Recommender Systems: Latent Factor Models

CS547 Machine Learning for Big Data
Tim Althoff

PAUL G. ALLEN SCHOOL OF COMPUTER SCIENCE & ENGINEERING
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 Square Error (RMSE) = \( \sqrt{\frac{1}{|R|} \sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2} \)
  - Netflix’s system RMSE: 0.9514

- **Competition**
  - 2,700+ teams
  - **$1 million** prize for 10% improvement on Netflix
Competition Structure

Training Data

Labels known publicly

100 million ratings

Held-Out Data

Labels only known to Netflix

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

4/20/2020
The Netflix Utility Matrix $R$

Matrix $R$

480,000 users

17,700 movies
Utility Matrix $R$: Evaluation

Matrix $R$

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

480,000 users

17,700 movies

Training Data Set

Test Data Set

Predicted rating

True rating of user $x$ on item $i$
BellKor Recommender System

- 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
Global:

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

  \[ \Rightarrow \text{Baseline estimation:} \]
  
  *Joe* will rate *The Sixth Sense* 4 stars

  \[ \text{That is } 4 = 3.7 + 0.5 - 0.2 \]

Local neighborhood (CF/NN):

- *Joe* didn’t like related movie *Signs*

  \[ \Rightarrow \text{Final estimate:} \]
  
  *Joe* will rate *The Sixth Sense* 3.8 stars
The earliest and the most popular collaborative filtering method
Derive unknown ratings from those of “similar” movies (item-item variant)
Define similarity metric $s_{ij}$ of items $i$ and $j$
Select $k$-nearest neighbors, compute the rating

**$N(i; x)$**: items most similar to $i$ that were rated by $x$

$$\hat{r}_{xi} = \frac{\sum_{j \in N(i; x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i; x)} s_{ij}}$$

$s_{ij}$... similarity of items $i$ and $j$
$r_{xj}$... rating of user $x$ on item $j$
$N(i; x)$... set of items similar to item $i$ that were rated by $x$
Modeling Local & Global Effects

- In practice we get better estimates if we model deviations:

\[
\hat{r}_{xi} = b_{xi} + \frac{\sum_{j \in N(i; x)} S_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i; x)} S_{ij}}
\]

Baseline estimate for \( r_{xi} \)

- \( b_{xi} = \mu + b_x + b_i \)

\( \mu \) = overall mean rating

\( b_x \) = rating deviation of user \( x \)

\( = (\text{avg. rating of user } x) - \mu \)

\( b_i \) = (avg. rating of movie \( i \)) - \( \mu \)

Problems/Issues:

1) Similarity metrics are “arbitrary”
2) Pairwise similarities neglect interdependencies among users
3) Taking a weighted average can be restricting

Solution: Instead of \( s_{ij} \) use \( w_{ij} \) that we estimate directly from data
Idea: Interpolation Weights $w_{ij}$

- Use a **weighted sum** rather than **weighted avg.**:

$$
\hat{r}_{xi} = b_{xi} + \sum_{j \in N(i;x)} w_{ij} (r_{xj} - b_{xj})
$$

- **A few notes:**
  - $N(i; x)$ ... set of movies rated by user $x$ that are similar to movie $i$
  - $w_{ij}$ is the **interpolation weight** (some real number)
    - Note, we allow: $\sum_{j \in N(i;x)} w_{ij} \neq 1$
  - $w_{ij}$ models interaction between pairs of movies (it does not depend on user $x$)
Idea: Interpolation Weights $w_{ij}$

- $\hat{r}_{xi} = b_{xi} + \sum_{j \in N(i,x)} w_{ij} (r_{xj} - b_{xj})$
- How to set $w_{ij}$?
  
  - Remember, error metric is:  
    $$\frac{1}{|R|} \sqrt{\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2}$$
  
  or equivalently $\text{SSE:} \sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2$
  
  - Find $w_{ij}$ that minimize $\text{SSE}$ on training data!
    - Models relationships between item $i$ and its neighbors $j$
    - $w_{ij}$ can be learned/estimated based on $x$ and all other users that rated $i$

Why is this a good idea?
Goal: Make good recommendations

- Quantify goodness using **RMSE**: Lower RMSE $\Rightarrow$ better recommendations
- Really want to make good recommendations on items that user has not yet seen. *Can’t really do this!*

Let’s set build a system such that it works well on known (user, item) ratings
And **hope** the system will also predict well the unknown ratings
Recommendations via Optimization

- **Idea:** Let’s set values $w$ such that they work well on known (user, item) ratings
- **How to find such values $w$?**
  - **Idea:** Define an objective function and solve the optimization problem
  - Find $w_{ij}$ that minimize $SSE$ on training data!

$$J(w) = \sum_{x,i \in R} \left( \left[ b_{xi} + \sum_{j \in N(i;x)} w_{ij} (r_{xj} - b_{xj}) \right] - r_{xi} \right)^2$$

- Think of $w$ as a vector of numbers
Detour: Minimizing a function

- A simple way to minimize a function $f(x)$:
  - Compute the derivative $\nabla f(x)$
  - Start at some point $y$ and evaluate $\nabla f(y)$
  - Make a step in the reverse direction of the gradient: $y = y - \nabla f(y)$
  - Repeat until convergence
Interpolation Weights

- We have the optimization problem, now what?
- Gradient descent:
  - Iterate until convergence: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} J$  
    
    $\eta$ ... learning rate

  where $\nabla_{\mathbf{w}} J$ is the gradient (derivative evaluated on data):

  $\nabla_{\mathbf{w}} J = \left[ \frac{\partial J(\mathbf{w})}{\partial w_{ij}} \right] = 2 \sum_{x,i \in R} \left( b_{xi} + \sum_{k \in N(i; x)} w_{ik} (r_{xk} - b_{xk}) \right) (r_{xj} - b_{xj}) - r_{xi}$

  for $j \in \{N(i; x), \forall i, \forall x\}$

  else $\frac{\partial J(\mathbf{w})}{\partial w_{ij}} = 0$

  - Note: We fix movie $i$, go over all $r_{xi}$, for every movie $j \in N(i; x)$, we compute $\frac{\partial J(\mathbf{w})}{\partial w_{ij}}$

  **while** $|\mathbf{w}_{new} - \mathbf{w}_{old}| > \varepsilon$:

  $\mathbf{w}_{old} = \mathbf{w}_{new}$

  $\mathbf{w}_{new} = \mathbf{w}_{old} - \eta \cdot \nabla_{\mathbf{w}} \mathbf{w}_{old}$
So far: $\hat{r}_{xi} = b_{xi} + \sum_{j \in N(i; x)} w_{ij} (r_{xj} - b_{xj})$

- Weights $w_{ij}$ derived based on their roles; **no use of an arbitrary similarity metric** ($w_{ij} \neq s_{ij}$)
- Explicitly account for interrelationships among the neighboring movies

Next: Latent factor model
- Extract “regional” correlations
Performance of Various Methods

- Grand Prize: 0.8563
- Netflix: 0.9514
- Movie average: 1.0533
- User average: 1.0651
- Global average: 1.1296

Basic Collaborative filtering: 0.94
CF+Biases+learned weights: 0.91

Grand Prize: 0.8563
Latent Factor Models (e.g., SVD)

- The Color Purple
- The Princess Diaries
- Sense and Sensibility
- Geared towards females

- Serious Amadeus
- Ocean’s 11
- Funny

- Braveheart
- Lethal Weapon
- Dumb and Dumber
- Geared towards males

[Slide from BellKor team]
Latent Factor Models

- “SVD” on Netflix data: $R \approx Q \cdot P^T$

- For now let’s assume we can approximate the rating matrix $R$ as a product of “thin” $Q \cdot P^T$
  - $R$ has missing entries but let’s ignore that for now!
    - Basically, we want the reconstruction error to be small on known ratings and we don’t care about the values on the missing ones
Ratings as Products of Factors

How to estimate the missing rating of user $x$ for item $i$?

$$\hat{r}_{xi} = q_i \cdot p_x$$

$$\approx \sum_f q_{if} \cdot p_{xf}$$

$q_i$ = row $i$ of $Q$
$p_x$ = column $x$ of $P^T$
Ratings as Products of Factors

How to estimate the missing rating of user $x$ for item $i$?

\[ \hat{r}_{xi} = q_i \cdot p_x \]

\[ = \sum_f q_{if} \cdot p_{xf} \]

$q_i = \text{row } i \text{ of } Q$

$p_x = \text{column } x \text{ of } P^T$
Ratings as Products of Factors

- How to estimate the missing rating of user $x$ for item $i$?

$$\hat{r}_{xi} = q_i \cdot p_x = \sum_f q_{if} \cdot p_{xf}$$

$q_i = \text{row } i \text{ of } Q$

$p_x = \text{column } x \text{ of } P^T$
Latent Factor Models

The Color Purple
Sense and Sensibility
The Princess Diaries
The Lion King
Ocean's 11
Braveheart
Lethal Weapon
Dumb and Dumber
Sense and Sensibility
Geared towards females
Serious
Amadeus
Funny
Factor 1
Geared towards males
Factor 2
Independence Day
4/20/2020
Latent Factor Models

The Color Purple

Sense and Sensibility

Geared towards females

Serious Amadeus

Ocean’s 11

Factor 1

Geared towards males

The Lion King

Factor 2

Funny

Dumb and Dumber

The Princess Diaries

Lethal Weapon

Braveheart

4/20/2020
Recap: SVD

- **Remember SVD:**
  - **A**: Input data matrix
  - **U**: Left singular vecs
  - **V**: Right singular vecs
  - **Σ**: Singular values

- **So in our case:**
  - "SVD" on Netflix data: \( R \approx Q \cdot P^T \)
  - \( A = R, \ Q = U, \ P^T = \Sigma V^T \)
  - \( \hat{r}_{xi} = q_i \cdot p_x \)
SVD: More good stuff

- We already know that SVD gives minimum reconstruction error (Sum of Squared Errors):

\[
\min_{U,V,\Sigma} \sum_{i,j \in A} (A_{ij} - [U\Sigma V^T]_{ij})^2
\]

- Note two things:
  - **SSE** and **RMSE** are monotonically related:
    - \(RMSE = \frac{1}{c} \sqrt{SSE}\)  
      Great news: SVD is minimizing RMSE!
  - **Complication**: The sum in SVD error term is over all entries (no-rating is interpreted as zero-rating). But our \(R\) has missing entries!
Latent Factor Models

- SVD isn’t defined when entries are missing!
- Use specialized methods to find $P, Q$

- \[
    \min_{P,Q} \sum_{(i,x) \in R} (r_{xi} - q_i \cdot p_x)^2
    \]
    \[
    \hat{r}_{xi} = q_i \cdot p_x
    \]

- Note:
  - We don’t require cols of $P, Q$ to be orthogonal/unit length
  - $P, Q$ map users/movies to a latent space
  - This was the most popular model among Netflix contestants
Finding the Latent Factors
Latent Factor Models

- Our goal is to find $P$ and $Q$ such that:

$$\min_{P,Q} \sum_{(i,x) \in R} (r_{xi} - q_i \cdot p_x)^2$$
Want to minimize SSE for unseen test data

Idea: Minimize SSE on training data

- Want large $k$ (# of factors) to capture all the signals
- But, SSE on test data begins to rise for $k > 2$

This is a classical example of overfitting:

- With too much freedom (too many free parameters) the model starts fitting noise
  - That is, the model fits too well the training data and is thus not generalizing well to unseen test data
Dealing with Missing Entries

- To solve overfitting we introduce regularization:
  - Allow rich model where there is sufficient data
  - Shrink aggressively where data is scarce

\[
\min_{P,Q} \sum_{i \in \text{training}} (r_{xi} - q_i p_x)^2 + \left[ \lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2 \right]
\]

\( \lambda_1, \lambda_2 \ldots \text{user set regularization parameters} \)

**Note:** We do not care about the “raw” value of the objective function, but we care about \( P,Q \) that achieve the minimum of the objective.
The Effect of Regularization

The Color Purple

The Lion King

Ocean's 11

Braveheart

Lethal Weapon

Sense and Sensibility

Geared towards females

Geared towards males

The Princess Diaries

Amadeus

Independence Day

Dumb and Dumber

Factor 1

Factor 2

Geared towards females

serious

funny

min \sum_{i=1}^{n} (x_i - q_i)^2 + \lambda \left( \sum_{i=1}^{m} p_i^2 + \sum_{i=1}^{m} q_i^2 \right)

min factors “error” + \lambda “length”
The Effect of Regularization

<table>
<thead>
<tr>
<th>Serious</th>
<th>Serious</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amadeus</td>
<td>The Color Purple</td>
</tr>
<tr>
<td>Ocean’s 11</td>
<td>The Color Purple</td>
</tr>
<tr>
<td>Lethal Weapon</td>
<td>The Color Purple</td>
</tr>
<tr>
<td>Dumb and Dumber</td>
<td>The Color Purple</td>
</tr>
</tbody>
</table>

Geared towards females

Geared towards males

Sense and Sensibility

Funny

The Princess Diaries

The Lion King

Independence Day

Factor 1

Factor 2

\[
\min_{\mathbf{p}, \mathbf{q}} \sum_{x \in \text{training}} (r_{xi} - q_i p_x)^2 + \lambda \left[ \sum_x \| \mathbf{p}_x \|^2 + \sum_i \| \mathbf{q}_i \|^2 \right]
\]

\[
\min_{\text{factors}} \text{“error”} + \lambda \text{“length”}
\]
The Effect of Regularization

The Color Purple
The Lion King
The Princess Diaries
Sense and Sensibility
serious
Amadeus
Braveheart
Lethal Weapon
Geared towards females
Geared towards males
Ocean's 11
Independence Day
Dumb and Dumber

\[
\min_{p,q} \sum_{i \in \text{training}} (r_i - q_i p_i)^2 + \lambda \left( \sum_{i} \|x_i\|^2 + \sum_{i} \|q_i\|^2 \right)
\]

min factors “error” + \lambda “length”
The Effect of Regularization

\[
\min_{p,q} \sum_{i \in \text{training}} (r_{xi} - q_i p_i)^2 + \lambda \left[ \sum x_i \|p_i\|^2 + \sum \|q_i\|^2 \right]
\]

\[
\min_{\text{factors}} \text{“error”} + \lambda \text{“length”}
\]
Stochastic Gradient Descent

- Want to find matrices $P$ and $Q$:

$$\min_{P,Q} \sum_{\text{training}} (r_{xi} - q_i p_x)^2 + \left[ \lambda_1 \sum_x \| p_x \|^2 + \lambda_2 \sum_i \| q_i \|^2 \right]$$

- Gradient descent:
  - Initialize $P$ and $Q$ (using SVD, pretend missing ratings are 0)
  - Do gradient descent:
    - $P \leftarrow P - \eta \cdot \nabla P$
    - $Q \leftarrow Q - \eta \cdot \nabla Q$
    - where $\nabla Q$ is gradient/derivative of matrix $Q$:
      $\nabla Q = [\nabla q_{if}]$ and $\nabla q_{if} = \sum_{x,i} -2(r_{xi} - q_i p_x) p_{xf} + 2\lambda_2 q_{if}$
      - Here $q_{if}$ is entry $f$ of row $q_i$ of matrix $Q$
  - Observation: Computing gradients is slow!
Stochastic Gradient Descent

- **Gradient Descent (GD) vs. Stochastic GD**
  - **Observation:** \( \nabla Q = [\nabla q_{if}] \) where
    \[
    \nabla q_{if} = \sum_{x,i} -2(r_{xi} - q_{if}p_{xf})p_{xf} + 2\lambda q_{if} = \sum_{x,i} \nabla Q(r_{xi})
    \]
  - Here \( q_{if} \) is entry \( f \) of row \( q_i \) of matrix \( Q \)
  - **Q** ← **Q** − \( \eta \nabla Q = Q − \eta[\sum_{x,i} \nabla Q(r_{xi})] \)
  - **Idea:** Instead of evaluating gradient over all ratings evaluate it for each individual rating and make a step

- **GD:** \( Q ← Q − \eta[\sum r_{xi} \nabla Q(r_{xi})] \)
- **SGD:** \( Q ← Q − \mu \nabla Q(r_{xi}) \)
  - **Faster convergence!**
    - Need more steps but each step is computed much faster
Convergence of GD vs. SGD

- GD improves the value of the objective function at every step.
- SGD improves the value but in a “noisy” way.
- GD takes fewer steps to converge but each step takes much longer to compute.
- In practice, SGD is much faster!
Extending Latent Factor Model to Include Biases
**Modeling Biases and Interactions**

- **User bias**: $\mu = \text{overall mean rating}$
- **Movie bias**: $b_x = \text{bias of user } x$
- **User-movie interaction**: $b_i = \text{bias of movie } i$

---

**Global: Baseline predictor**
- Separates users and movies
- Benefits from insights into user’s behavior
- Among the main practical contributions of the competition

**Local: User-Movie interaction**
- Characterizes the matching between users and movies
- Attracts most research in the field
- Benefits from algorithmic and mathematical innovations
Baseline Predictor

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

  - Rating scale of user $x$
  - 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”)
Putting It All Together

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

- **Example:**
  - Mean rating: \( \mu = 3.7 \)
  - You are a critical reviewer: your mean rating is 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:
    \[ = 3.7 - 1 + 0.5 = 3.2 \] (before user movie interaction)
Fitting the New Model

- **Solve:**

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

**goodness of fit**

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

\(\lambda\) is selected via grid-search on a validation set

- **Stochastic gradient decent to find parameters**
  - **Note:** Both biases \(b_x, b_i\) as well as interactions \(q_i, p_x\) are treated as parameters (and we learn them)
Performance of Various Methods

Global average: 1.1296
User average: 1.0651
Movie average: 1.0533
Netflix: 0.9514
Basic Collaborative filtering: 0.94
CF with learned weights: 0.91
Latent factors: 0.90
Latent factors + Biases: 0.89
Grand Prize: 0.8563
The Netflix Challenge: 2006-09
Temporal Biases Of Users

- **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 & Factors

- **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 \( b_x \) and \( b_i \) to depend on time
  - (1) Parameterize time-dependence by linear trends
  - (2) Each bin corresponds to 10 consecutive weeks
    \[ b_i(t) = b_i + b_i,\text{Bin}(t) \]

- **Add temporal dependence to factors**
  - \( p_x(t) \)... user preference vector on day \( t \)
Performance of Various Methods

Global average: 1.1296
User average: 1.0651
Movie average: 1.0533
Netflix: 0.9514

Basic Collaborative filtering: 0.94
Collaborative filtering++: 0.91
Latent factors: 0.90
Latent factors+Biases: 0.89
Latent factors+Biases+Time: 0.876

Still no prize! 😞
Getting desperate.
Try a “kitchen sink” approach!

Grand Prize: 0.8563
The big picture

Solution of BellKor's Pragmatic Chaos

All developed CF models

BRISMF, MF1, NSVDD, SVD-Time, Split RBM, BK3, BK5-SVD++, BMF2
SBRAMF, MF1 NSVDD, RBM daze, FRBM, BK3, BK5-SVD++, GTE
Movie KNN, V Baseline, 1/2/3, DRBM, SVM++, ISVD2, MF2
KNN+time, NSVD1, SBM, Integrated M, RBM, GTE
User KNN, Movie KNN, CTD/MTD, SVDDNN
User KNN, Classif. ModeKNN 1...5, Asym. 1/2/3

Latent User and
Movie Features

Probe Blending

approx. 500 predictors

Probe Blending

200 blends

30 blends

Linear Blend 10.09 % improvement
Standing on June 26th 2009

June 26th submission triggers 30-day “last call”
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 get small improvements in their scores
  - Realize they are in direct competition with team *Ensemble*

- **Strategy**
  - Both teams carefully monitoring the leader board
  - 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**
  - Only 1 final submission could be made in the last 24h

- **24 hours before deadline...**
  - **BellKor** team member in Austria notices (by chance) that **Ensemble** posts a score that is slightly better than BellKor’s

- **Frantic last 24 hours for both teams**
  - Much computer time on final optimization
  - 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....
## Leaderboard

Showing Test Score. Click here to show quiz score

Display top leaders.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Team Name</th>
<th>Best Test Score</th>
<th>% Improvement</th>
<th>Best Submit Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BellKor's Pragmatic Chaos</td>
<td>0.8567</td>
<td>10.06</td>
<td>2009-07-26 18:18:28</td>
</tr>
<tr>
<td>2</td>
<td>The Ensemble</td>
<td>0.8567</td>
<td>10.06</td>
<td>2009-07-26 18:38:22</td>
</tr>
<tr>
<td>3</td>
<td>Grand Prize Team</td>
<td>0.8592</td>
<td>9.91</td>
<td>2009-07-10 23:41:48</td>
</tr>
<tr>
<td>4</td>
<td>Opera Solutions and Vandelay United</td>
<td>0.8588</td>
<td>9.84</td>
<td>2009-07-10 01:12:31</td>
</tr>
<tr>
<td>5</td>
<td>Vandelay Industries I</td>
<td>0.8591</td>
<td>9.81</td>
<td>2009-07-10 00:32:20</td>
</tr>
<tr>
<td>6</td>
<td>PragmaticTheory</td>
<td>0.8594</td>
<td>9.77</td>
<td>2009-06-24 12:06:56</td>
</tr>
<tr>
<td>7</td>
<td>BellKor in BigChaos</td>
<td>0.8601</td>
<td>9.70</td>
<td>2009-05-13 08:14:09</td>
</tr>
<tr>
<td>8</td>
<td>Dace</td>
<td>0.8612</td>
<td>9.59</td>
<td>2009-07-24 17:18:43</td>
</tr>
<tr>
<td>9</td>
<td>Feeds2</td>
<td>0.8622</td>
<td>9.48</td>
<td>2009-07-12 13:11:51</td>
</tr>
<tr>
<td>10</td>
<td>BigChaos</td>
<td>0.8623</td>
<td>9.47</td>
<td>2009-07-24 00:34:07</td>
</tr>
<tr>
<td>11</td>
<td>Opera Solutions</td>
<td>0.8624</td>
<td>9.46</td>
<td>2009-07-26 17:19:11</td>
</tr>
<tr>
<td>12</td>
<td>BellKor</td>
<td>0.8642</td>
<td>9.27</td>
<td>2009-07-15 14:53:22</td>
</tr>
<tr>
<td>13</td>
<td>xianglang</td>
<td>0.8643</td>
<td>9.26</td>
<td>2009-04-22 18:31:32</td>
</tr>
<tr>
<td>14</td>
<td>Gravity</td>
<td>0.8651</td>
<td>9.18</td>
<td>2009-06-21 19:24:53</td>
</tr>
<tr>
<td>15</td>
<td>Ces</td>
<td>0.8653</td>
<td>9.15</td>
<td>2009-07-15 15:53:04</td>
</tr>
<tr>
<td>16</td>
<td>Invisible Ideas</td>
<td>0.8662</td>
<td>9.06</td>
<td>2009-05-24 10:02:54</td>
</tr>
<tr>
<td>17</td>
<td>Just a guy in a garage</td>
<td>0.8666</td>
<td>9.02</td>
<td>2009-07-25 16:00:54</td>
</tr>
<tr>
<td>18</td>
<td>J Dennis Su</td>
<td>0.8666</td>
<td>9.02</td>
<td>2009-03-21 16:20:50</td>
</tr>
<tr>
<td>19</td>
<td>Craig Carmichael</td>
<td>0.8688</td>
<td>9.00</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>acmehill</td>
<td>0.8688</td>
<td>9.00</td>
<td></td>
</tr>
</tbody>
</table>
Million $ Awarded Sept 21st 2009
What’s the moral of the story?

Submit early! 😊
Some slides and plots borrowed from Yehuda Koren, Robert Bell and Padhraic Smyth

Further reading:

- Y. Koren, Collaborative filtering with temporal dynamics, KDD ’09