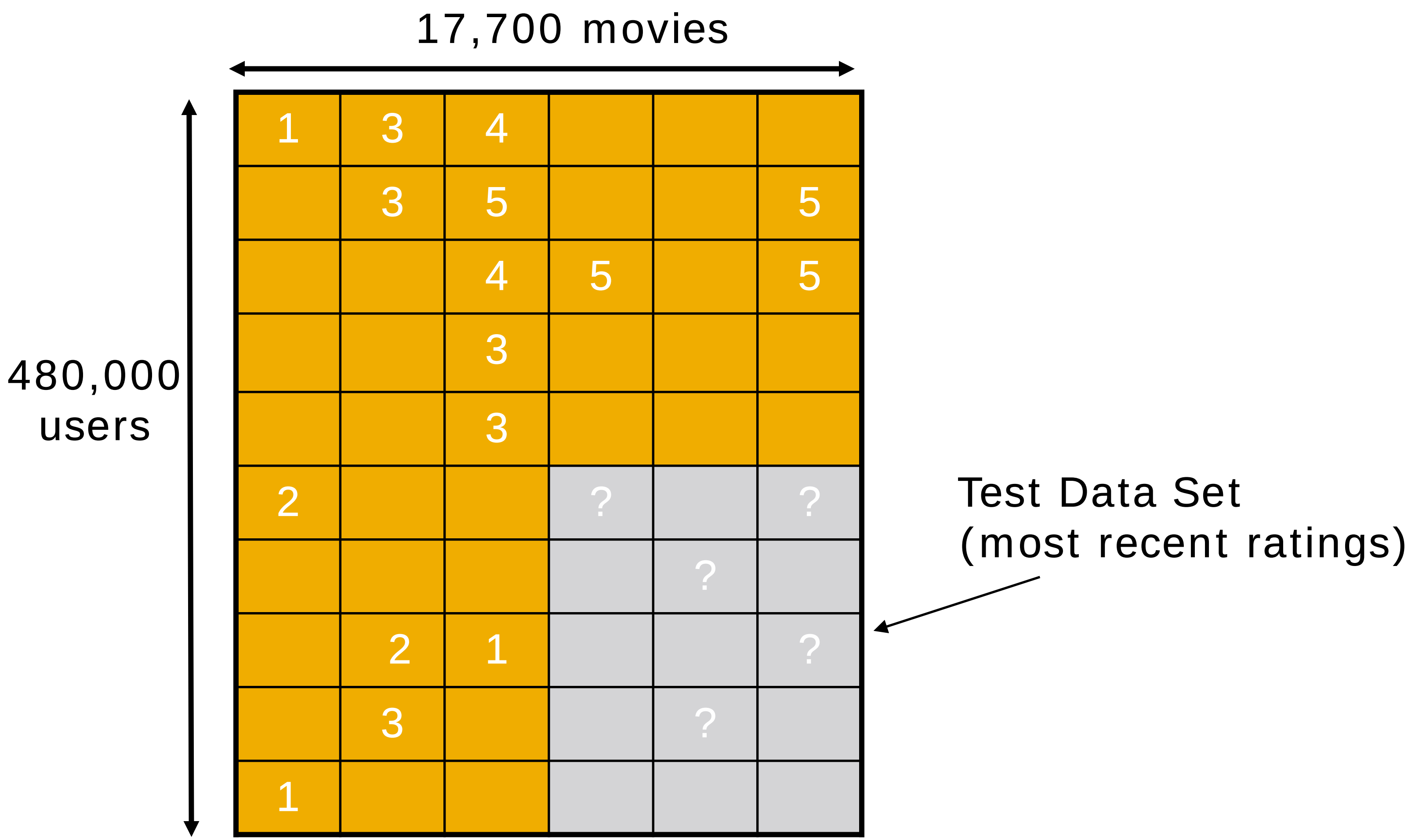
Utility Matrix

17,700 movies

480,000 users

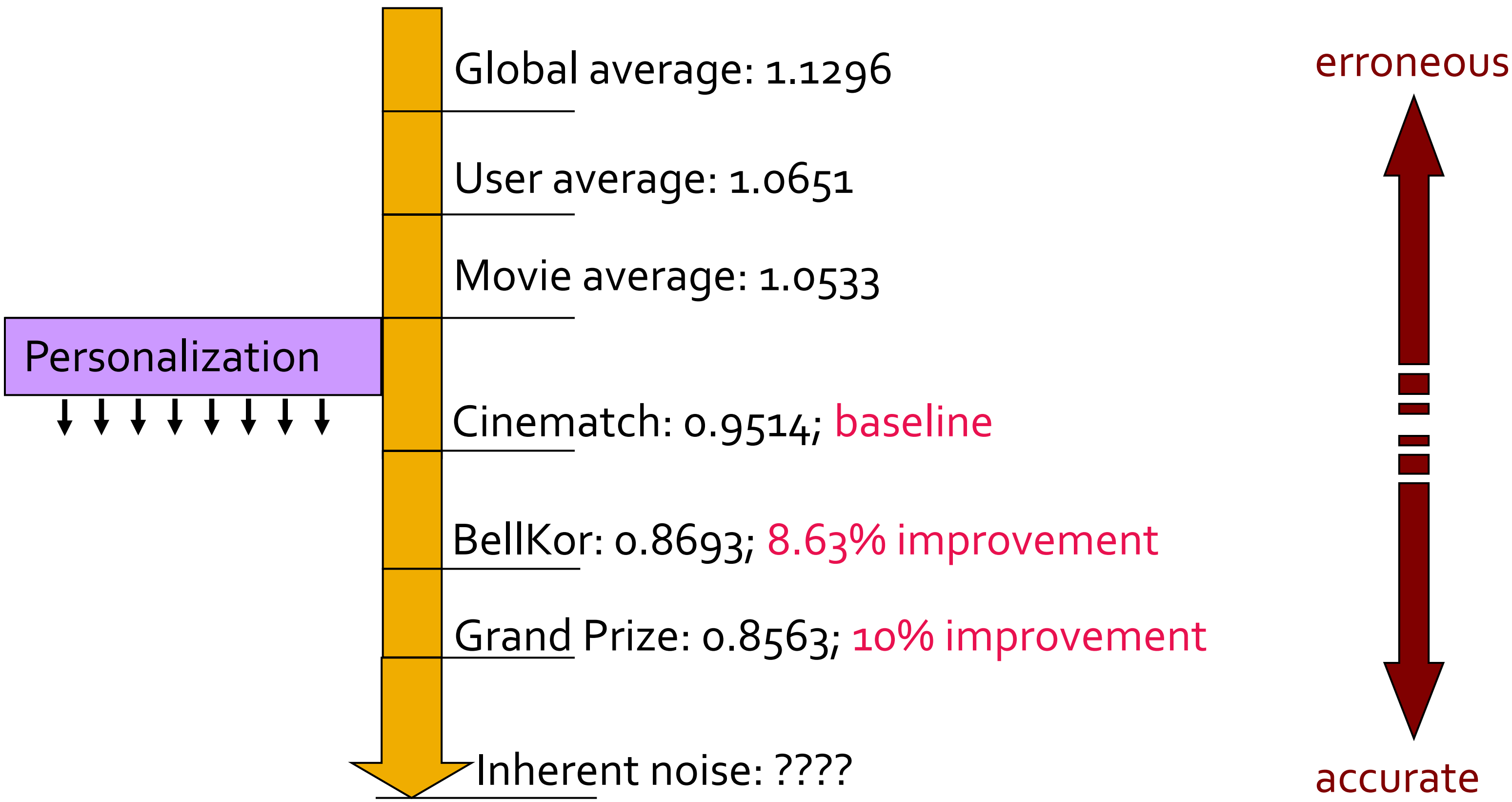
1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

Utility Matrix: Evaluation



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

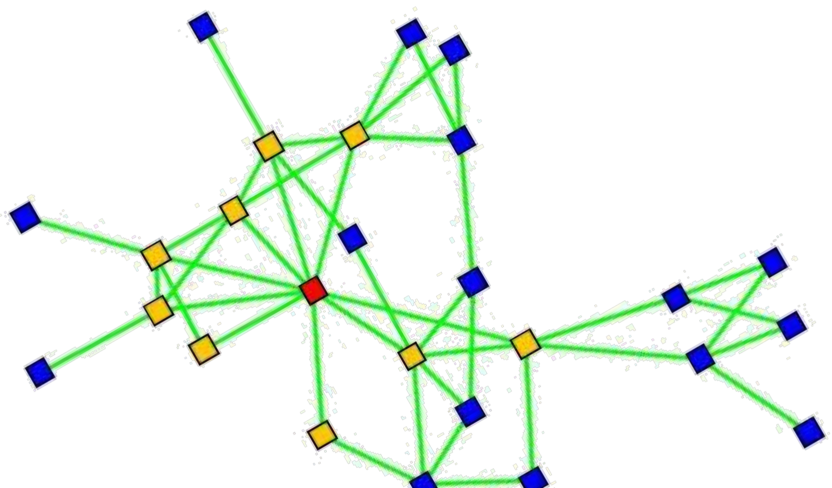
Important RMSEs



[Bellkor Team]

Local modeling through k-NN

- Earliest and most popular collaborative filtering method
- Derive unknown ratings from those of "similar" items (movie-movie variant)
- A parallel user-user flavor: rely on ratings of like-minded users (not in this talk)



[Bellkor Team]

2-Nearest Neighbor

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- unknown rating

- rating between 1 to 5

[Bellkor Team]

2-Nearest Neighbor

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- estimate rating of movie 1 by user 5

2-Nearest Neighbor

users

movies

	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Neighbor selection:
Identify movies similar to 1, rated by user 5

[Bellkor Team]

2-Nearest Neighbor

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Compute similarity weights:

$s_{13}=0.2$, $s_{16}=0.3$

2-Nearest Neighbor

[Bellkor Team]

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		**2.6**	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

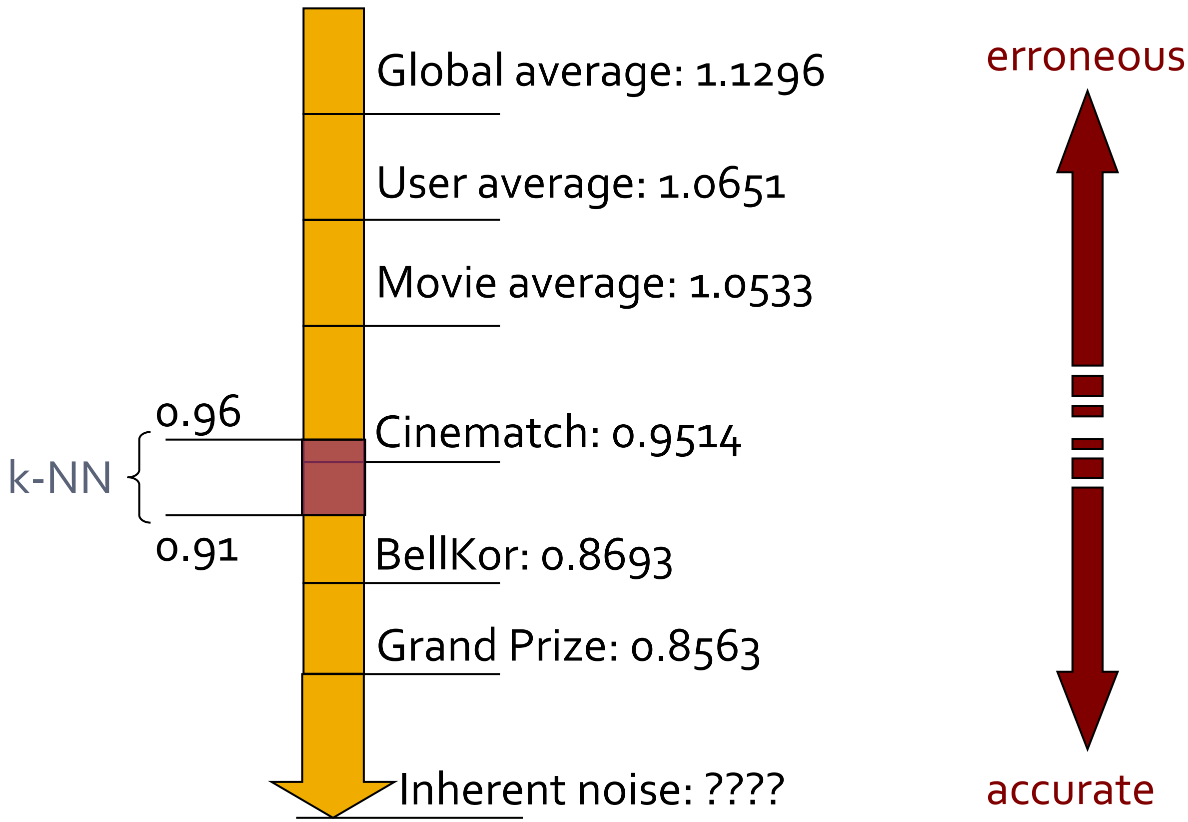
Predict by taking weighted average:

$(0.2*2+0.3*3)/(0.2+0.3)=2.6$

Properties of k-NN

- Intuitive
- No substantial preprocessing is required
- Easy to explain reasoning behind a recommendation
- Accurate?

k-NN on the RMSE scale



k-NN - Common practice

1. Define a similarity measure between items: s_{ij}
2. Select neighbors -- N(i;u):
   items most similar to i, that were rated by u
3. Estimate unknown rating, r_{ui}, as the weighted average:

$$r_{ui} = b_{ui} + \frac{\sum_{j \in N(i;u)} s_{ij} \left(r_{uj} - b_{uj} \right)}{\sum_{j \in N(i;u)} s_{ij}}$$

baseline estimate for r_{ui}

Interpolation weights

- Use a weighted sum rather than a weighted average:

$$\boxed{r_{ui} = b_{ui} + \sum_{j \in N(i;u)} w_{ij} \left(r_{uj} - b_{uj} \right)}$$

(Allow $\sum_{j \in N(i;u)} w_{ij} \neq 1$)

- Model relationships between item i and its neighbors
- Can be learnt through a least squares problem from all other users that rated i:

$$\mathrm{Min}_w \sum_{v \neq u} \left(\left(r_{vi} - b_{vi} \right) - \sum_{j \in N(i;u)} w_{ij} \left(r_{vj} - b_{vj} \right) \right)^2$$

[Bellkor Team]

Interpolation weights

$$\operatorname{Min}_w \sum_{v \neq u} \left((r_{vi} - b_{vi}) - \sum_{j \in N(i;u)} w_{ij} \left(\underline{r_{vj}} - b_{vj} \right) \right)^2$$

Mostly unknown

- Interpolation weights derived based on their role; no use of an arbitrary similarity measure
- Explicitly account for interrelationships among the neighbors

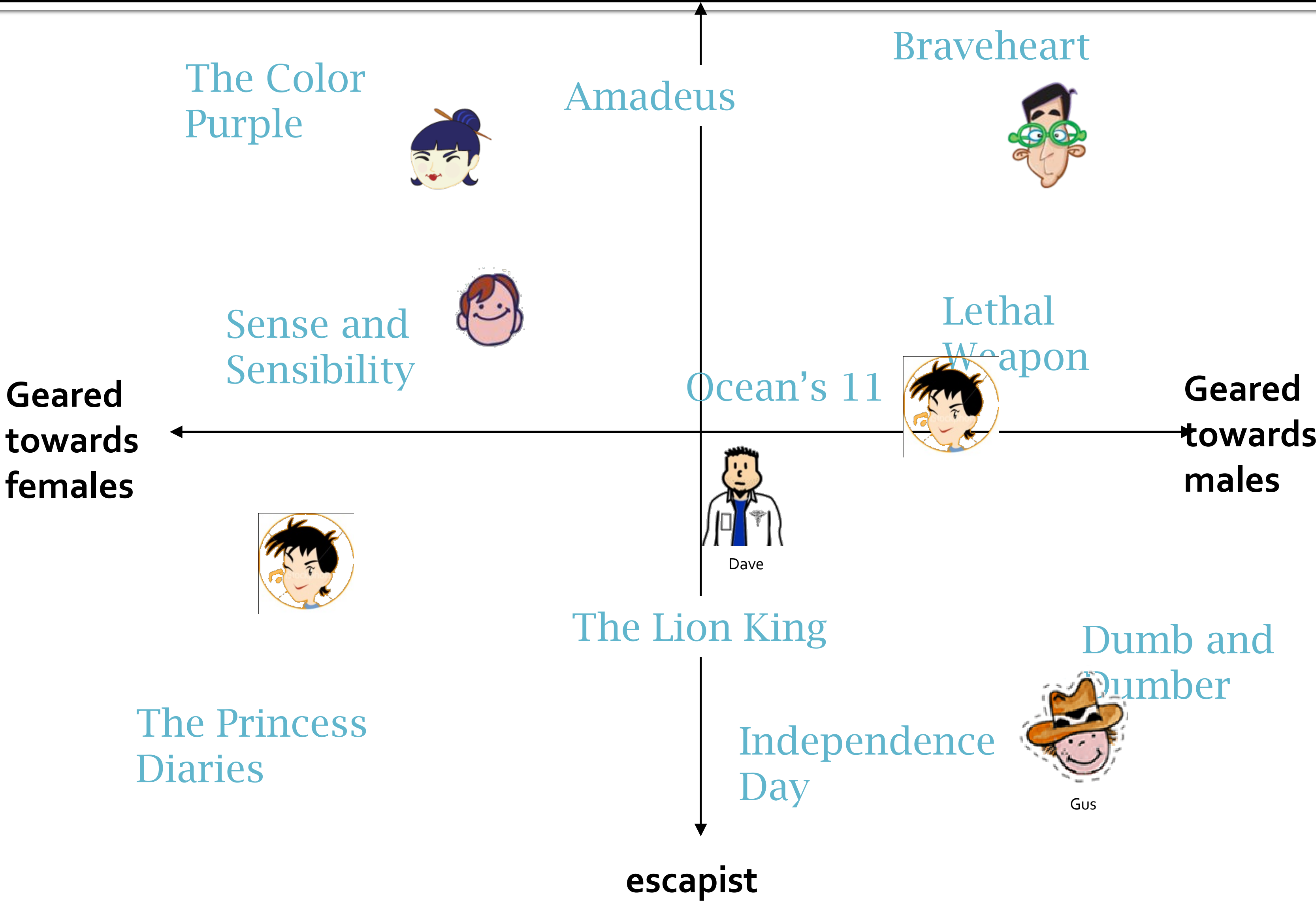
Challenges:

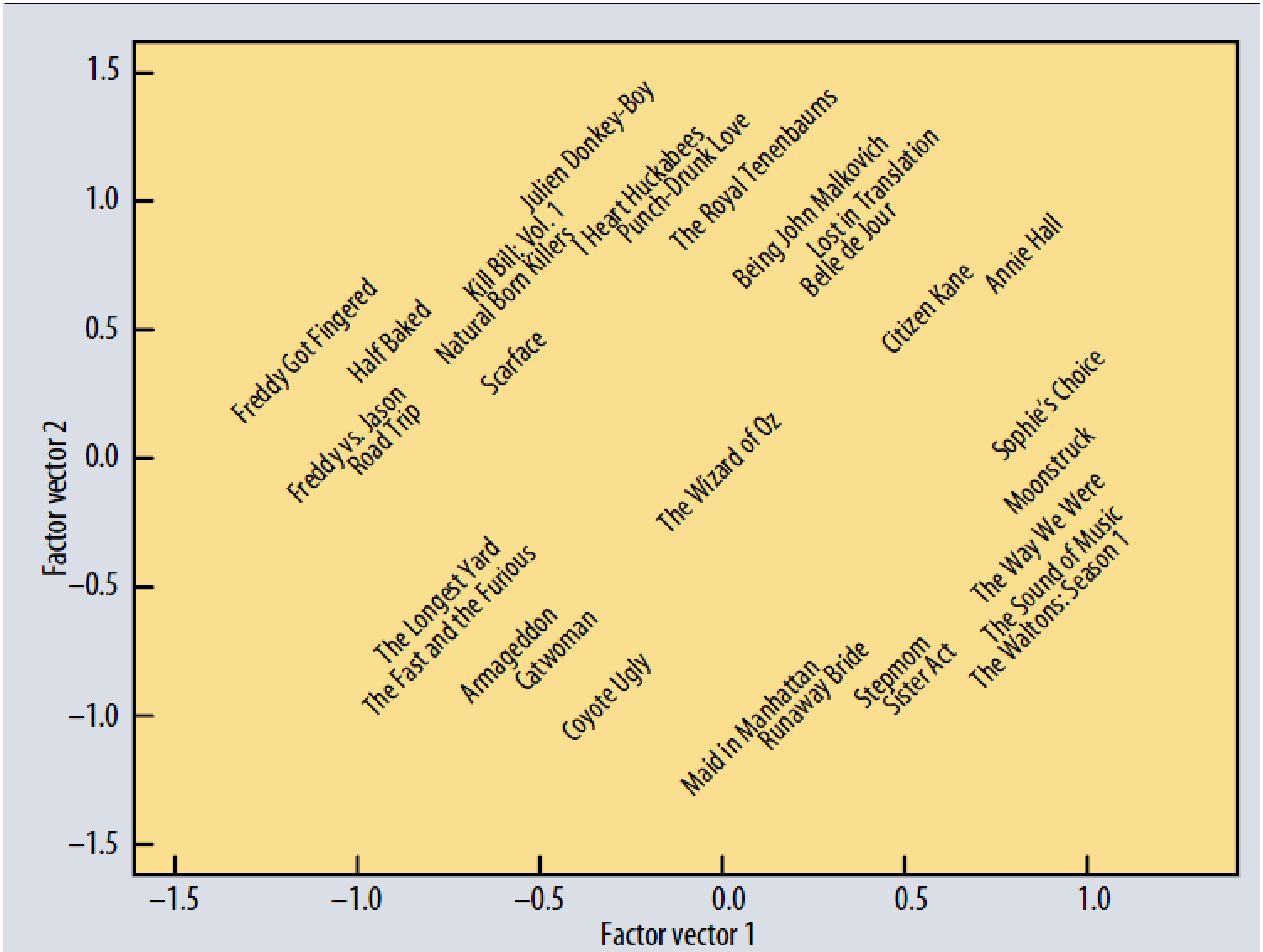
- Deal with missing values
- Avoid overfitting
- Efficient implementation

Estimate inner-products among movie ratings

[Bellkor Team]

Latent Factor Models (i.e., SVD)

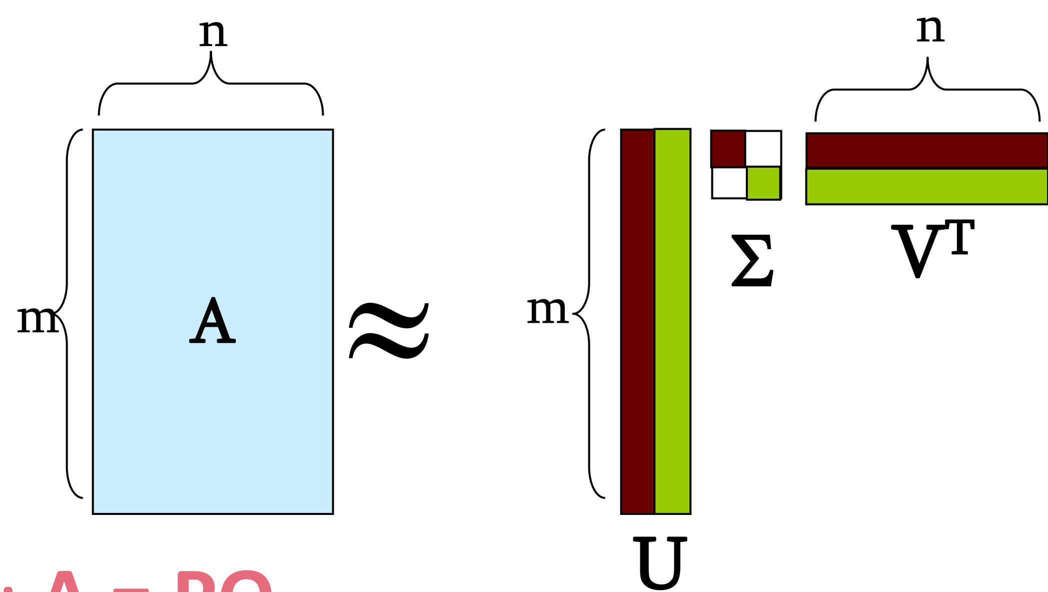




Koren, Bell, Volinksy, IEEE Computer, 2009

Latent Factor Models

- Recap: SVD



- SVD on Netflix data: **A = PQ**

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

~

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

A rank-3 SVD approximation

Ratings as products of factors:

users

items

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

~

~ items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

•

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

A rank-3 SVD approximation

[Bellkor Team]

Ratings as products of factors:

users

items

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

~

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

•

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

A rank-3 SVD approximation

[Bellkor Team]

Ratings as products of factors:

users

items												
	1		3			5			5		4	
			5		2.4		4			2	1	3
	2	4		1	2		3		4	3	5	
		2	4		5			4			2	
			4	3	4	2					2	5
	1		3		3			2			4	

~

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

●

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

A rank-3 SVD approximation

[Bellkor Team]

Latent Factor Models

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

~

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

Properties:

- SVD isn't defined when entries are unknown → use specialized methods
- Very powerful model → can easily overfit
- Probably most popular model among contestants

[Bellkor Team]

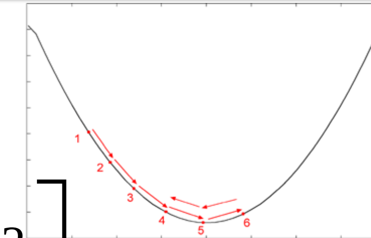
SVD: Dealing with missing data

- Want to minimize SSE for Test data
- One idea: Minimize SSE for Training data
  - Want large d to capture all the signals
  - But, Test RMSE begins to rise for d > 2
- Regularization is needed
  - Allow rich model where there are sufficient data
  - Shrink aggressively where data are scarce

$$\min_{P,Q} \sum_{training} (r_{ui} - q_i^T p_u)^2 + \lambda \left[\sum_u \|p_u\|^2 + \sum_i \|q_i\|^2 \right]$$

[Bellkor Team]

Stochastic Gradient Descent



- Want to find matrices P and Q:

$$\min_{P,Q} \sum_{training} (r_{ui} - q_i^T p_u)^2 + \lambda \left[\sum_u \|p_u\|^2 + \sum_i \|q_i\|^2 \right]$$

- Online "stochastic" gradient decent:
  - Initialize P and Q (random?, using SVD?)
  - Then iterate over ratings and update q_i, p_u:
  - $\varepsilon_{ui} = r_{ui} - q^T{}_i \; p_u$
  - $q_i \leftarrow q_i + \gamma \, (\varepsilon_{ui} \, p_u - \lambda q_i)$
  - $p_u \leftarrow p_u + \gamma \, (\varepsilon_{ui} \, q_i - \lambda p_u)$

γ… learning rate

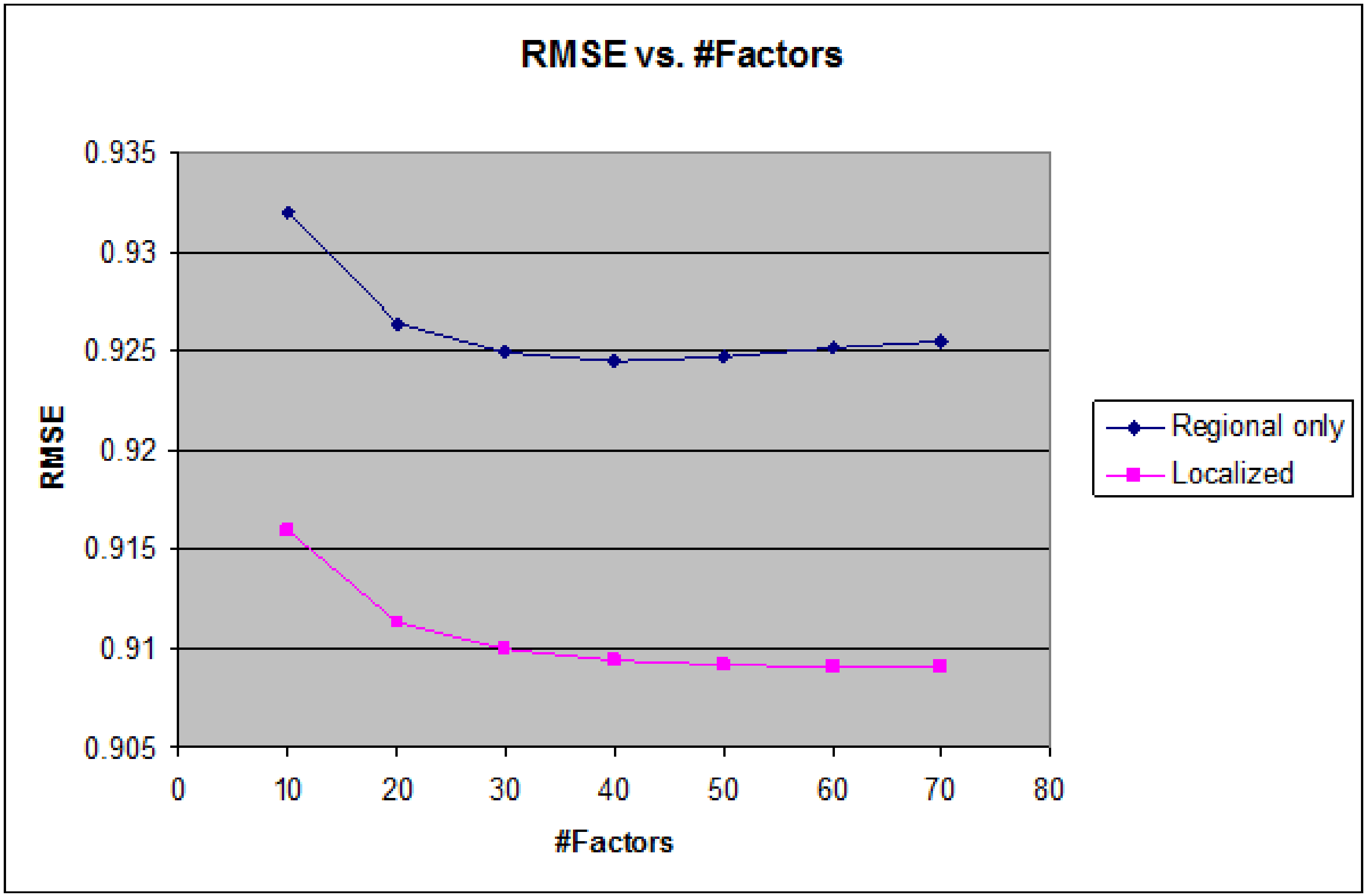
Localized "SVD"

- SVD uses all of a user's ratings to train the user's factors
- But what if the user is multiple people?
  - Different factor values may apply to movies rated by Mom vs. Dad vs. the Kids
- This approach computes user factors, p_u , specific to the movie being predicted:

$$r_{ui} = q_i^T p_u(i)$$

  - Vector $p_u(i)$ models behavior of u on items like i

Improvement from Localized SVD



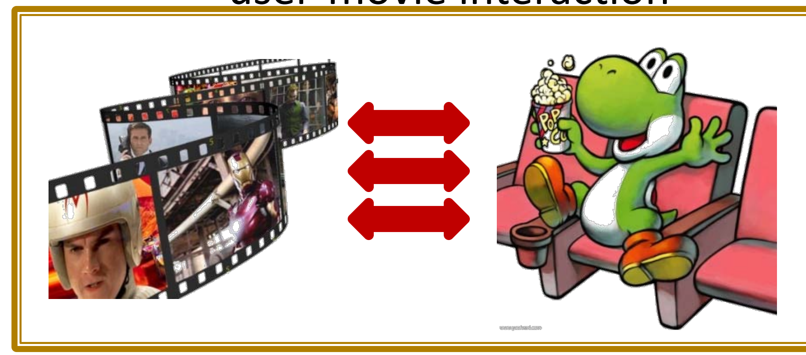
[Bellkor Team]

Modeling Systematic Biases

user bias

movie bias

user-movie interaction



Baseline predictor

- Separates users and movies
- Often overlooked
- Benefits from insights into users' behavior
- Among the main practical contributions of the competition

User-movie interaction

- Characterizes the matching between users and movies
- Attracts most research in the field
- Benefits from algorithmic and mathematical innovations

- μ = overall mean rating
- b_u = mean rating for user *u*
- b_i = mean rating for movie *i*

Baseline Predictor

- We have expectations on the rating by user u of movie i, even without estimating u's attitude towards movies like i

- Rating scale of user u
- Values of other ratings user gave recently (day-specific mood, anchoring, multi-user accounts)

- (Recent) popularity of movie i
- Selection bias; related to number of ratings user gave on the same day ("frequency")

Modeling Systematic Biases

$$r_{ui} \approx \mu + b_u + b_i + \text{user-movie interactions } q^T_i \; p_u$$

overall mean rating | mean rating for user u | mean rating for movie i

- Example:
  - Mean rating m = 3.7
  - You are a critical reviewer: your ratings are 1 lower than the mean: b_u = -1
  - Star Wars gets a mean rating of 0.5 higher than average movie: b_i = + 0.5
  - Predicted rating for you on Star Wars
    = 3.7 - 1 + 0.5 = 3.2

Objective Function

- Solve:

$$\min_{Q,P} \sum_{(u,i)\in R} \left(r_{ui} - (\mu + b_u + b_i + q_i^T p_u)\right)^2$$

goodness of fit

$$+ \lambda \left(\|q_i\|^2 + \|p_u\|^2 + \|b_u\|^2 + \|b_i\|^2 \right)$$

regularization

Typically selected via grid-search on a validation set

- Stochastic gradient decent to the rescue!

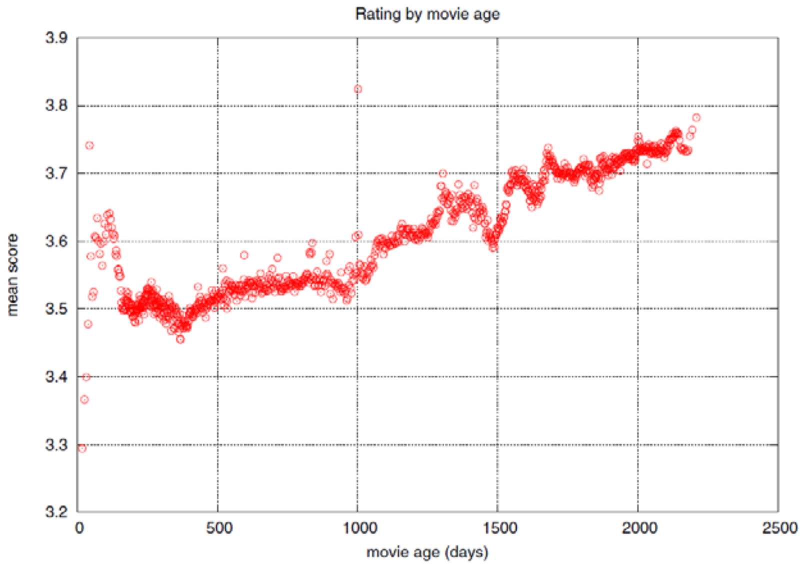
Temporal Biases

- Original model:

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

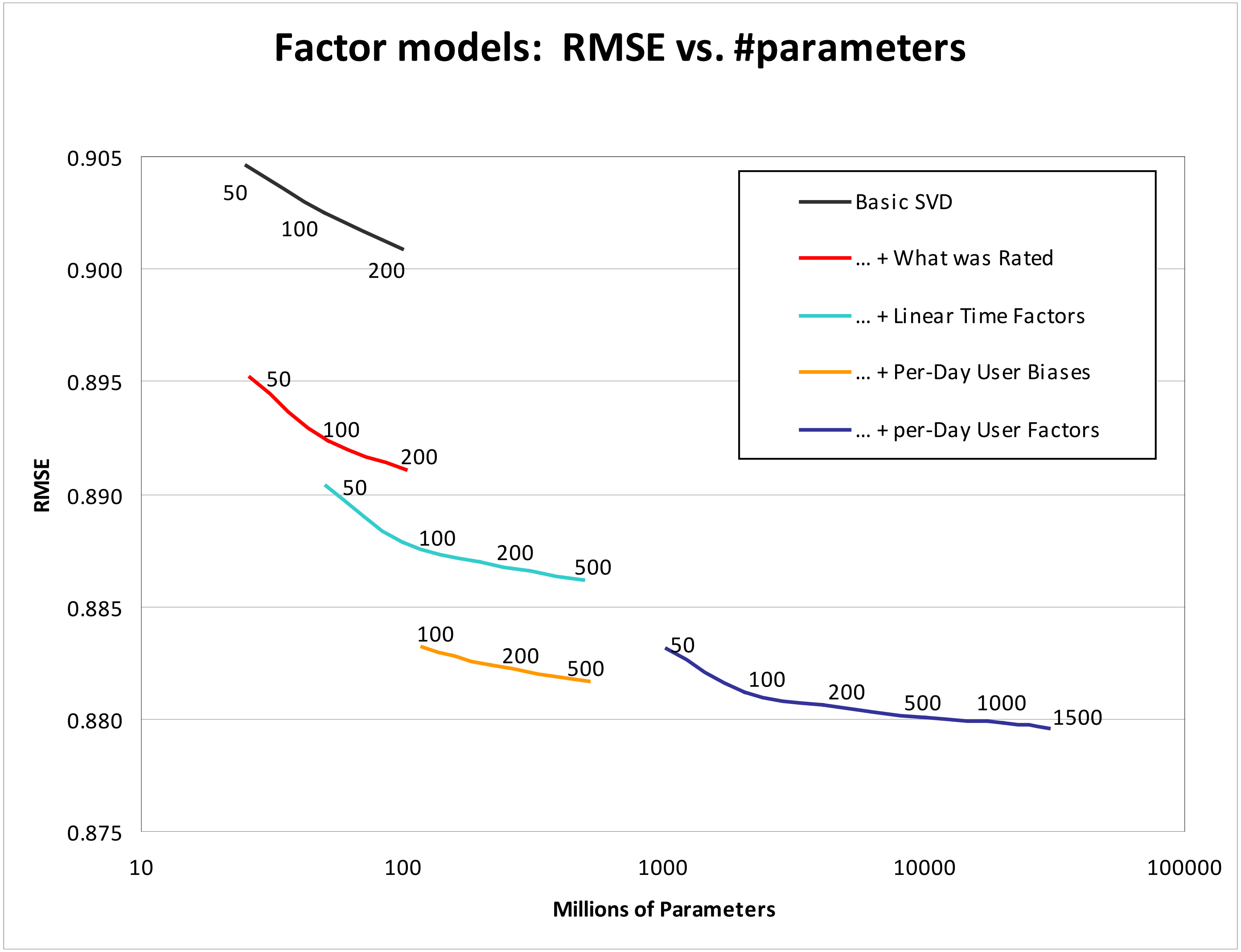
- Add time dependence to biases:

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



  - Time-dependence parametrized by linear trends
  - Add time dependence to user "factor weights"
  - Models the fact that user's interests over "genres" (the *q*s) may change over time
  - Y. Koren, Collaborative filtering with temporal dynamics, KDD '09

Netflix: Performance



June 26th 2009: after 1000 days & 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

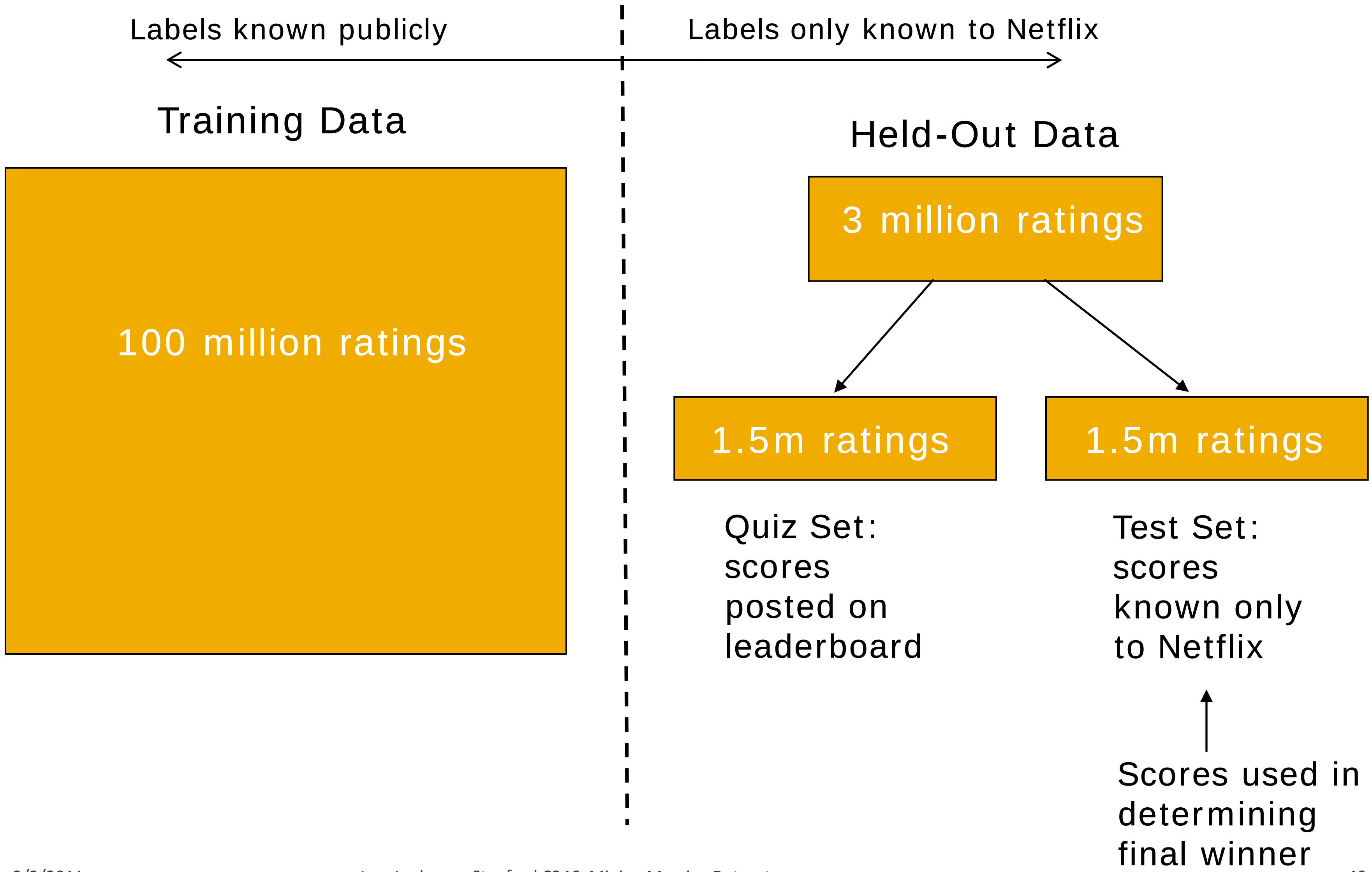
The Leading Team

- 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
- 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

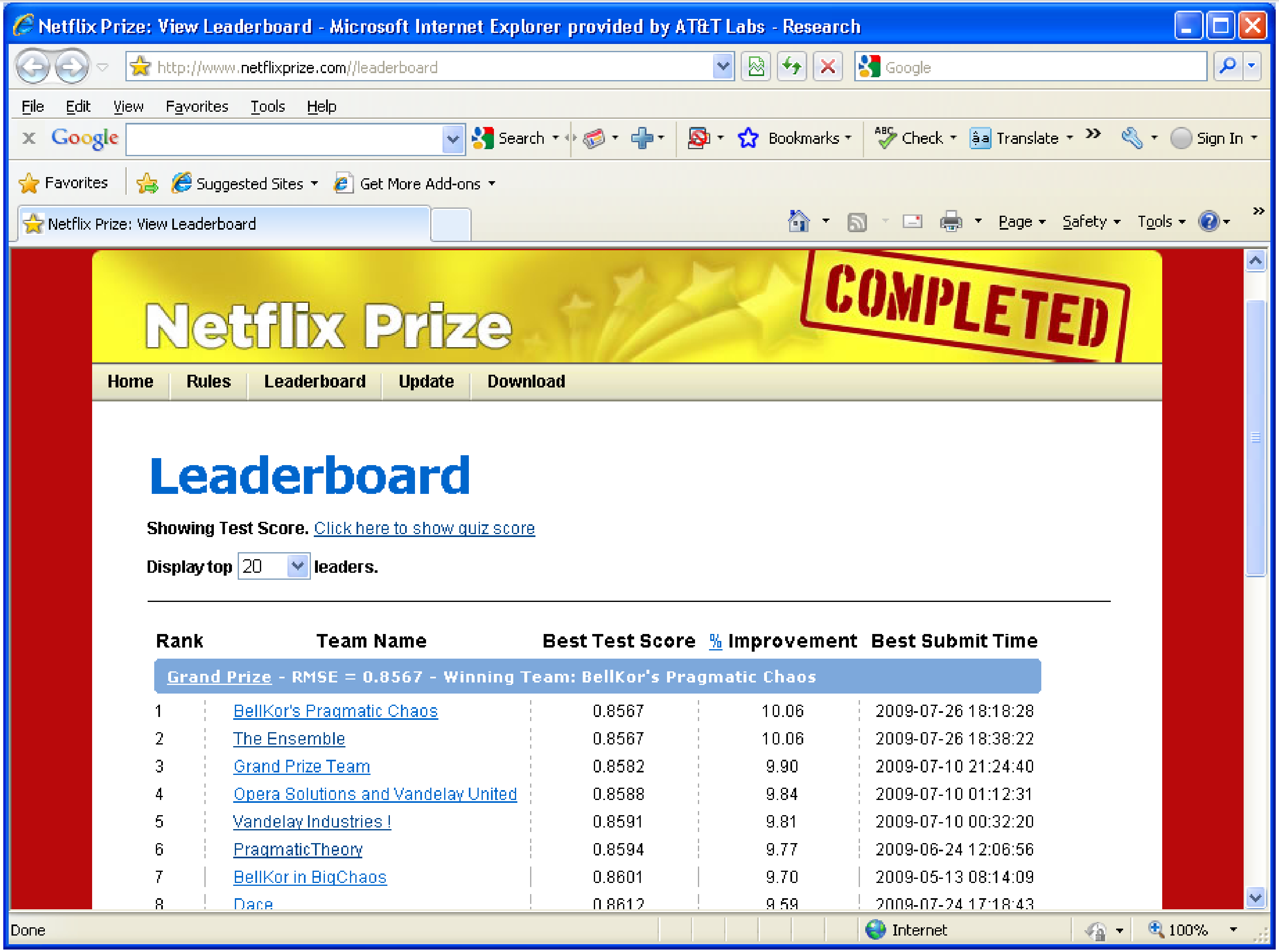
Competition Structure



24 Hours from the Deadline

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

Final Test Set Leader Board



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	0.8582	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

Million Dollars: Sept 21st 2009

Acknowledgments

- Most slides and plots borrowed from Yehuda Koren, Robert Bell and Padhraic Smyth