movie #15868

Lessons from the Netflix Prize

Yehuda Koren
The **BellKor** team
(with Robert Bell & Chris Volinsky)

at&t

movie #7614

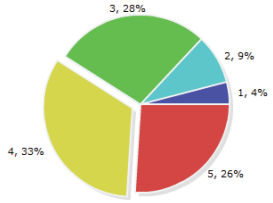
movie #3250

Recommender systems

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 grand prize for 10% improvement on Cinematch result
  - $50,000 2007 progress prize for 8.43% improvement

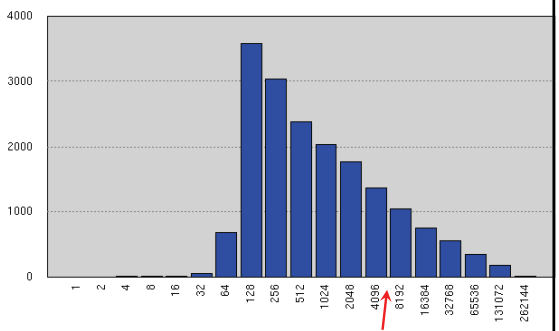
Overall rating distribution



- Third of ratings are 4s
- Average rating is 3.68

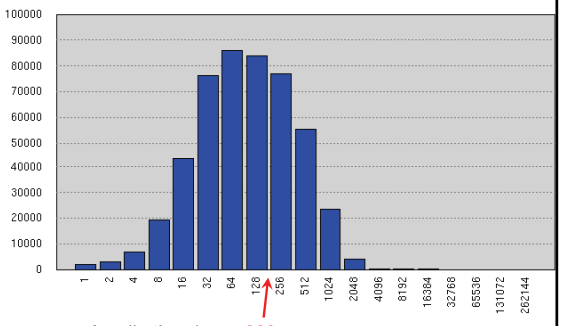
From TimelyDevelopment.com

#ratings per movie



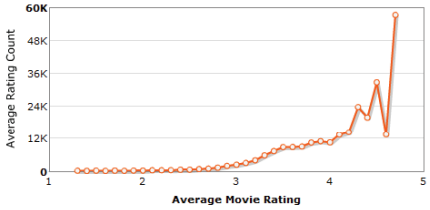
- Avg #ratings/movie: 5627

#ratings per user



- Avg #ratings/user: 208

Average movie rating by movie count



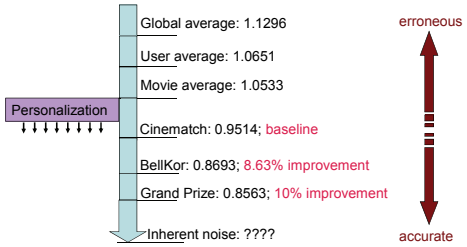
- More ratings to better movies

From TimelyDevelopment.com

Most loved movies

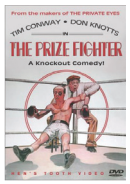
Title	Avg rating	Count
The Shawshank Redemption	4.593	137812
Lord of the Rings: The Return of the King	4.545	133597
The Green Mile	4.306	180883
Lord of the Rings: The Two Towers	4.460	150676
Finding Nemo	4.415	139050
Raiders of the Lost Ark	4.504	117456
Forrest Gump	4.299	180736
Lord of the Rings: The Fellowship of the ring	4.433	147932
The Sixth Sense	4.325	149199
Indiana Jones and the Last Crusade	4.333	144027

Important RMSEs



Challenges

- Size of data
  - Scalability
  - Keeping data in memory
- Missing data
  - 99 percent missing
  - Very imbalanced
- Avoiding overfitting
- Test and training data differ significantly



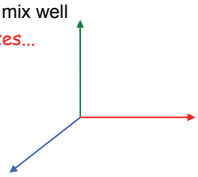
movie #16322

The BellKor recommender system

- Use an ensemble of complementing predictors
- Two, half tuned models worth more than a single, fully tuned model

The BellKor recommender system

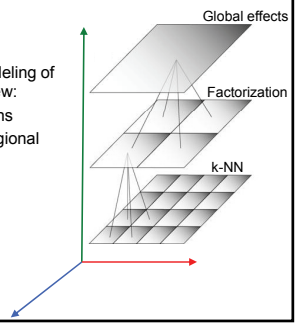
- Use an ensemble of complementing predictors
- Two, half tuned models worth more than a single, fully tuned model
- But:
  Many seemingly different models expose similar characteristics of the data, and won't mix well
- Concentrate efforts along three axes…



The three dimensions of the BellKor system

The first axis:

- Multi-scale modeling of the data
- Combine top level, regional modeling of the data, with a refined, local view:
  - k-NN: Extracting local patterns
  - Factorization: Addressing regional effects



Multi-scale modeling – 1st tier

Global effects:

- Mean rating: 3.7 stars
- *The Sixth Sense* is 0.5 stars above avg
- *Joe* rates 0.2 stars below avg

➔Baseline estimation:
Joe will rate *The Sixth Sense* 4 stars

Multi-scale modeling – 2nd tier

Factors model:

- Both *The Sixth Sense* and *Joe* are placed high on the "Supernatural Thrillers" scale

➔Adjusted estimate:
Joe will rate *The Sixth Sense* 4.5 stars

Multi-scale modeling – 3rd tier

Neighborhood model:

- *Joe* didn't like related movie *Signs*
- ➔Final estimate:
  Joe will rate *The Sixth Sense* 4.2 stars

Important technique: Shrinking

- Data is very sparse, and many parameters need to be estimated with small number of observations
- Shrinking: (in estimating s with n observations)

$$s' = \frac{ns}{n+\beta}$$

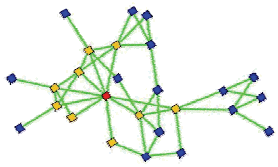
- Can be justified using a Bayesian approach

First step: removing global effects

- Some users systematically give higher scores.
- Some movies have better ratings on average (simply good movies.)
- Take into account time effects (not covered in detail):
  - Movies may go out of fashion
  - People may change their tastes

Local level: 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)



k-NN

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

k-NN

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

k-NN

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

k-NN

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

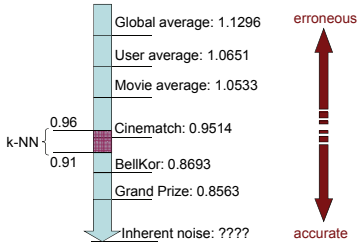
k-NN

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

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}

k-NN - Common practice

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

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

Problems:

1. Similarity measures are arbitrary; no fundamental justification
2. Pairwise similarities isolate each neighbor; neglect interdependencies among neighbors
3. Taking a weighted average is restricting; e.g., when neighborhood information is limited

Their approach: solving an optimization problem

- Use a weighted sum rather than a weighted average:

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

(We 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:

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

Interpolation weights

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

Mostly unknown

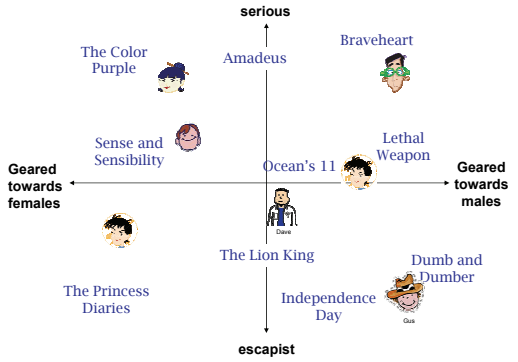
- Interpolation weights derived based on their role; no use of an arbitrary similarity measure
- Explicitly account for interrelationships among the neighbors

Challenges:

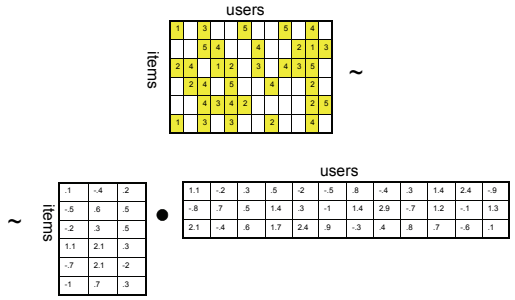
- Deal with missing values
- Avoid overfitting
- Efficient implementation

⇒ Estimate inner-products among movie ratings

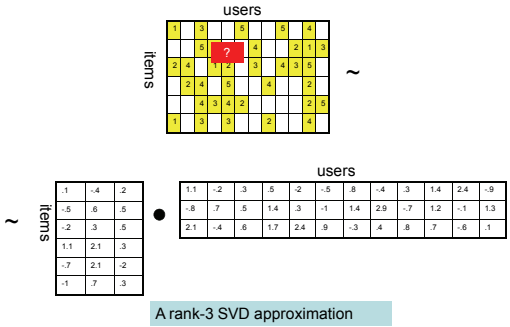
Regional level: Latent factor models



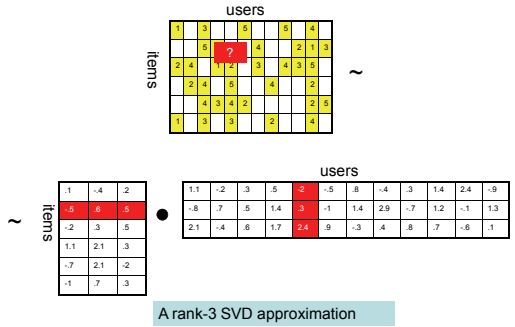
Latent factor models



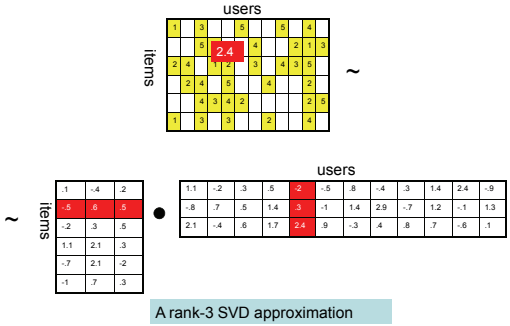
Estimate unknown ratings as inner-products of factors:



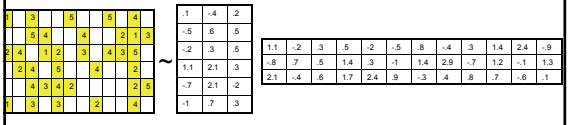
Estimate unknown ratings as inner-products of factors:



Estimate unknown ratings as inner-products of factors:



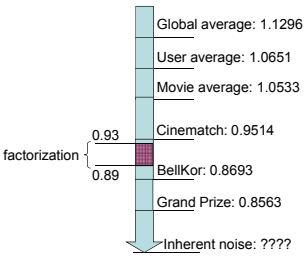
Latent factor models



Properties:

- SVD isn't defined when entries are unknown ➔ use specialized methods
- Very powerful model ➔ can easily overfit, sensitive to regularization
- Probably most popular model among contestants
  - 12/11/2006: Simon Funk describes an SVD based method
  - 12/29/2006: Free implementation at timelydevelopment.com

Factorization on the RMSE scale



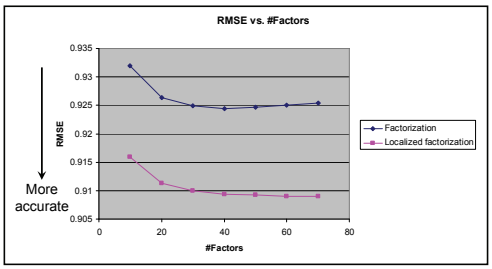
Our approach

- User factors:
  Model a user u as a vector $p_u \sim N_k(\mu, \Sigma)$
- Movie factors:
  Model a movie i as a vector $q_i \sim N_k(\gamma, \Lambda)$
- Ratings:
  Measure "agreement" between u and i: $r_{ui} \sim N(p_u^T q_i, \varepsilon^2)$
- Maximize model's likelihood:
  - Alternate between recomputing user-factors, movie-factors and model parameters
  - Special cases:
    - Alternating Ridge regression
    - Nonnegative matrix factorization

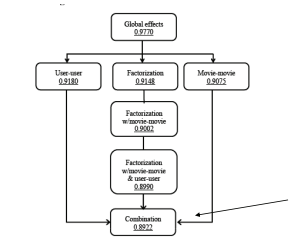
Localized factorization model

- Standard factorization:
  User u is a linear function parameterized by p_u
  $r_{ui} = p_u^T q_i$
- Allow user factors – p_u – to depend on the item being predicted
  $r_{ui} = p_u(i)^T q_i$
- Vector $p_u(i)$ models behavior of u on items like i

Results on Netflix Probe set



Combining multi-scale views



Combination using confidence scores

Figure 4: Our results (measured by RMSE) on the Quiz set. For comparison, the Netflix Cinematch reported result is RMSE=0.9514.

Confidence Scores

- All algorithms presented are based on optimizing some objective
  - Can use the objective value as measure of confidence
- Use these confidence scores when combining results of different models

Item-item suggestions are very good when confidence is high

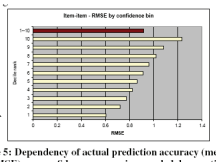
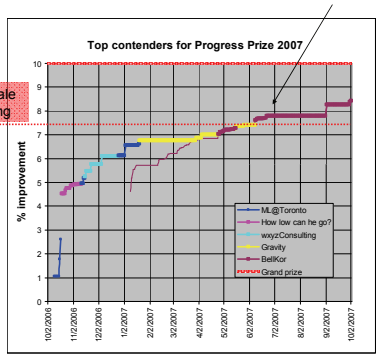


Figure 5: Dependency of actual prediction accuracy (measured by RMSE) on confidence scores is revealed by partitioning Probe results into ten deciles according to confidence scores

Later work (not in this paper)

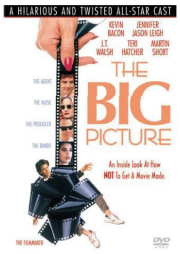
multi-scale modeling



The third dimension of the BellKor system

Great news to recommender systems:

- Works even better on real life datasets (?)
- Improve accuracy by exploiting implicit feedback
- Implicit behavior is abundant and easy to collect:
  - Rental history
  - Search patterns
  - Browsing history
  - ...
- Implicit feedback allows predicting personalized ratings for users that never rated!



movie #17270

Later work (not in this paper)

Class comments

- Interesting that combining different algorithms improves performance.
- Add item content information?
- Good paper; learnt a lot
- Alternative approach to SVD: Gaussian process regression models to model non-linearity?
- Small improvement after a lot of work.