Collaborative Filtering Overview 1

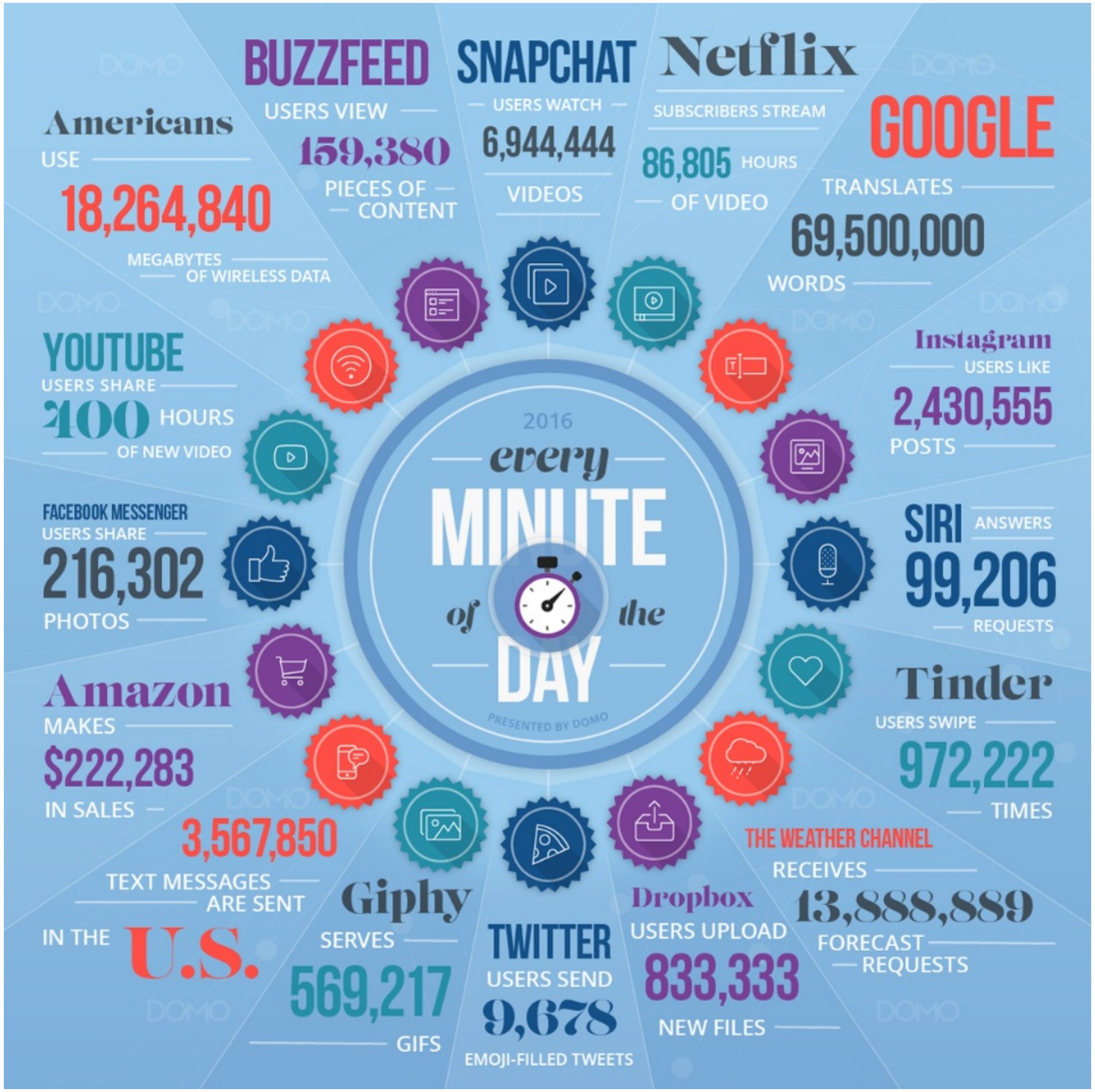
BY GUOKAN SHANG

Research Engineer @ Labs, Linagora &
PhD Candidate @ DaSciM, LIX

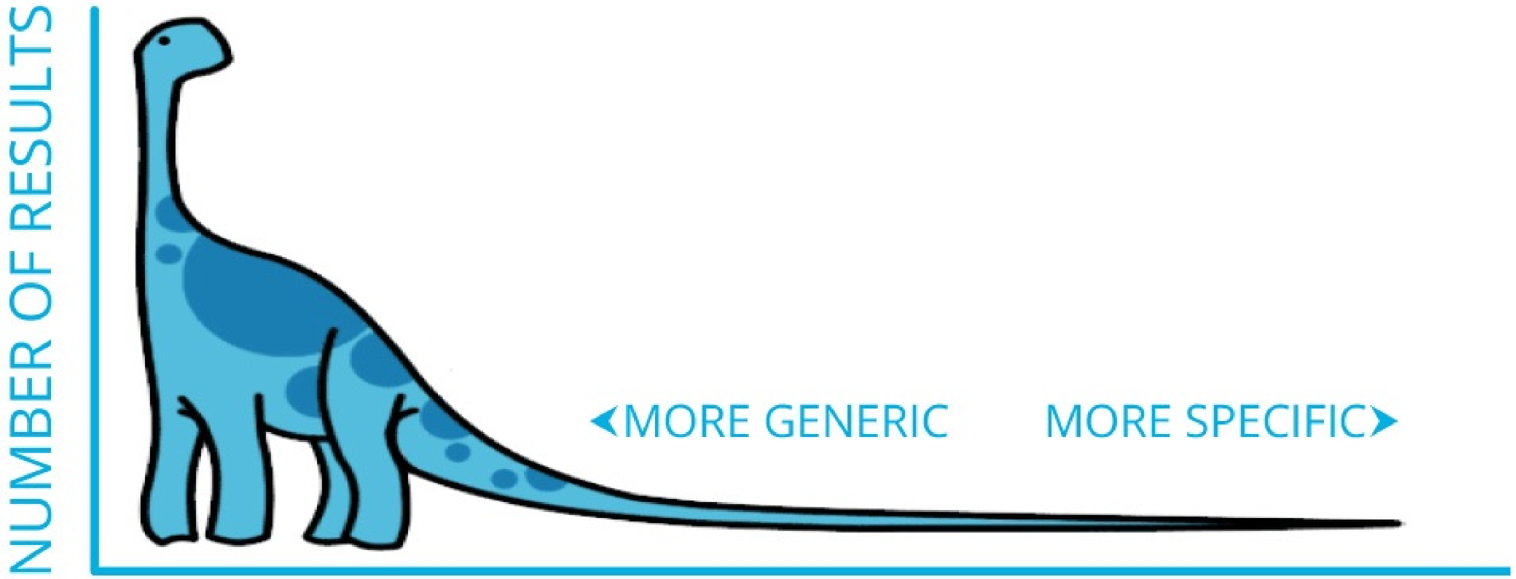
Email: `guokan.shang@hotmail.com`
Web: `http://www.lix.polytechnique.fr/Labo/Guokan.Shang/`

July 27, 2017

Data Never Sleeps



Long Tail



Information Overload

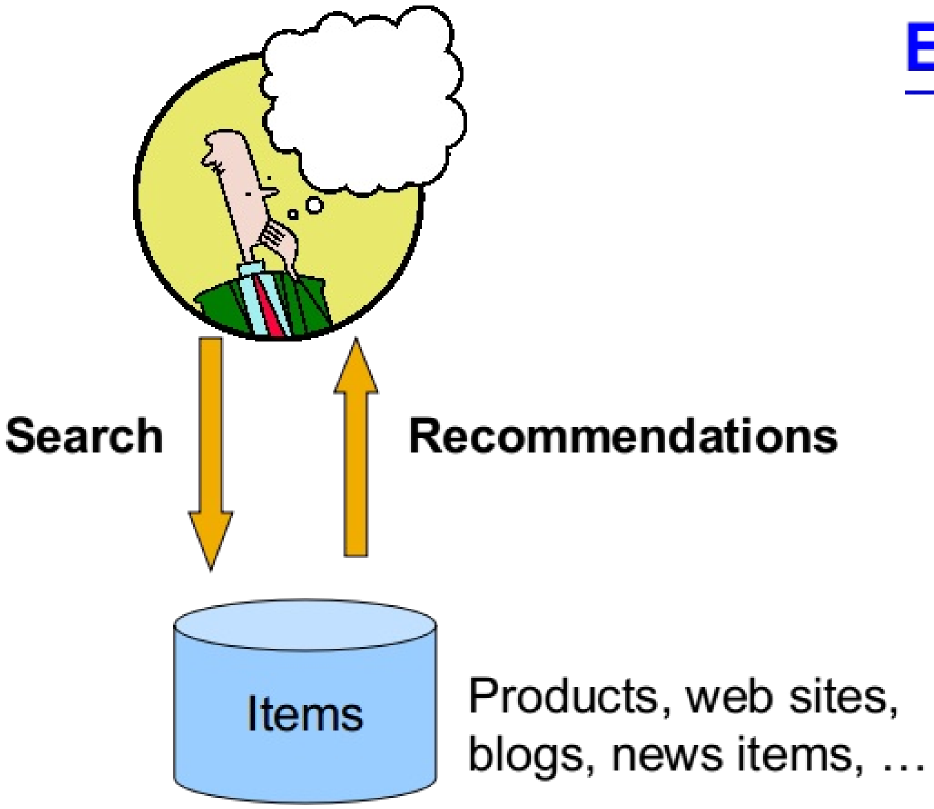
Information overload refers to the difficulty a person can have understanding an issue and making decisions that can be caused by the **presence of too much information**.

> Information overload occurs when the amount of input to a system exceeds its processing capacity. Decision makers have fairly limited ***cognitive processing capacity***. Consequently, when information overload occurs, it is likely that a **reduction in decision quality** will occur.
>
> — Bertram Myron Gross (1964)

There are various solutions in computer science that can be used to mitigate information overload,

- Search
- Recommendation

Introduction



Examples:

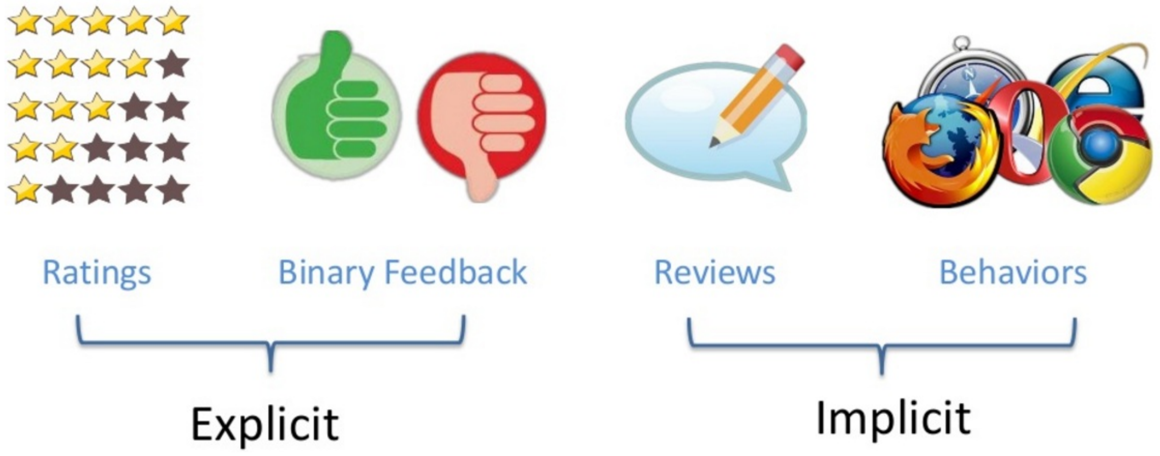
Recommendation

The basic idea of ***Recommender Systems*** is to utilize various sources of data to infer customer **interests**.

Recommendation analysis is often based on the previous **interaction** between users and items, because ***past interests and proclivities are often good indicators of future choices***.

- Demographic-based
- Content-based
- **Collaborative Filtering**
  - Memory-based (kNN approaches)
  - Model-based (latent factor models)
- Hybrid

Rating - Feedback - Preference



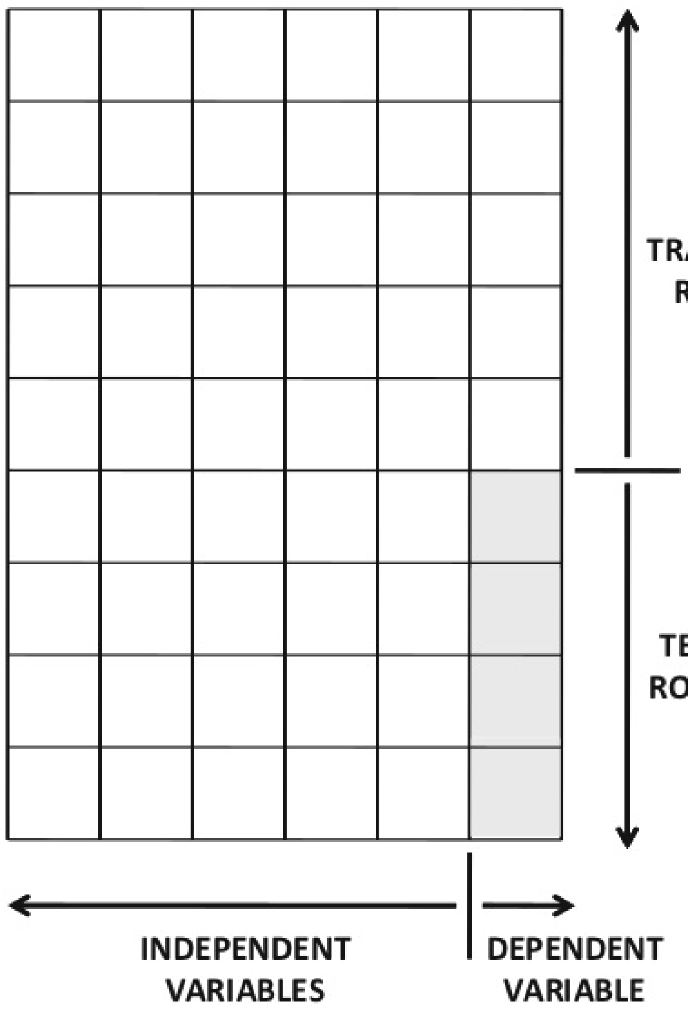
Explicit feedback:

continuous / interval-based: $\{-2, -1, 0, 1, 2\}$, forced choice rating system: $\{-2, -1, 1, 2\}$, ordinal ratings: {Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree}, binary ratings: {like, dislike, unspecified}

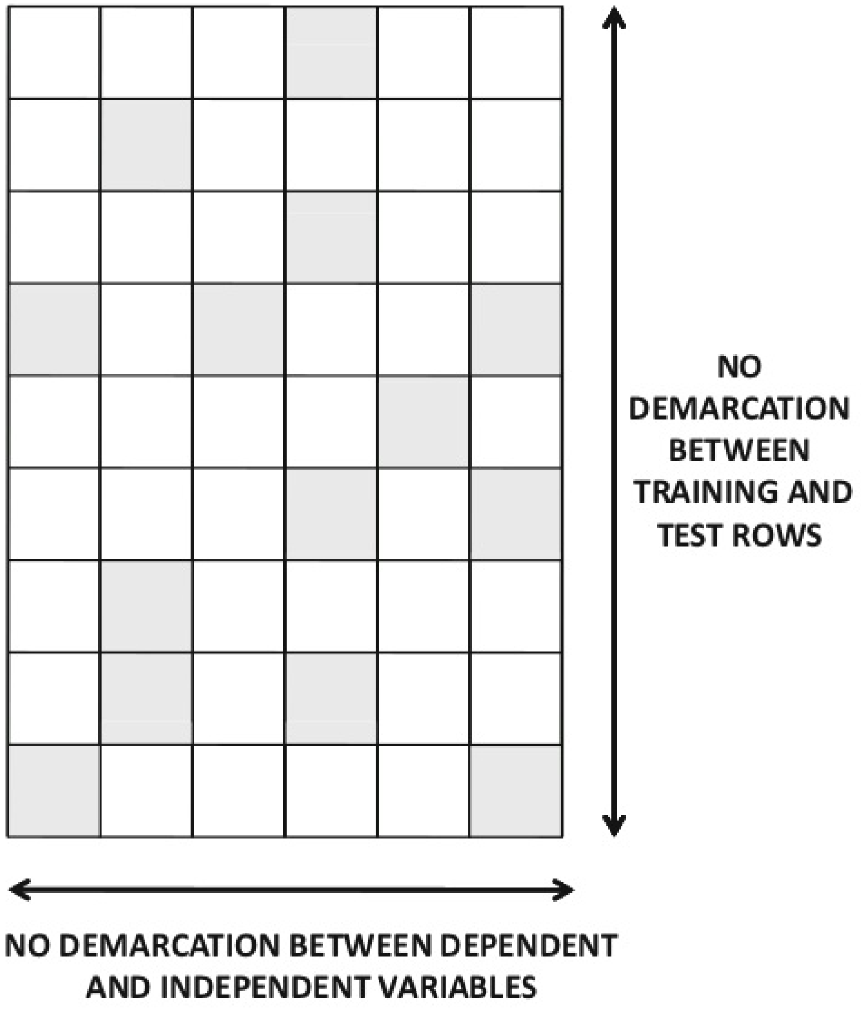
Implicit feedback:

buying, browsing, viewing, clicking behavior

Matrix Completion



(a) Classification


NO DEMARCATION BETWEEN TRAINING AND TEST ROWS
NO DEMARCATION BETWEEN DEPENDENT AND INDEPENDENT VARIABLES

(b) Collaborative filtering

Notation

$R, R_{\text{train}}, R_{\text{test}}, \hat{R}$: the set of all ratings, the training set, the test set, and the set of predicted ratings.

U: the set of all users, u and v denotes users.

I: the set of all items, i and j denotes items.

I_u: the set of all items rated by user u.

I_{uv}: the set of all items rated by both users u and v.

r_{ui}: the true rating of user u for item i.

$\hat{r}_{ui}$: the estimated rating of user u fot item i.

μ, μ_u, μ_i: the mean of all ratings, the mean of all ratings given by user u, the mean of all ratings to item i.

Evaluation Metrics

Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} |r_{ui} - \hat{r}_{ui}|$$

Root Mean Squared Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2}$$

Normal Predictor

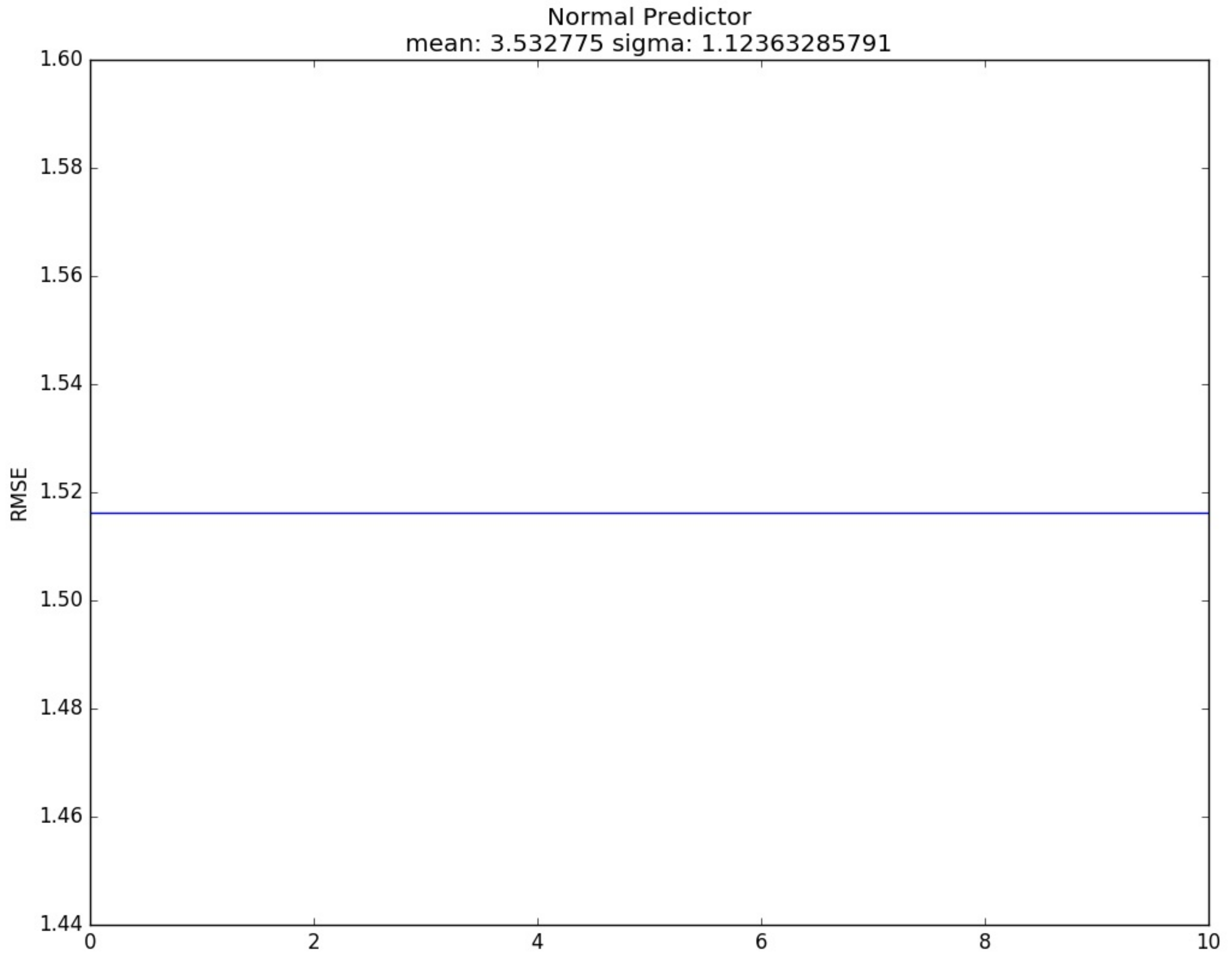
The prediction $\hat{r}_{ui}$ is generated from a normal distribution $N(\hat{\mu}, \hat{\sigma}^2)$,

where $\hat{\mu}$ and $\hat{\sigma}$ are estimated from the training data using Maximum Likelihood Estimation:

$$\hat{\mu} = \frac{1}{|R_{train}|} \sum_{r_{ui} \in R_{train}} r_{ui}$$

$$\hat{\sigma} = \sqrt{\sum_{r_{ui} \in R_{train}} \frac{(r_{ui} - \hat{\mu})^2}{|R_{train}|}}$$

Experiment


Normal Predictor
mean: 3.532775 sigma: 1.12363285791
RMSE
1.60
1.58
1.56
1.54
1.52
1.50
1.48
1.46
1.44
0
2
4
6
8
10

Baseline Estimate

Typical CF data exhibit large user and item biases—i.e., systematic tendencies for some users to give higher ratings than others, and for some items to receive higher ratings than others.

We will encapsulate those effects, which do not involve user-item interaction, within the baseline predictors (also known as biases).

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i$$

μ is the overall average rating. The parameters b_u and b_i indicate the observed deviations of user u and item i, respectively, from the average. b_{ui} is the baseline rating of user u for item i.

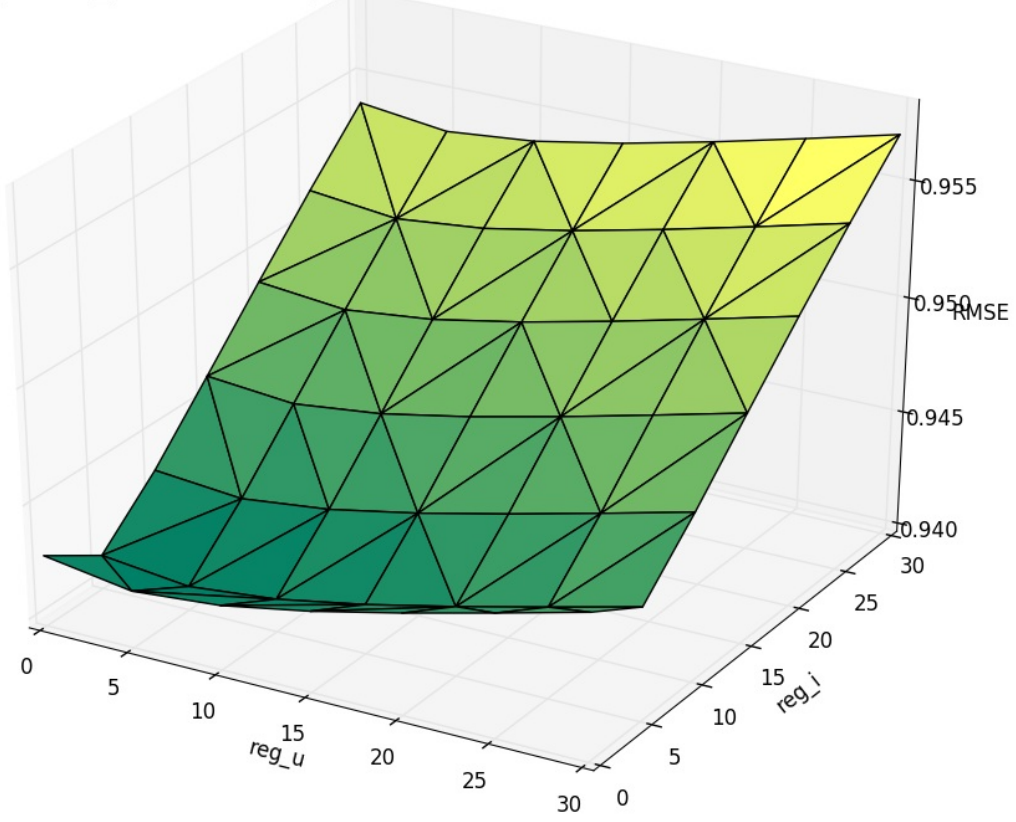
$$\min_{b_u, b_i} \sum_{u \in U, v \in V} (r_{ui} - (\mu + b_u + b_i))^2 + \lambda_u \sum_u b_u^2 + \lambda_i \sum_i b_i^2$$

gradient descent

Experiment

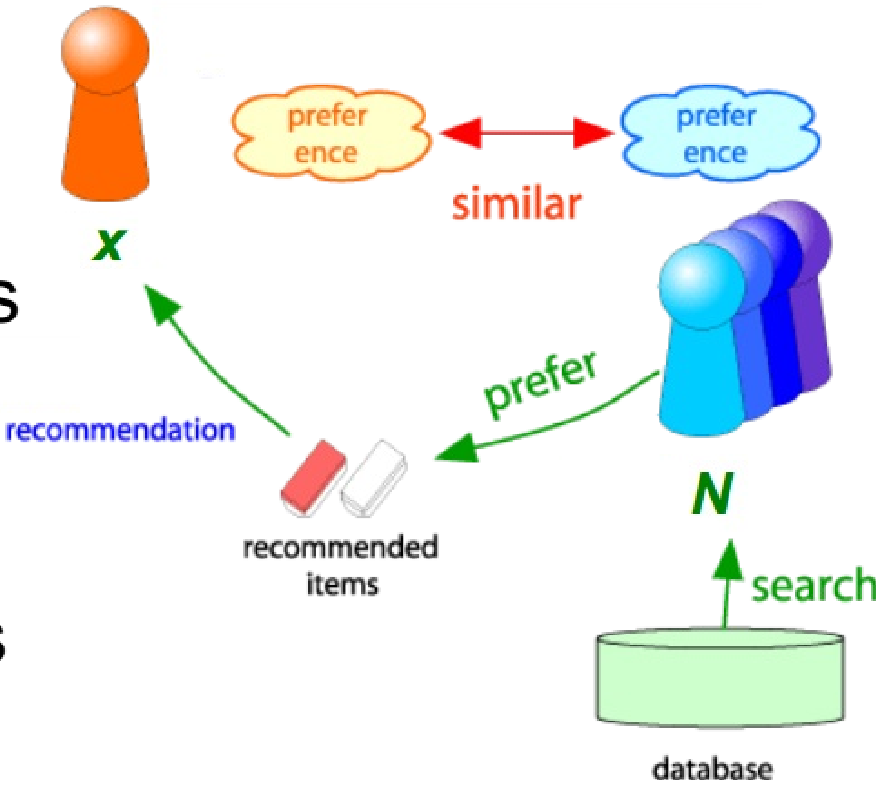
ALS Baseline estimate

{'reg_u': 5, 'n_epochs': 20, 'reg_i': 5, 'method': 'als', 'rmse': 0.94067412640646286}



Neighborhood Approaches

- Consider user ***x***
- Find set ***N*** of other users whose ratings are "**similar**" to ***x***'s ratings
- Estimate ***x***'s ratings based on ratings of users in ***N***



$N_i^k(u)$: the k nearest neighbors of user u that have rated item i.

Similarity Metrics

$$\text{CosineSim}(u,v) = \frac{\sum_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}}$$

$$\text{MeanSquaredDifference}(u,v) = \frac{1}{|I_{uv}|} \cdot \sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2$$

$$\text{MeanSquaredDifferenceSim}(u,v) = \frac{1}{\text{msd}(u,v) + 1}$$

$$\text{PearsonCorrelation}(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \mu_u) \cdot (r_{vi} - \mu_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \mu_u)^2} \cdot \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \mu_v)^2}}$$

$$\text{PearsonBaselineSim}(u,v) = \hat{\rho}_{uv} = \frac{\sum_{i \in I_{uv}} (r_{ui} - b_{ui}) \cdot (r_{vi} - b_{vi})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - b_{ui})^2} \cdot \sqrt{\sum_{i \in I_{uv}} (r_{vi} - b_{vi})^2}}$$

$$\text{PearsonBaselineShrunkSim}(u,v) = \frac{|I_{uv}| - 1}{|I_{uv}| - 1 + \text{shrinkage}} \cdot \hat{\rho}_{uv}$$

The shrinkage parameter helps to avoid overfitting when only few ratings are available.

kNN Inspired Algorithms

KNNBasic
$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot r_{vi}}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$

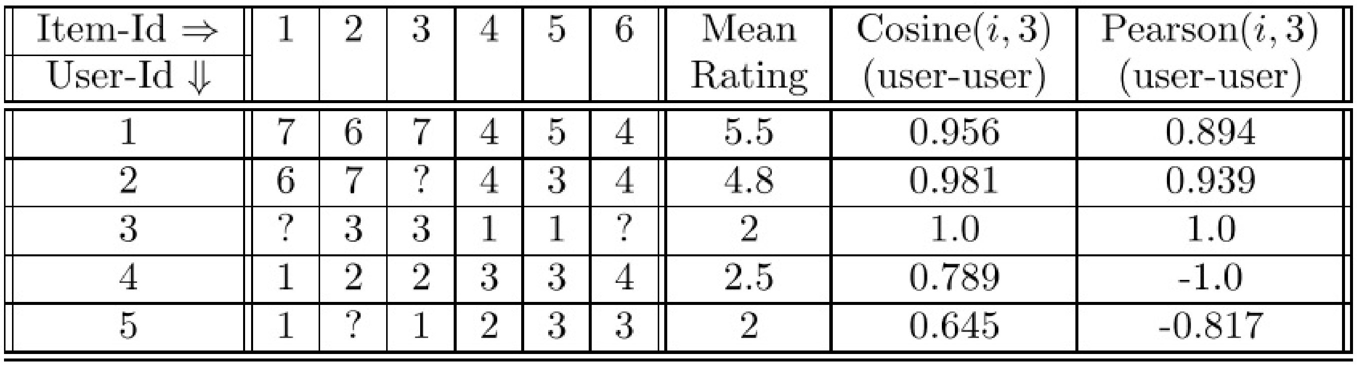
KNNWithMeans
$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$

KNNWithZscore
$$\hat{r}_{ui} = \mu_u + \sigma_u \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot z_{vi}}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$
$$z_{vi} = \frac{r_{vi} - \mu_v}{\sigma_v}$$
$$\sigma_v = \sqrt{\frac{\sum_{i \in I_v} (r_{vi} - \mu_v)^2}{|I_v| - 1}}$$

KNNBaseline
$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - b_{vi})}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$

kNN Example

Table 2.1: User-user similarity computation between user 3 and other users

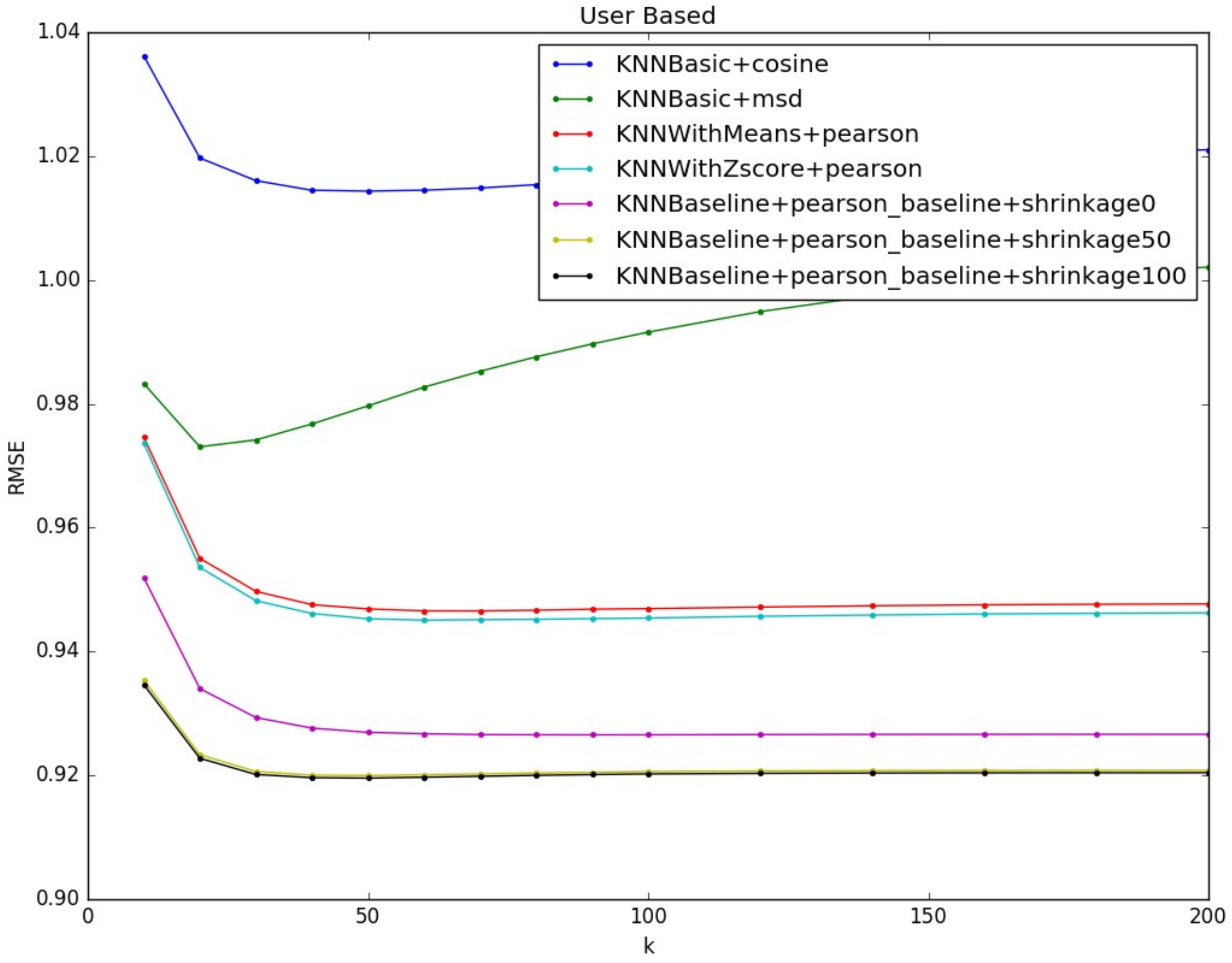


Item-Id ⇒ / User-Id ⇓	1	2	3	4	5	6	Mean Rating	Cosine$(i,3)$ (user-user)	Pearson$(i,3)$ (user-user)
1	7	6	7	4	5	4	5.5	0.956	0.894
2	6	7	?	4	3	4	4.8	0.981	0.939
3	?	3	3	1	1	?	2	1.0	1.0
4	1	2	2	3	3	4	2.5	0.789	-1.0
5	1	?	1	2	3	3	2	0.645	-0.817

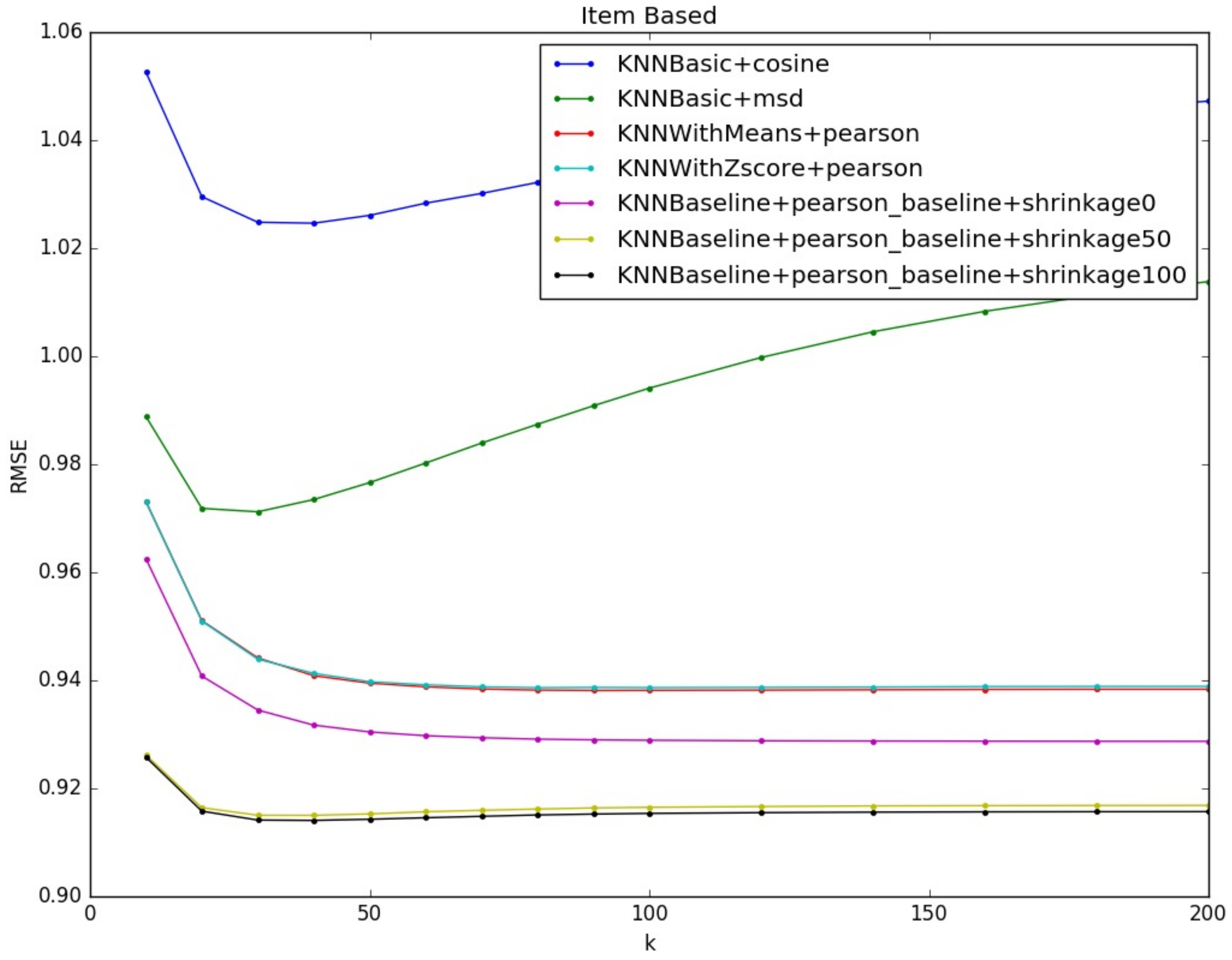
$$\hat{r}_{31} = \frac{7 * 0.894 + 6 * 0.939}{0.894 + 0.939} \approx 6.49$$

$$\hat{r}_{36} = \frac{4 * 0.894 + 4 * 0.939}{0.894 + 0.939} = 4$$

User-based Experiment



Item-based Experiment



SVD and PCA

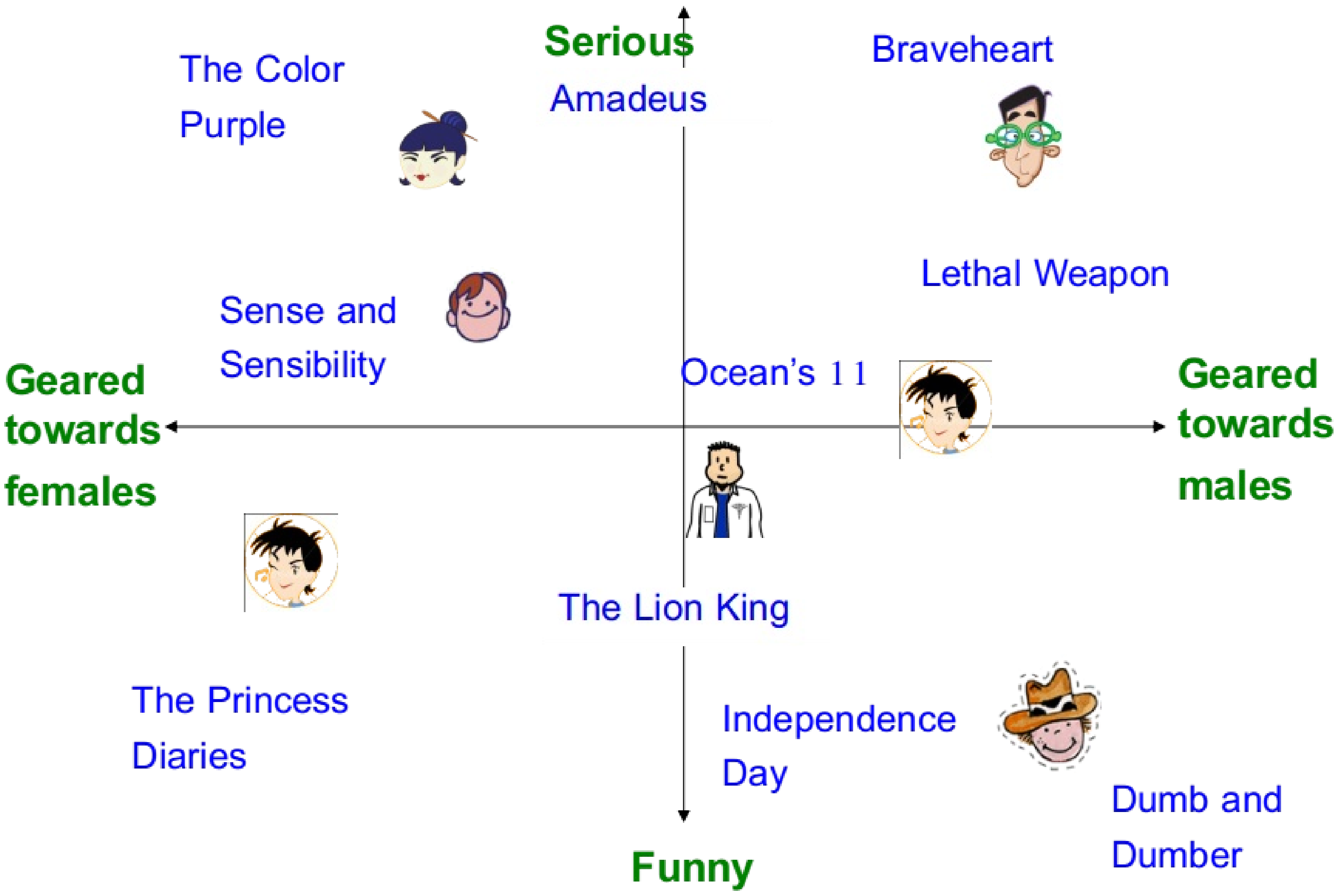
Pairwise similarities are hard to robustly compute in sparse rating matrices, dimensionality reduction provides a dense low-dimensional representation in terms of latent factors. Even when two users have very few items rated in common, a distance can be computed between their low-dimensional latent vectors.

The similarity computations in the reduced representation are more robust because the new low-dimensional vector is fully specified. Furthermore, the similarity computations are more efficient because of the low dimensionality of the latent representation.

The main problem of using such an approach is that the ratings matrix is **not fully specified**. As a result, this factorization is undefined.

Nevertheless, it is possible to recast the formulation as an optimization problem, in which the squared error of factorization is optimized only over the **observed** entries of the ratings matrix.

Latent Factor Models



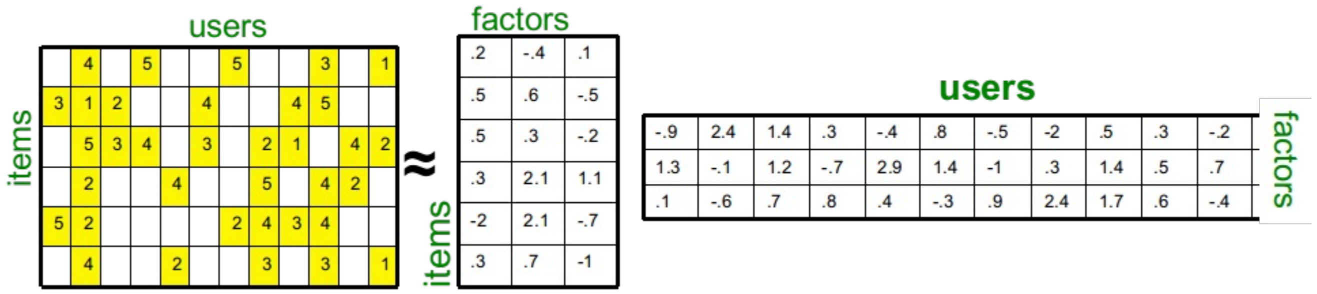
Matrix Factorization

Any $m \times n$ matrix R of rank $k \ll \min\{m, n\}$ can always be expressed in the following product form of rank-k factors:

$$R = UV^T$$

here, U is an $m \times k$ matrix, and V is an $n \times k$ matrix.

The ability to factorize any rank-k matrix in this form is a fundamental fact of linear algebra, and there are an infinite number of such factorizations corresponding to various sets of basis vectors. SVD is one example of such a factorization in which the basis vectors represented by the columns of U (and the columns of V) are orthogonal to one another.



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

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

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

Unconstrained Matrix Factorization

Let the set of all user-item paires (i, j), which are observed in R, be denoted by S.

$$\text{Minimize } J = \frac{1}{2}\sum_{(i,j)\in S} e_{ij}^2 + \frac{\lambda}{2}\sum_{i=1}^{m}\sum_{s=1}^{k} u_{is}^2 + \frac{\lambda}{2}\sum_{j=1}^{n}\sum_{s=1}^{k} v_{js}^2$$

$$= \frac{1}{2}\sum_{(i,j)\in S}\left(r_{ij} - \sum_{s=1}^{k} u_{is}\cdot v_{js}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{m}\sum_{s=1}^{k} u_{is}^2 + \frac{\lambda}{2}\sum_{j=1}^{n}\sum_{s=1}^{k} v_{js}^2$$

subject to: No constraints on U and V

$$\frac{\partial J}{\partial u_{iq}} = \sum_{j:(i,j)\in S}\left(r_{ij} - \sum_{s=1}^{k} u_{is}\cdot v_{js}\right)(-v_{jq}) + \lambda u_{iq} = \sum_{j:(i,j)\in S}(e_{ij})(-v_{jq}) + \lambda u_{iq}$$

$$\forall j \in \{1 \ldots m\}, q \in \{1 \ldots k\}$$

$$\frac{\partial J}{\partial v_{jq}} = \sum_{i:(i,j)\in S}\left(r_{ij} - \sum_{s=1}^{k} u_{is}\cdot v_{js}\right)(-u_{iq}) + \lambda v_{jq} = \sum_{i:(i,j)\in S}(e_{ij})(-u_{iq}) + \lambda v_{jq}$$

$$j \in \{1 \ldots n\}, q \in \{1 \ldots k\}$$

$$u_{iq} \Leftarrow u_{iq} + \alpha\frac{\partial J}{\partial u_{iq}}, v_{jq} \Leftarrow v_{jq} + \alpha\frac{\partial J}{\partial v_{jq}}$$

SVD and NMF

$$\text{Minimize}\, J = \frac{1}{2}\sum_{(i,j)\in S} e_{ij}^2 + \frac{\lambda}{2}\sum_{i=1}^{m}\sum_{s=1}^{k} u_{is}^2 + \frac{\lambda}{2}\sum_{j=1}^{n}\sum_{s=1}^{k} v_{js}^2$$

$$= \frac{1}{2}\sum_{(i,j)\in S}\left(r_{ij} - \sum_{s=1}^{k} u_{is}\cdot v_{js}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{m}\sum_{s=1}^{k} u_{is}^2 + \frac{\lambda}{2}\sum_{j=1}^{n}\sum_{s=1}^{k} v_{js}^2$$

subject to:

Columns of U are mutually orthogonal

Columns of V are mutually orthogonal

Non negative Matrix Factorization

subject to:

$U \geqslant 0$

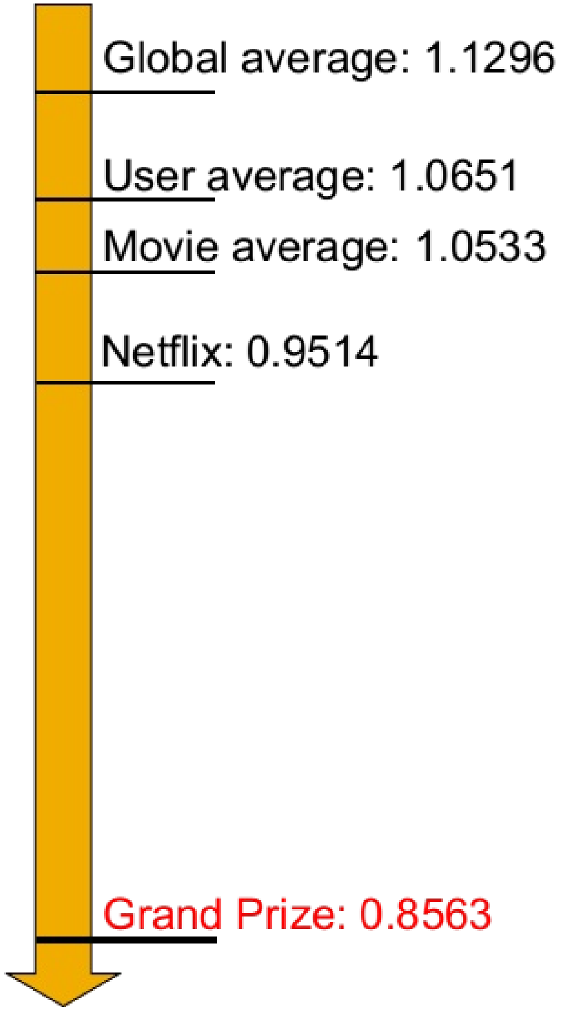
$V \geqslant 0$

Netflix Data Challenge

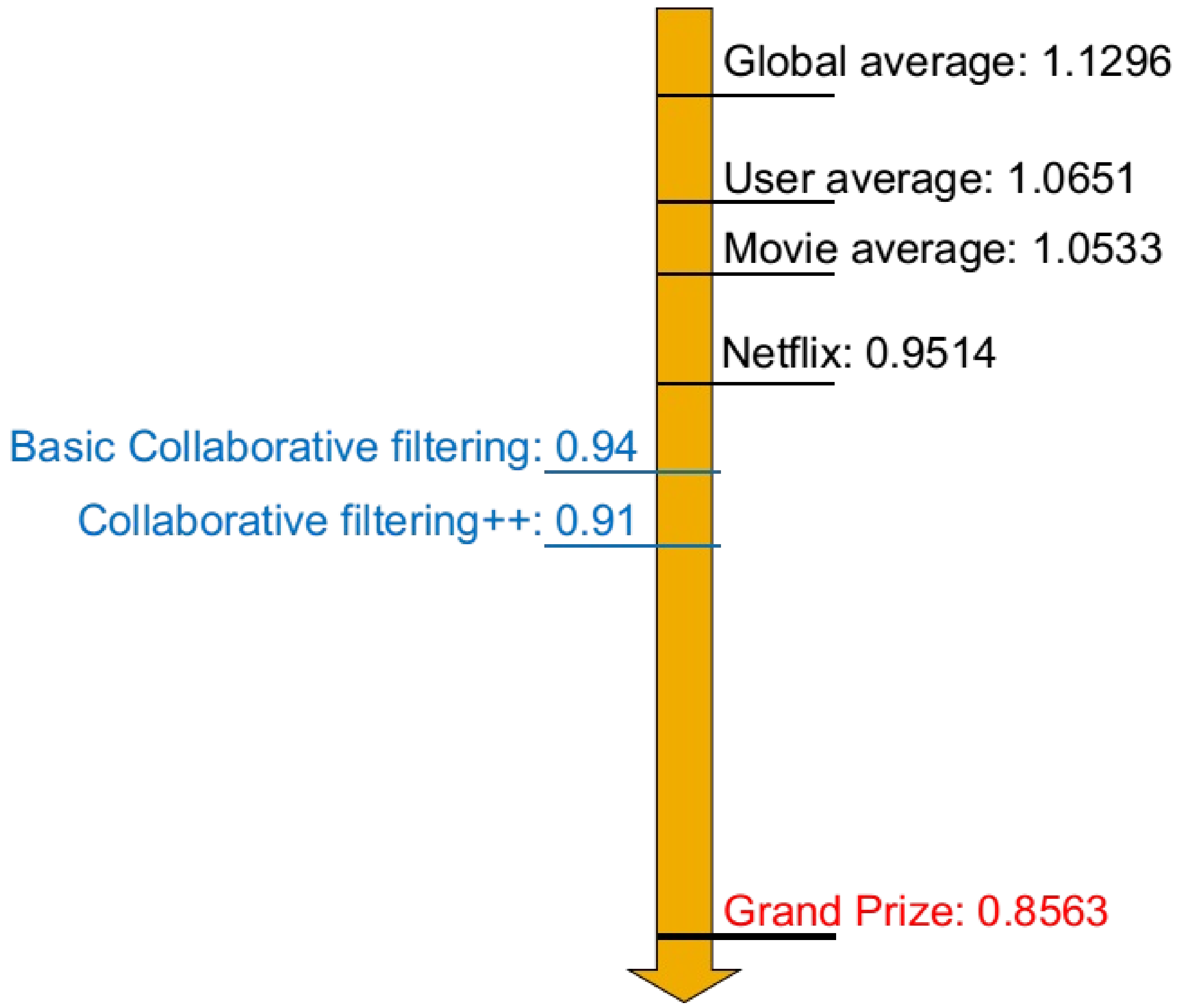
- Training data
  - 100 million ratings, 480,000 users, 17,770 movies
  - 6 years of data: 2000-2005
- Test data
  - Last few ratings of each user (2.8 million)
  - **Evaluation criterion:** Root Mean Square Error (RMSE)

    $$= \frac{1}{|R|} \sqrt{\sum_{(i,x)\in R} (\hat{r}_{xi} - r_{xi})^2}$$
  - **Netflix's system RMSE: 0.9514**
- Competition
  - 2,700+ teams
  - **$1 million** prize for 10% improvement on Netflix

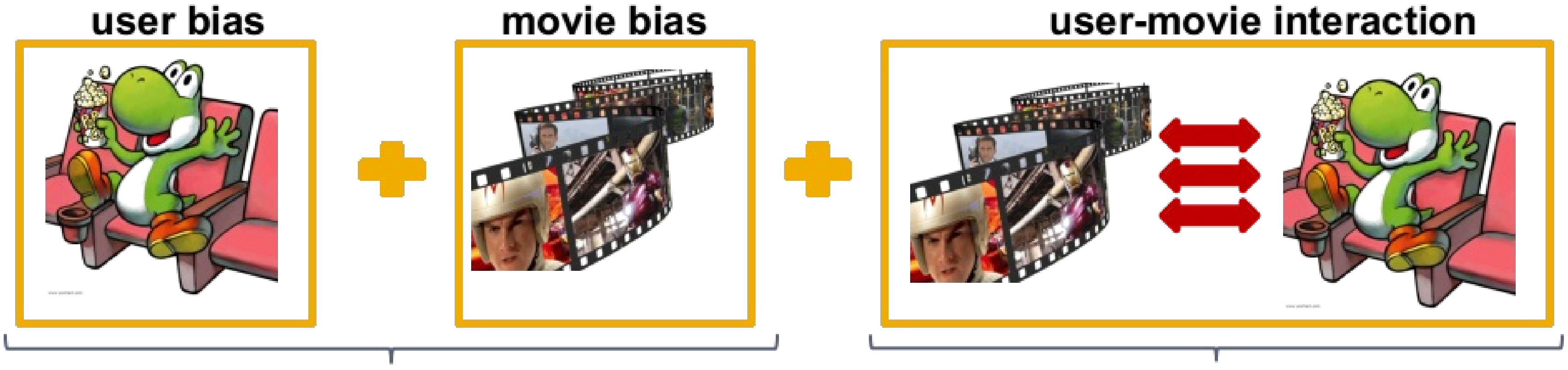
Starting Point



Basic Collaborative Filtering



Modeling Biases and Interactions



Baseline predictor

- Separates users and movies
- Benefits from insights into user's 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

- $\boldsymbol{\mu}$ = overall mean rating
- $\boldsymbol{b_x}$ = bias of user $\boldsymbol{x}$
- $\boldsymbol{b_i}$ = bias of movie $\boldsymbol{i}$

Modeling Biases and Interactions

$$r_{xi} = \underbrace{\mu}_{\text{Overall mean rating}} + \underbrace{b_x}_{\text{Bias for user } x} + \underbrace{b_i}_{\text{Bias for movie } i} + \underbrace{q_i \cdot p_x}_{\text{User-Movie interaction}}$$

- **Example:**
  - Mean rating: μ = ***3.7***
  - You are a critical reviewer: your ratings are 1 star lower than the mean: b_x = ***-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

Modeling Biases and Interactions

- **Solve:**

$$\min_{Q,P} \sum_{(x,i)\in R} \Big(r_{xi} - (\mu + b_x + b_i + q_i\, p_x) \Big)^2$$

goodness of fit

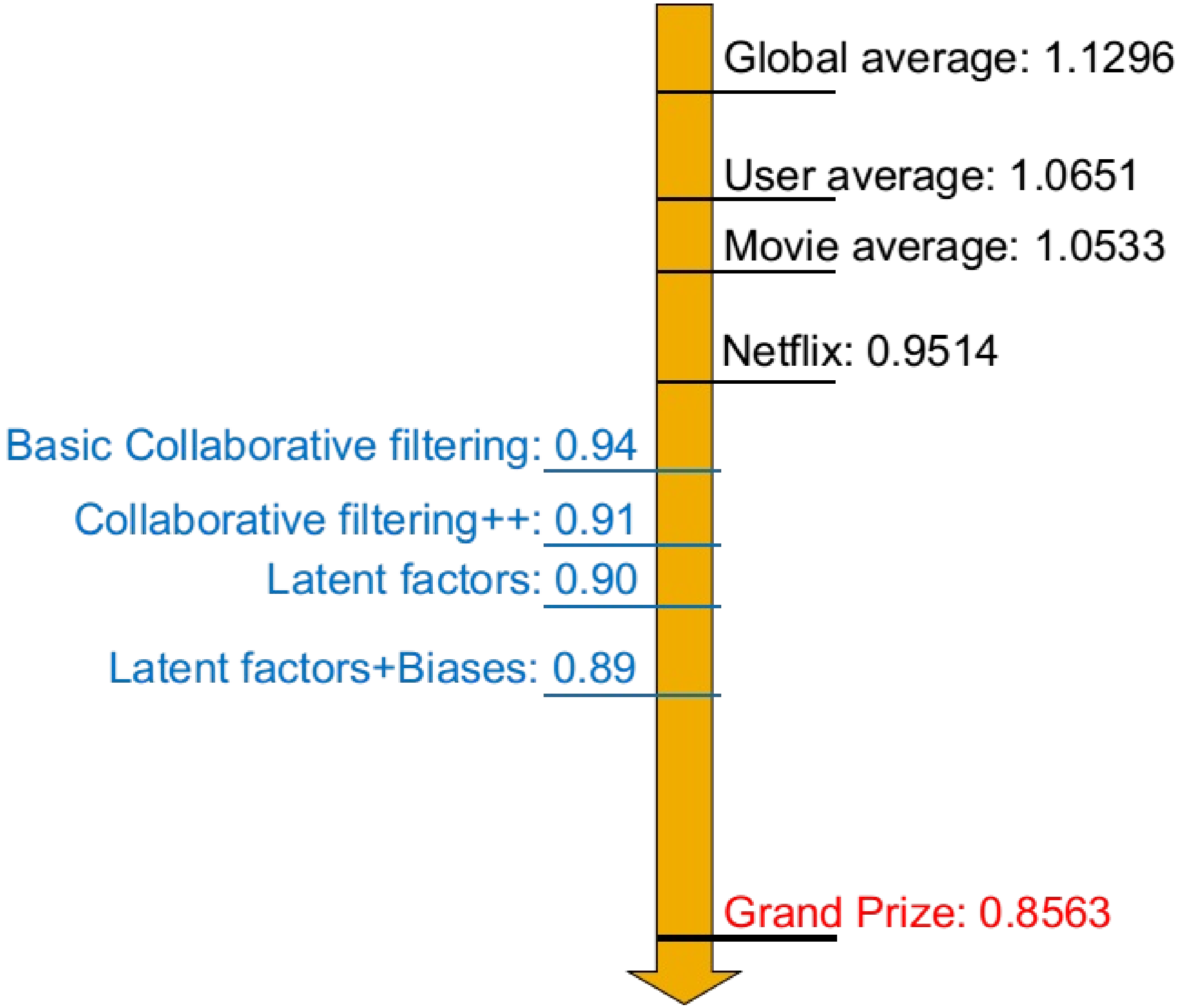
$$+ \left(\lambda_1 \sum_i \|q_i\|^2 + \lambda_2 \sum_x \|p_x\|^2 + \lambda_3 \sum_x \|b_x\|^2 + \lambda_4 \sum_i \|b_i\|^2 \right)$$

regularization

λ is selected via grid-search on a validation set

- **Stochastic gradient decent to find parameters**
  - **Note:** Both biases ***b***$\boldsymbol{b_x}$, $\boldsymbol{b_i}$ as well as interactions $\boldsymbol{q_i}$, $\boldsymbol{p_x}$ are treated as parameters (we estimate them)

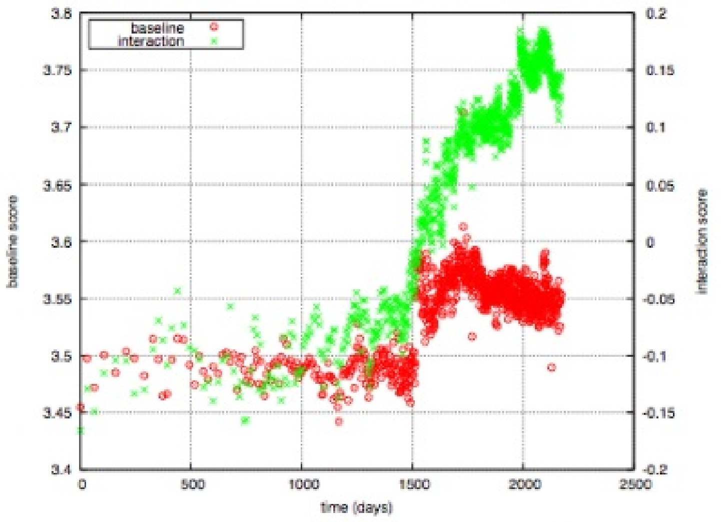
Latent factors + Baseline

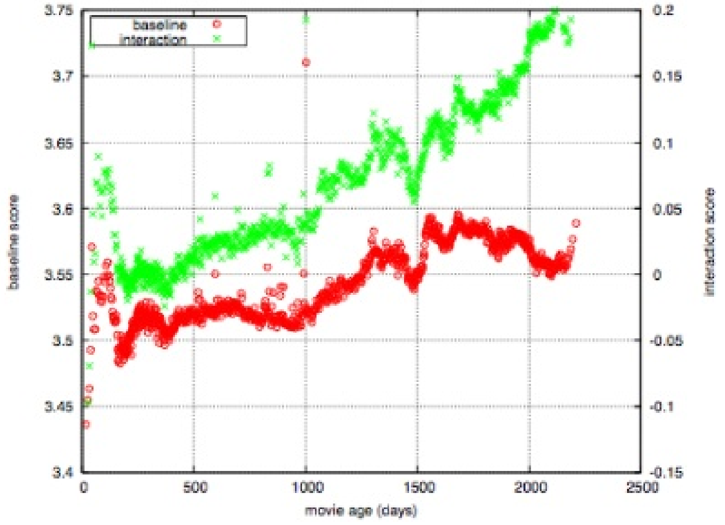


Temporal Biases

- **Sudden rise in the average movie rating** (early 2004)
  - Improvements in Netflix
  - GUI improvements
  - Meaning of rating changed
- **Movie age**
  - Users prefer new movies without any reasons
  - Older movies are just inherently better than newer ones

Y. Koren, Collaborative filtering with temporal dynamics, KDD '09





Temporal Biases

- **Original model:**

  $$r_{xi} = \mu + b_x + b_i + q_i \cdot p_x$$

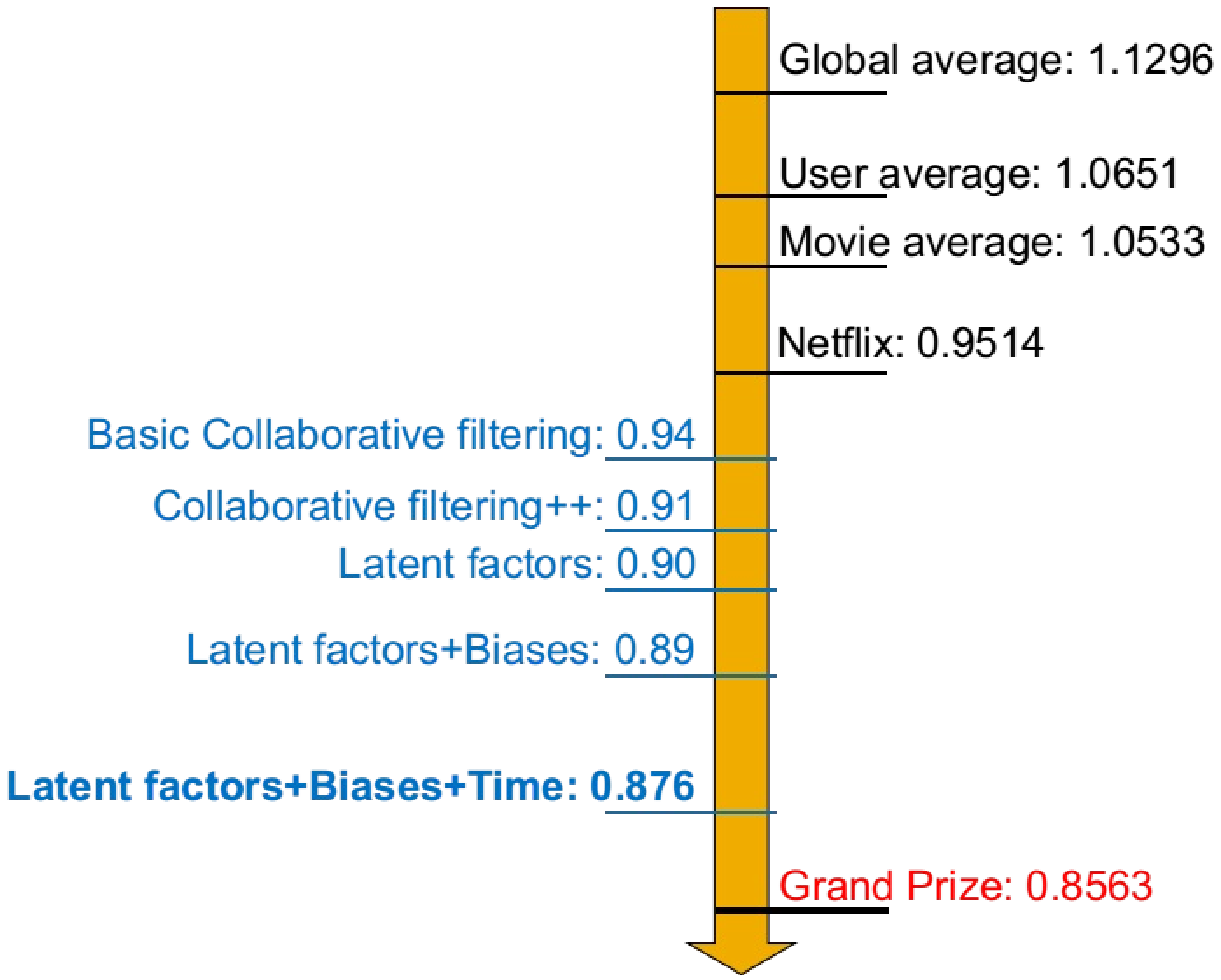
- **Add time dependence to biases:**

  $$r_{xi} = \mu + b_x(t) + b_i(t) + q_i \cdot p_x$$

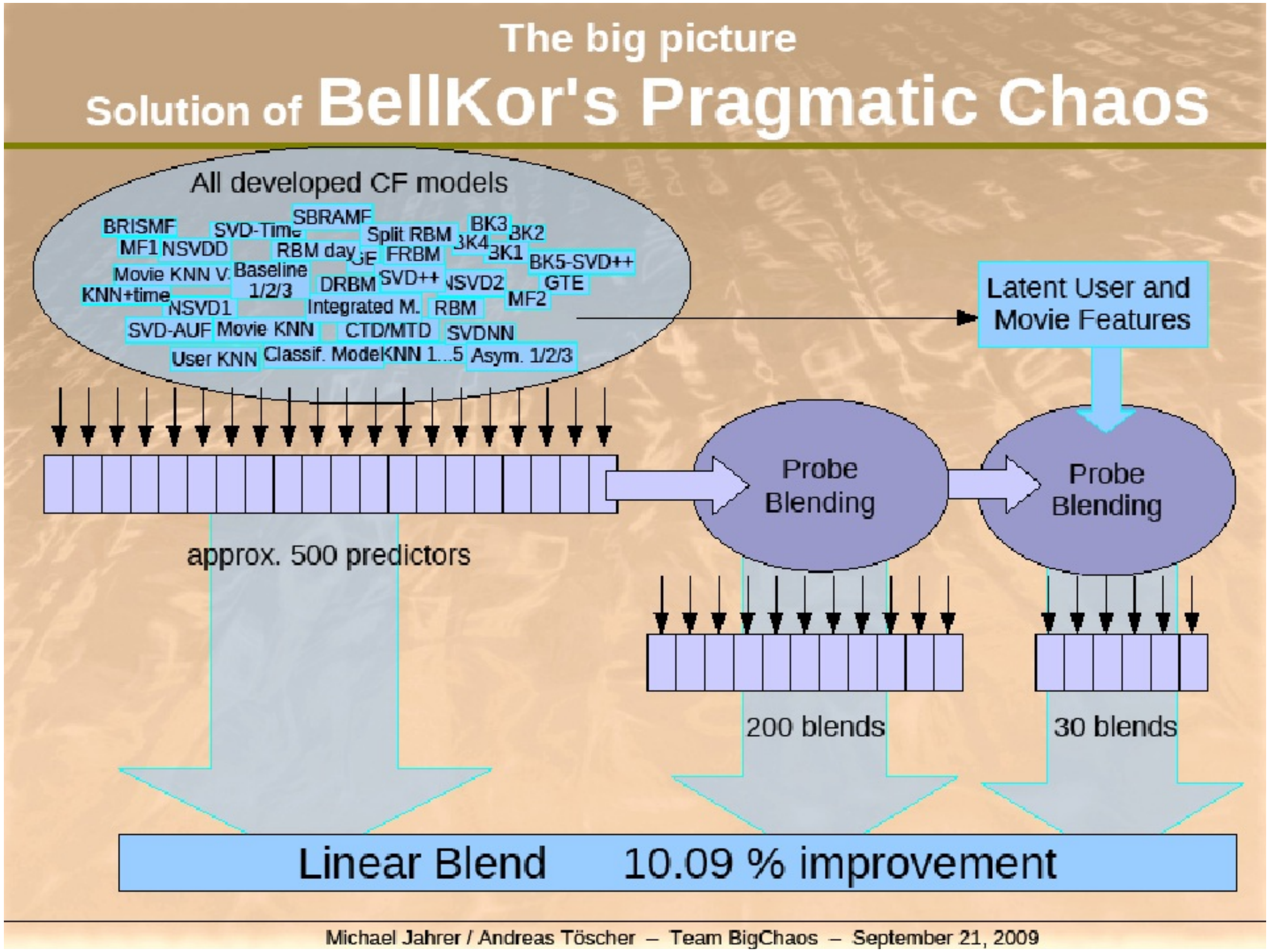
  - Make parameters ***b***$\boldsymbol{b_x}$ and $\boldsymbol{b_i}$ to depend on time
  - **(1)** Parameterize time-dependence by linear trends
    (2) Each bin corresponds to 10 consecutive weeks

- **Add temporal dependence to factors**
  - $\boldsymbol{p_x(t)}$… user preference vector on day $\boldsymbol{t}$

Temporal Biases

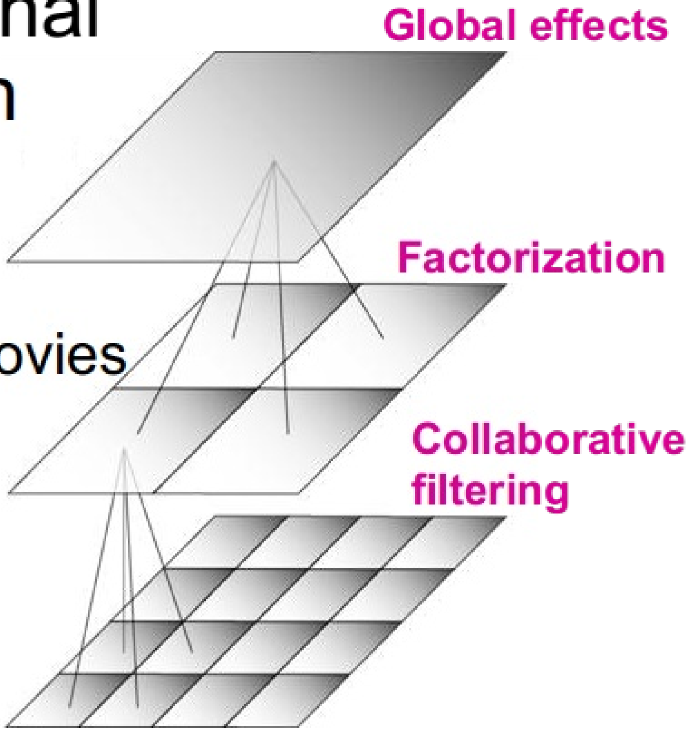


Kitchen Sink Approach

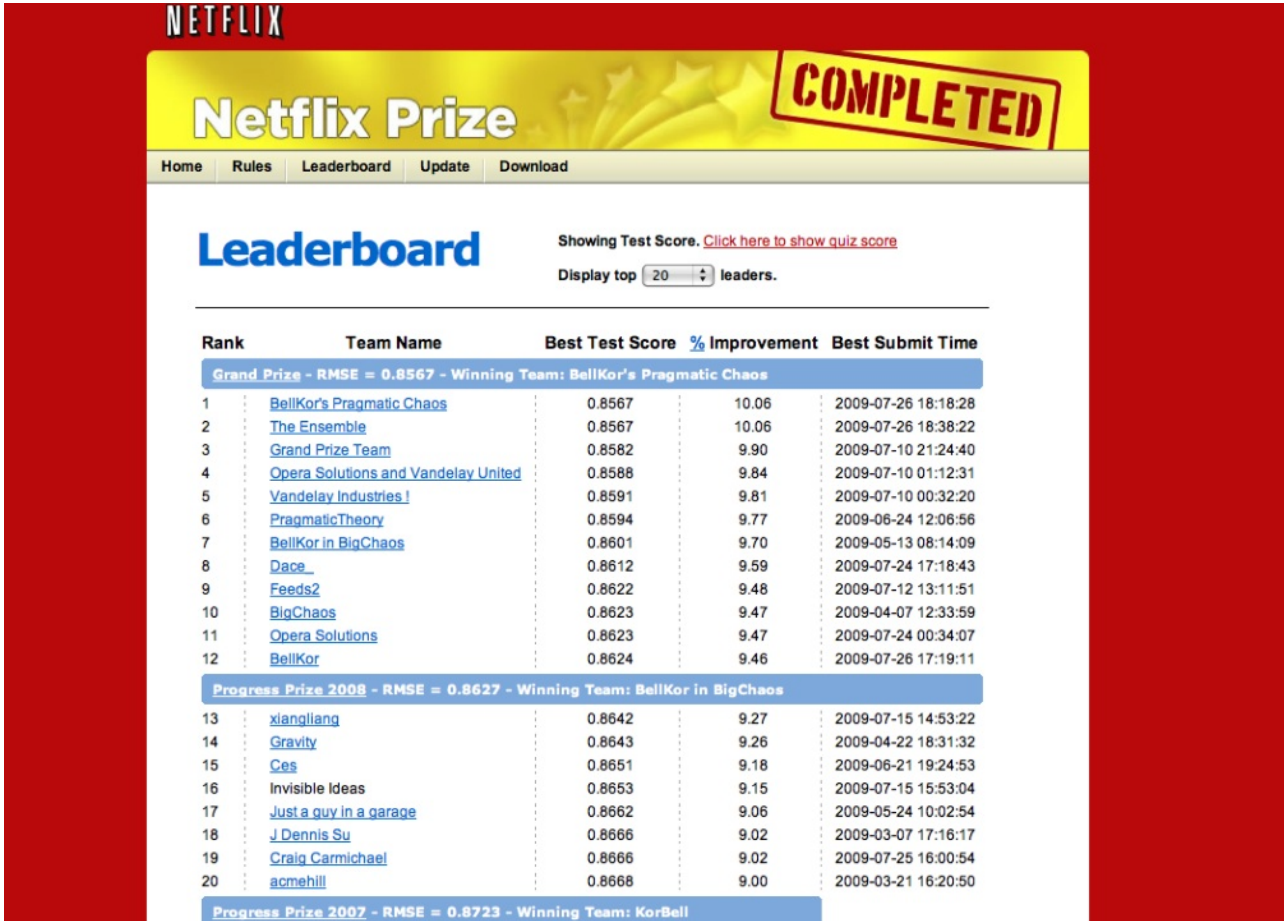


Explication

- The winner of the Netflix Challenge!
- Multi-scale modeling of the data:
  Combine top level, "regional" modeling of the data, with a refined, local view:
  - Global:
    - Overall deviations of users/movies
  - Factorization:
    - Addressing "regional" effects
  - Collaborative filtering:
    - Extract local patterns



Leaderboard



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	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
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 $ Awarded Sept 21st 2009

References

Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). *Mining of massive datasets*. Cambridge university press. http://www.mmds.org

Abu-Mostafa, Y. S., Magdon-Ismail, M., & Lin, H. T. (2012). *Learning from data* (Vol. 4). New York, NY, USA:: AMLBook.

Kantor, P. B. (2015). *Recommender systems handbook*. F. Ricci, L. Rokach, & B. Shapira (Eds.). Berlin, Germany:: Springer.

Aggarwal, C. C. (2016). *Recommender systems*. Springer International Publishing.

http://surprise.readthedocs.io/en/stable/index.html