CS 277, Data Mining

Recommender Systems

Padhraic Smyth
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

Thanks to Yehuda Koren and Jure Leskovec for contributing material for many of these slides.
Progress Reports

• First Progress Report due on Monday

• Hand in hardcopy in class, also upload electronic version to EEE directory

• Required:
  – 2 to 4 pages
  – Clear indication of work that was done *since* initial proposal
  – You can assume the reader (me) has read your proposal (so no need to repeat information)
  – Feel free to use figures/tables
  – Expect at a minimum that you have started to look at your data
  – For teams please report at the beginning “who did what”
Reading on Recommender Systems (on Web page)

• Good overviews
  – Recommender systems, Melville and Sindhwani, Encyclopaedia of Machine Learning, 2010 (a good starting point)
  – Chapter on recommendation algorithms from the online text Mining of Massive Data Sets, Rajaraman, Leskovec, and Ullman.
  – Amazon.com recommendations: item-to-item collaborative filtering, Linden, Smith, and York, 2003 (overview of the basic components of Amazon's recommender system)

• Matrix Factorization
  – Matrix factorization techniques for recommender systems, Koren, Bell, Volinsky, IEEE Computer, 2009
  – Advances in collaborative filtering, Koren and Bell, chapter from the Recommender Systems Handbook, 2011

• Other Aspects
  – Recommender systems: from algorithms to user experience, Konstain and Riedl, 2012 (emphasizes that the user experience is important, not just predictive accuracy)
  – Factorization machines with libFM, S. Rendle, 2012, with associated publicly-available software for libFM
“Ratings” Data

- Data with users \( u \) and items \( i \)
  - E.g., items are products purchases, movies viewed, songs listened to, etc

- Can represent as an \( N \times M \) sparse matrix
  - \( N = \) number of users, \( M = \) number of items

- Entries \( r_{ui} \)
  - **Explicit Ratings**: \( r_{ui} = \) user \( u \)’s rating of item \( i \) (e.g. on a scale of 1 to 5)
  - **Implicit Ratings**: \( r_{ui} = 1 \) if user \( u \) purchased/read/listened to item \( i \)

  \( r_{ui} = 0 \) if no purchase or rating
  (note that 0 means a user’s preference is unknown, not that they don’t like the item)

- Automated recommender systems
  - Given a user and their ratings (if any) recommend to this user other items that the user may be interested in
The Recommender Space as a Bipartite Graph

- User-User Links
- Item-Item Links

Links derived from similar attributes, explicit connections

Observed preferences
(Ratings, purchases, page views, play lists, bookmarks, etc)

Links derived from similar attributes, similar content, explicit cross references
General Approaches to Automated Recommender Systems

1. Content-based Recommendations
   - Use attributes/features of items to recommend similar items
   - Ignores ratings data

2. Collaborative filtering
   - Use ratings matrix to recommend items, ignores item and user content data
   - 2 broad types:
     (1) Nearest-neighbor methods
     (2) Matrix factorization methods

3. Hybrid methods
   - Combine both content and ratings data (often provides “state of the art” performance)
The $1 Million Question
### Ratings Data

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>5</th>
<th>5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

- **17,700 movies**
- **480,000 users**
Training Data

100 million ratings (matrix is 99% sparse)

Rating = [user, movie-id, time-stamp, rating value]

Generated by users between Oct 1998 and Dec 2005

Users randomly chosen among set with at least 20 ratings
  - Small perturbations to help with anonymity
Ratings Data

17,700 movies

480,000 users

Test Data Set
(most recent ratings)
Ratings per Movie in the Training Data

Average number of ratings/movie: 5672
Ratings per User in the Training Data

Average number of ratings/user: 208
### Most Loved Movies

<table>
<thead>
<tr>
<th>Movie</th>
<th>Avg rating</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Shawshank Redemption</td>
<td>4.593</td>
<td>137812</td>
</tr>
<tr>
<td>Lord of the Rings : The Return of the King</td>
<td>4.545</td>
<td>133597</td>
</tr>
<tr>
<td>The Green Mile</td>
<td>4.306</td>
<td>180883</td>
</tr>
<tr>
<td>Lord of the Rings : The Two Towers</td>
<td>4.460</td>
<td>150676</td>
</tr>
<tr>
<td>Finding Nemo</td>
<td>4.415</td>
<td>139050</td>
</tr>
<tr>
<td>Raiders of the Lost Ark</td>
<td>4.504</td>
<td>117456</td>
</tr>
</tbody>
</table>

### 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
### Users with the Most Ratings

<table>
<thead>
<tr>
<th>User ID</th>
<th># Ratings</th>
<th>Mean Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>305344</td>
<td>17,651</td>
<td>1.90</td>
</tr>
<tr>
<td>387418</td>
<td>17,432</td>
<td>1.81</td>
</tr>
<tr>
<td>2439493</td>
<td>16,560</td>
<td>1.22</td>
</tr>
<tr>
<td>1664010</td>
<td>15,811</td>
<td>4.26</td>
</tr>
<tr>
<td>2118461</td>
<td>14,829</td>
<td>4.08</td>
</tr>
<tr>
<td>1461435</td>
<td>9,820</td>
<td>1.37</td>
</tr>
<tr>
<td>1639792</td>
<td>9,764</td>
<td>1.33</td>
</tr>
<tr>
<td>1314869</td>
<td>9,739</td>
<td>2.95</td>
</tr>
</tbody>
</table>
Scoring

Minimize root mean square error

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

Does not necessarily correlate well with user satisfaction

But is a widely-used well-understood quantitative measure
Key Technical Ideas
Key Technical Ideas

• Matrix factorization for recommender systems

• Learning the parameters of our model by gradient descent

• Making this learning very fast with stochastic gradient descent
Matrix Factorization of Ratings Data

Factorize the large $N \times M$ matrix into 2 “skinny” matrices, $k$ is much smaller than $N$ or $M$, e.g., $k = 100$
Matrix Factorization of Ratings Data

Idea: represent users and movies in a k-dimensional “latent factor” space, to capture general properties of user and movie characteristics.
Matrix Approximation with SVD

\[ \mathbf{R} \approx \mathbf{U} \Sigma \mathbf{V}^t = \mathbf{P} \mathbf{Q}^t \]

where:
- columns of \( \mathbf{V} \) are first \( f \) eigenvectors of \( \mathbf{R}^t \mathbf{R} \)
- \( \Sigma \) is diagonal with \( f \) largest eigenvalues
- rows of \( \mathbf{U} \) are coefficients in reduced dimension \( \mathbf{V} \)-space

This approximation is the best rank-\( f \) approximation to matrix \( \mathbf{R} \) in a least squares sense (principal components analysis)
The Princess Diaries
The Color Purple
Sense and Sensibility
The Lion King
Ocean’s 11
The Lion King
Independence Day
Dumb and Dumber

Geared towards females
Geared towards males
Serious
Funny

Factor 1
Factor 2
The image contains a two-dimensional chart with movies plotted on different axes. The horizontal axis is labeled "Geared towards females," and the vertical axis is labeled "Geared towards males." The chart includes the following movies:

- **Serious**
  - Amadeus
  - The Color Purple

- **Funny**
  - Ocean’s 11
  - Lethal Weapon
  - The Princess Diaries
  - The Lion King

- **Factor 1**
  - Independence Day
  - Dumb and Dumber

- **Factor 2**
  - Sense and Sensibility
  - The Princess Diaries

The chart is used to illustrate the different genres and target audiences of these movies.
The diagram shows a two-dimensional space with two factors:

- Factor 1: Geared towards males vs. Geared towards females
- Factor 2: Funny vs. Serious

The movies are placed on the diagram as follows:
- The Color Purple: Geared towards females, Serious
- Amadeus: Geared towards females, Serious
- The Lion King: Geared towards males, Serious
- Braveheart: Geared towards males, Serious
- Lethal Weapon: Geared towards males, Serious
- Ocean’s 11: Geared towards females, Funny
- Sense and Sensibility: Geared towards females, Funny
- The Princess Diaries: Geared towards females, Funny
- Independence Day: Funny
- Factor 1: Geared towards males
- Factor 2: Funny

There is a note indicating that if there is little data for this user, regularize towards 0.
Figure from Koren, Bell, Volinksy, IEEE Computer, 2009
Computation of Matrix Factors

Problem 1:
Finding the $k$ factors is equivalent to performing a singular value decomposition of a matrix, i.e.,

Let $R$ be an $N \times M$ matrix

SVD computation has complexity $O(NM^2 + M^3)$
Computation of Matrix Factors

Problem 1:
Finding the k factors is equivalent to performing a singular value decomposition of a matrix, i.e.,

Let R be an \( N \times M \) matrix

SVD computation has complexity \( O(NM^2 + M^3) \)

Problem 2:
Most of the entries in R are missing, i.e.,
only \( 100 \times 10^6 / (480k \times 17k) \approx 1\% \) are present
Dealing with Missing Data

\[ r_{ui} \approx q_t^i \ p_u \]

\[
\min_{q,p} \sum_{(u,i) \in R} \left( r_{ui} - q_t^i \ p_u \right)^2
\]

sum is only over known ratings
Dealing with Missing Data

\[ r_{ui} \approx q_i^T p_u \]

\[
\min_{q,p} \sum_{(u,i) \in R} (r_{ui} - q_i^T p_u)^2
\]

Add regularization

\[
\min_{q,p} \sum_{(u,i) \in R} (r_{ui} - q_i^T p_u)^2 + \lambda (|q_i|^2 + |p_u|^2)
\]
Components of a rating predictor

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

**movie bias**
- 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

* (slide from Yehuda Koren)
A 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”)

(slide from Yehuda Koren)
Modeling Systematic Biases

\[ r_{ui} \approx \mu + b_u + b_i + q_t^i \cdot p_u \]

overall “intercept”

bias for user \( u \)

bias for movie \( i \)

user-movie interactions

Example:

Intercept/mean \( \mu = 3.7 \)

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: Minimize as a Function of Parameters

\[
\min_{q, p} \left\{ \sum_{(u, i) \in R} \left( r_{ui} - (\mu + b_u + b_i + q_i^t p_u) \right)^2 + \lambda \left( |q_i|^2 + |p_u|^2 + |b_u|^2 + |b_i|^2 \right) \right\}
\]

goodness of fit

Typically selected via grid-search on a validation set

We will use stochastic gradient descent to find parameters

Note: biases \(b_u, b_i\) as well as interactions \(q_i, p_u\) are treated as parameters (we estimate them)
Principle of Gradient Descent

Let $\theta$ be our current parameter vector and $E(\theta)$ be our error function.

$\theta$ is a vector of parameters of dimension $p \times 1$.

The gradient of $\theta$, $\nabla_\theta(E)$ is a $p$-dimensional gradient vector, where entry $j$ is the partial derivative of $E$ with respect to the $j$th component of the parameter vector $\theta$.

To move downhill in $\theta$ space, we take a step in the opposite direction of the gradient, i.e.,

$$\theta^{\text{new}} = \theta^{\text{old}} - \gamma \nabla_\theta(E)$$

where $\gamma$ is the step size.

For the $j$th component, we move/update the parameter $\theta_j$ as follows:

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} - \gamma \frac{\partial E}{\partial \theta_j}$$
Gradient Descent Algorithm for Parameter Learning

1. Initialize all parameters to some initial values (e.g., randomly or heuristically)

2. Given the current parameter vector $\theta$, compute the gradient $\nabla_{\theta}(E)$

3. Update each parameter $j$:

$$
\theta_j^{\text{new}} = \theta_j^{\text{old}} - \gamma \frac{\partial E}{\partial \theta_j}
$$

4. Check if the algorithm has converged, e.g., if $E(\theta^{\text{new}}) - E(\theta^{\text{old}}) < \epsilon$

5. If not converged, return to step 2
How to set the Learning Rate?

$$\theta_{j}^{\text{new}} = \theta_{j}^{\text{old}} - \gamma \frac{\partial E}{\partial \theta_{j}}$$

In practice we may also want to decrease the step-size at each iteration, e.g., replace $\gamma$ by $\alpha \gamma$ at each iteration, where (e.g.,) $\alpha = 0.9$. Theoretically this is necessary to ensure convergence.

There is no general theory on how to select the step size $\alpha$ and the schedule for decreasing it over time.

Good settings will vary by problem and data set.

In practice finding good settings requires trial and error.
Learning a Recommender Model with Biases

Prediction model:

\[ \hat{r}_{ui} = \mu + b_u + b_i \]

Error:

\[ E = \sum_{u,i} \left( r_{ui} - \hat{r}_{ui} \right)^2 = \sum_{u,i} e_{u,i}^2 \]

where

\[ e_{u,i} = \text{prediction error on rating } r_{ui} = r_{ui} - \hat{r}_{ui} = r_{ui} - (\mu + b_u + b_i) \]
Gradient Descent for One Parameter

Gradient descent to learn $\mu$:

$$\frac{\partial E}{\partial \mu} = \frac{\partial}{\partial \mu} \sum_{u,i} e_{u,i}^2 = \sum_{u,i} \frac{\partial}{\partial \mu} e_{u,i}^2$$

Chain rule:

$$\frac{\partial e_{u,i}^2}{\partial \mu} = \frac{\partial e_{u,i}^2}{\partial e_{u,i}} \times \frac{\partial e_{u,i}}{\partial \mu}$$

where

$$\frac{\partial e_{u,i}^2}{\partial e_{u,i}} \times \frac{\partial e_{u,i}}{\partial \mu} = 2e_{u,i} \times \frac{\partial e_{u,i}}{\partial \mu} = 2e_{u,i} \times -1 = -2e_{u,i}$$
Gradient Descent for One Parameter

Gradient descent to learn $\mu$:

$$\frac{\partial E}{\partial \mu} = \frac{\partial}{\partial \mu} \sum_{u,i} e_{u,i}^2 = \sum_{u,i} \frac{\partial}{\partial \mu} e_{u,i}^2$$

Chain rule:

$$\frac{\partial e_{u,i}^2}{\partial \mu} = \frac{\partial e_{u,i}^2}{\partial e_{u,i}} \times \frac{\partial e_{u,i}}{\partial \mu}$$

where

$$\frac{\partial e_{u,i}^2}{\partial e_{u,i}} \times \frac{\partial e_{u,i}}{\partial \mu} = 2e_{u,i} \times \frac{\partial e_{u,i}}{\partial \mu} = 2e_{u,i} \times -1 = -2e_{u,i}$$

The total gradient, summed across all rating pairs $(u, i)$ is:

$$\frac{\partial E}{\partial \mu} = -\sum_{u,i} e_{u,i}$$

where the sum is over all $R$ observed ratings.

To minimize a function we need to move in the opposite direction of the gradient, so the gradient direction we take for the $\mu$ component of the gradient is in the direction $\sum_{u,i} e_{u,i}$. 
Gradient Descent for User and Item Bias Parameters

We can use a similar derivation to get the updates for $b_u$ and $b_i$ (these are vectors of parameters of size $N$ and $M$ respectively)

$$\frac{\partial E}{\partial b_u} = - \sum_{u,i} e_{u,i}$$

$$\frac{\partial E}{\partial b_i} = - \sum_{u,i} e_{u,i}$$

Recall that the general gradient update equation is as follows:

$$\theta^{\text{new}}_j = \theta^{\text{old}}_j - \gamma \frac{\partial E}{\partial \theta_j}$$
Gradient Updates for All Bias Parameters

This leads to gradient update equations for $\mu, b_u, b_i$ as follows:

\[
\begin{align*}
\mu^{\text{new}} &= \mu^{\text{old}} + \gamma \sum_{u,i} e_{u,i} \\
b_u^{\text{new}} &= b_u^{\text{old}} + \gamma \sum_{u,i} e_{u,i}, \quad u = 1, \ldots, N \\
b_i^{\text{new}} &= b_i^{\text{old}} + \gamma \sum_{u,i} e_{u,i}, \quad i = 1, \ldots, M
\end{align*}
\]
Gradient Updates for All Bias Parameters

This leads to gradient update equations for $\mu, b_u, b_i$ as follows:

\[
\begin{align*}
    \mu^{\text{new}} &= \mu^{\text{old}} + \gamma \sum_{u,i} e_{u,i} \\
    b_u^{\text{new}} &= b_u^{\text{old}} + \gamma \sum_{u,i} e_{u,i}, \quad u = 1, \ldots, N \\
    b_i^{\text{new}} &= b_i^{\text{old}} + \gamma \sum_{u,i} e_{u,i}, \quad i = 1, \ldots, M
\end{align*}
\]

The complexity of each update is $O(R + M + N)$ where:

- $R$ is the number of ratings in the sum
- $N$ and $M$ are the number of users and items respectively
First-Order and Second-Order Methods

The algorithm on the previous slide is “first-order.”

More sophisticated algorithms also use “second-order” information, with a $p \times p$ matrix of 2nd-order partial derivatives, where $p = M + N$ is the number of parameters.

This incurs a cost of $O(p^2)$ for updates versus $O(p)$ for gradient descent.

Even though the 2nd-order may take more accurate steps per iteration, overall convergence time may be slower, particularly for large $p$.

Impractical for large $p$, e.g., large $N$ and/or $M$. 

Stochastic Gradient Method

Idea: compute an estimate of the gradient on small parts of the data and then update parameters.

For example, we can compute the gradient using just a single rating \((u^*, i^*)\), and then update:

\[
\begin{align*}
\mu^\text{new} &= \mu^\text{old} + \gamma e_{u^*, i^*} \\
\mu^\text{new} &= \mu^\text{old} + \gamma e_{u^*, i^*} \\
\mu^\text{new} &= \mu^\text{old} + \gamma e_{u^*, i^*}
\end{align*}
\]

This is called the **stochastic gradient** method.

Each update is computationally cheap.

So we update the parameters \(R\) times per iteration compared to conventional gradient descent (e.g., \(R = 100\) million ratings for Netflix) and take a lot of noisy/cheap steps in parameter space.

Empirically, we can often converge to a solution much faster with stochastic gradient than with the full gradient.
SG vs GD

Convergence of GD vs. SG

GD improves the value of the objective function at every step.

SG improves the value but in a “noisy” way.

GD takes fewer steps to converge but each step takes much longer to compute.

In practice, SG is much faster!
Learning the (Latent) Matrix Factors

We can use the same idea of (stochastic) gradient descent to learn our factors.

Recall that:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^t p_u$$

where $q_i$ and $p_u$ are the $k$-dimensional factor vectors for item $i$ and user $u$ respectively.

We can treat these factors as an additional set of $k(N + M)$ parameters to learn and use gradient descent to learn them.

Update equations for the factors (for stochastic gradient):

$$p_u^{\text{new}} = p_u^{\text{old}} + \gamma e_{u,i} q_i$$
$$q_i^{\text{new}} = q_i^{\text{old}} + \gamma e_{u,i} p_u$$
Regularizing the Parameters

As discussed earlier, it is often useful to regularize and add a penalty term to penalize large parameter values.

This gives a modified objective function of the form

\[ E = \sum_{u,i} \left( r_{u,i} - \hat{r}_{u,i} \right)^2 + \lambda \sum_{u,i} b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2 \]

Now we get modified gradient terms and our update equations change accordingly, e.g.,

\[ p_u^{\text{new}} = p_u^{\text{old}} + \gamma(e_{u,i}q_i^{\text{old}} - \lambda p_u^{\text{old}}) \]
\[ q_i^{\text{new}} = q_i^{\text{old}} + \gamma(e_{u,i}p_u^{\text{old}} - \lambda q_i^{\text{old}}) \]

In the Netflix competition the winning team used \( \gamma = 0.005, \lambda = 0.02 \) (determined by extensive experimentation on validation data)
Adding “Content” Features for Items and/or Users

Say we also have features for the items (will just do items here, same math for features for users)

Let $x_i$ be a $d$-dimensional feature vector for item $i$

We can use include the features in the prediction model, weighted by parameters $\alpha_j, j = 1, \ldots, d$

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^tp_u + \sum_{j=1}^{d} \alpha_j x_{ij}$$

where $x_{ij}$ is the $j$th feature value for item $i$.

Note that this is like a generalized form of linear regression, where we have bias and latent factor terms in the prediction as well as the more typical weighted sum of feature values.
Figure from Koren, Bell, Volinksy, IEEE Computer, 2009
Adding Implicit Information

<table>
<thead>
<tr>
<th>1</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>3</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

18000 movies

400,000 users

Test Data Set
Figure from Koren, Bell, Volinksy, IEEE Computer, 2009
Rating by movie age

Explanation for increase?
Adding Time Effects

\[ r_{ui} \approx \mu + b_u + b_i + \text{user-movie interactions} \]

Add time dependence to biases

\[ r_{ui} \approx \mu + b_u(t) + b_i(t) + \text{user-movie interactions} \]

Time-dependence parametrized by linear trends, binning, and other methods

For details see
Y. Koren, Collaborative filtering with temporal dynamics, ACM SIGKDD Conference 2009
Adding Time Effects

\[ r_{ui} \approx \mu + b_u(t) + b_i(t) + q_t^i p_u(t) \]

Add time dependence to user “factor weights”

Models the fact that user’s interests over “genres” (the q’s) may change over time
Figure from Koren, Bell, Volinksy, IEEE Computer, 2009
The Kitchen Sink Approach....

- Many options for modeling
  - Variants of the ideas we have seen so far
    - Different numbers of factors
    - Different ways to model time
    - Different ways to handle implicit information
    - ....
  - Other models (not described here)
    - Nearest-neighbor models
    - Restricted Boltzmann machines

- Model averaging was useful in the Netflix competition....
  - 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
The Netflix Competition: 2006-2009
Training Data

100 million ratings

Held-Out Data

3 million ratings

Labels known publicly

Labels only known to Netflix
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

Labels known publicly

Labels only known to Netflix
Structure of Competition

• Register to enter at Netflix site

• Download training data of 100 million ratings
  – 480k users x 17.7k movies
  – Anonymized

• Submit predictions for 3 million ratings in “test set”
  – True ratings are known only to Netflix

• Can submit multiple times (limit of once/day)

• Prize
  – $1 million dollars if error is 10% lower than Netflix current system
  – Annual progress prize of $50,000 to leading team each year
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
Other Aspects of Rules

• Rights
  – Software + non-exclusive license to Netflix
  – Algorithm description to be posted publicly

• Final prize details
  – If public score of any contestant is better than 10%, this triggers a 30-day final competition period
    – Anyone can submit scores in this 30-day period
    – Best score at the end of the 30-day period wins the $1 million prize

• Competition not open to entrants in North Korea, Iran, Libya, Cuba....and Quebec
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
Setting up and Launching...

- **Summer 2006**
  - Email from Netflix about large monetary award
  - Is this real?
  - Apparently so: serious and well-organized
  - Spent summer carefully designing data set and rules

- **Official Launch, Oct 2\(^{nd}\) 2006**
  - Email lists, conferences, press releases, etc

  - Significant initial interest in research community, blogs, etc
    - 40,000 teams (eventually) from over 150 countries.

  - Number of initial registrants significantly exceeded expectations
Progress in first 3 months

Oct 2, 2006  Launch of competition

Oct 8, 2006  WXY Consulting already better than Cinematch score

Oct 15, 2006 3 teams above Cinematch, one with 1.06% improvement (qualifying for $50k progress prize)

Dec, 2006: Jim Bennett from Netflix describes progress so far during an invited talk at NIPS
Prize Progress
Prize Submissions
Prize Submissions

Movie avg
Bayes?
Multinominal?
User avg
Pearson?
Leaders

Submissions

% improvement

0 2 4 6 8

Prize Submissions
KDD Cup

Co-organized by ACM SIGKDD and Netflix

KDD Cup is the first and the oldest data mining competition, and is an integral part of the annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). This year's KDD Cup will be related to (but different from) the current Netflix Prize competition (http://www.netflixprize.com/). There will also be a workshop at the KDD-07 conference, where the participants of both the KDD Cup and the current Netflix Prize competition will present their papers and exchange ideas.

There are 2 parallel options for participating:

1. The KDD Cup competition (open to all)
2. Workshop paper submissions (open to Netflix prize participants only)

Full details are provided at the KDD Cup and Workshop 2007 website:


If you have any comments and suggestions, please email liub@cs.uic.edu.

Organizing Committee

- Jim Bennett, Netflix, USA
- Charles Elkan, University of California, San Diego, USA
- Bing Liu (Chair), University of Illinois at Chicago, USA
- Padhraic Smyth, University of California, Irvine, USA
- Domonkos Tikk, Budapest University of Technology and Economics, Hungary
First Progress Prize, October 2007

Progress prize: $50k annually awarded to leading team provided there is at least 1% improvement over previous year

Sept 2\textsuperscript{nd}  First progress prize “30 day” last call

Oct 2\textsuperscript{nd}  Leaders were BellKor, 8.4% improvement  
(Yehuda Koren, Bob Bell, Chris Volinksy, AT&T Research)
First Progress Prize, October 2007

Progress prize: $50k annually awarded to leading team provided there is at least 1% improvement over previous year

Sept 2\textsuperscript{nd}  First progress prize “30 day” last call

Oct 2\textsuperscript{nd}  Leaders were BellKor, 8.4% improvement  
(Yehuda Koren, Bob Bell, Chris Volinksy, AT&T Research)

Oct/Nov  Code and documentation submitted for judging

Complicated methods: primarily relying on factor models

Nov 13  Winners officially declared and BellKor documentation published on Netflix Web site
Progress in 2008...

Progress slows down...improvements are incremental

Many of the leading prize contenders publishing their methods and techniques at academic conferences (2\textsuperscript{nd} KDD workshop in August)

Much speculation on whether the prize would ever be won – is 10% even attainable?

Many initial participants had dropped out – too much time and effort to seriously compete

But leaderboard and forum still very active
Progress Prize 2008

Sept 2\textsuperscript{nd} Only 3 teams qualify for 1% improvement over previous year

Oct 2\textsuperscript{nd} Leading team has 9.4% overall improvement

Oct/Nov Code/documentation reviewed and judged
## Progress Prize 2008

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sept 2(^{nd})</td>
<td>Only 3 teams qualify for 1% improvement over previous year</td>
</tr>
<tr>
<td>Oct 2(^{nd})</td>
<td>Leading team has 9.4% overall improvement</td>
</tr>
<tr>
<td>Oct/Nov</td>
<td>Code/documentation reviewed and judged</td>
</tr>
</tbody>
</table>

Progress prize ($50,000) awarded to BellKor team of 3 AT&T researchers (same as before) plus 2 Austrian graduate students, Andreas Toscher and Martin Jahrer

Key winning strategy: clever “blending” of predictions from models used by both teams

Speculation that 10% would be attained by mid-2009
Example of Predictor Specifications....

**OB-42 rmse=0.8998**
MovieKNNV3, Residual: OB-39, Pearson correlation, $K = 13$, $\alpha = 658$, $\beta = 2480$, $\gamma = -2.6$, $\delta = 7.5$

**OB-43 rmse=0.8971**
SVD-AUF, Residual: OB-30, adaptiveUserFactorMode=KRR, kernelType=extended-polynomial, $\lambda = 4.78376$, $\alpha = 0.657533$, $\gamma = 0.720031$, $\beta = 3.27554$

**OB-44 rmse=0.9245**
GE, Residual: OB-35, 16 effects, $\alpha_1 = 374.977$, $\alpha_2 = 8.90702e - 05$, $\alpha_3 = 2535.9$, $\alpha_4 = 900.414$, $\alpha_5 = 1.04115e - 05$, $\alpha_6 = 2087.92$, $\alpha_7 = 131.291$, $\alpha_8 = 3173.84$, $\alpha_9 = 1.45471e - 06$, $\alpha_{10} = 6.40823e - 08$, $\alpha_{11} = 4451.15$, $\alpha_{12} = 274.423$, $\alpha_{13} = 1020.64$, $\alpha_{14} = 0.00758424$, $\alpha_{15} = 3858.57$, $\alpha_{16} = 0.00346888$

**OB-45 rmse=0.8998**
MovieKNNV3, Residual: OB-40, Spearman’s rank correlation, $K = 38$, $\alpha = 667.6$, $\beta = 255.5$, $\gamma = -1.39$, $\delta = 3.3$

**OB-46 rmse=0.8958**
MovieKNNV3, Residual: OB-41, Pearson correlation, $K = 60$, $\alpha = 804$, $\beta = 231$, $\gamma = -2.6$, $\delta = 17$

**OB-47 rmse=0.9777**
NSVD2, Residual: no, $k = 50$, $\eta_i = 2e - 3$, $\eta_u = 2e - 3$, $\eta_{\mu_i} = 2e - 3$, $\eta_{\mu_u} = 2e - 3$, $\lambda_i = 5e - 4$, $\lambda_u = 5e - 4$, $\lambda_{\mu_i} = 1e - 4$, $\lambda_{\mu_u} = 1e - 4$, 3 epochs
Solution of BellKor's Pragmatic Chaos

All developed CF models
- BRISMF
- MF1
- NSVDD
- SVT-Time
- SVD++
- Split RBM
- DRBM
- SVD-AUF
- BM1
- BM2
- BM3
- BM4
- BM5
- SVD++
- BSVD
- ISVD
- UNKNN
- Classif.
- ModeKNN

Latent User and Movie Features

Probe Blending
- 200 blends

Probe Blending
- 30 blends

Linear Blend 10.09% improvement

approx. 500 predictors
June 26\textsuperscript{th} 2009: after 1000 Days and nights...

![NetflixFprize Leaderboard](image-url)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Team Name</th>
<th>Best Score</th>
<th>% Improvement</th>
<th>Last Submit Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BellKor's Pragmatic Chaos</td>
<td>0.8558</td>
<td>10.05</td>
<td>2009-06-26 18:42:37</td>
</tr>
<tr>
<td>2</td>
<td>PragmaticTheory</td>
<td>0.8582</td>
<td>8.80</td>
<td>2009-06-25 22:15:51</td>
</tr>
<tr>
<td>3</td>
<td>BellKor in BigChaos</td>
<td>0.8590</td>
<td>8.71</td>
<td>2009-05-13 08:14:09</td>
</tr>
<tr>
<td>4</td>
<td>Grand Prize Team</td>
<td>0.8593</td>
<td>8.68</td>
<td>2009-06-12 08:20:24</td>
</tr>
<tr>
<td>5</td>
<td>Dace</td>
<td>0.8604</td>
<td>8.56</td>
<td>2009-04-22 05:57:03</td>
</tr>
<tr>
<td>6</td>
<td>BigChaos</td>
<td>0.8613</td>
<td>8.47</td>
<td>2009-06-23 23:06:52</td>
</tr>
<tr>
<td>7</td>
<td>BellKor</td>
<td>0.8620</td>
<td>9.40</td>
<td>2009-06-24 07:16:02</td>
</tr>
<tr>
<td>8</td>
<td>Gravity</td>
<td>0.8634</td>
<td>9.25</td>
<td>2009-04-22 18:32:32</td>
</tr>
<tr>
<td>9</td>
<td>Opera Solutions</td>
<td>0.8638</td>
<td>9.21</td>
<td>2009-06-26 23:18:13</td>
</tr>
<tr>
<td>10</td>
<td>BruceDengDaoCIYiYou</td>
<td>0.8638</td>
<td>9.21</td>
<td>2009-06-27 00:55:55</td>
</tr>
<tr>
<td>11</td>
<td>pengpengzhou</td>
<td>0.8638</td>
<td>9.21</td>
<td>2009-06-27 01:06:43</td>
</tr>
<tr>
<td>12</td>
<td>lvector</td>
<td>0.8639</td>
<td>9.20</td>
<td>2009-06-26 13:49:04</td>
</tr>
<tr>
<td>13</td>
<td>xiangliang</td>
<td>0.8639</td>
<td>9.20</td>
<td>2009-06-26 07:47:34</td>
</tr>
<tr>
<td>14</td>
<td>Feeds2</td>
<td>0.8641</td>
<td>9.18</td>
<td>2009-06-26 22:51:55</td>
</tr>
<tr>
<td>15</td>
<td>Ces</td>
<td>0.8642</td>
<td>9.17</td>
<td>2009-06-24 14:34:14</td>
</tr>
</tbody>
</table>
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 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....
<table>
<thead>
<tr>
<th>Rank</th>
<th>Team Name</th>
<th>Best Score</th>
<th>% Improvement</th>
<th>Last Submit Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The Ensemble</td>
<td>0.6553</td>
<td>10.10</td>
<td>2009-07-26 18:38:22</td>
</tr>
<tr>
<td>2</td>
<td>BellKor Pragmatic Chaos</td>
<td>0.6554</td>
<td>10.09</td>
<td>2009-07-26 18:16:28</td>
</tr>
<tr>
<td>3</td>
<td>Grand Prize Team</td>
<td>0.6571</td>
<td>9.91</td>
<td>2009-07-24 13:07:49</td>
</tr>
<tr>
<td>4</td>
<td>Opera Solutions and Vandelay United</td>
<td>0.6573</td>
<td>9.89</td>
<td>2009-07-25 20:05:52</td>
</tr>
<tr>
<td>5</td>
<td>Vandelay Industries I</td>
<td>0.6579</td>
<td>9.83</td>
<td>2009-07-26 02:49:53</td>
</tr>
<tr>
<td>6</td>
<td>PragmaticTheory</td>
<td>0.6582</td>
<td>9.80</td>
<td>2009-07-12 15:09:53</td>
</tr>
<tr>
<td>7</td>
<td>BellKor in BigChaos</td>
<td>0.6590</td>
<td>9.71</td>
<td>2009-07-26 12:57:25</td>
</tr>
<tr>
<td>8</td>
<td>Dace</td>
<td>0.6603</td>
<td>9.58</td>
<td>2009-07-24 17:18:43</td>
</tr>
<tr>
<td>9</td>
<td>Opera Solutions</td>
<td>0.6611</td>
<td>9.49</td>
<td>2009-07-26 18:02:08</td>
</tr>
<tr>
<td>10</td>
<td>BellKor</td>
<td>0.6612</td>
<td>9.48</td>
<td>2009-07-26 17:19:11</td>
</tr>
<tr>
<td>11</td>
<td>BigChaos</td>
<td>0.6613</td>
<td>9.47</td>
<td>2009-06-23 23:06:52</td>
</tr>
<tr>
<td>12</td>
<td>Feeds2</td>
<td>0.6613</td>
<td>9.47</td>
<td>2009-07-24 20:06:46</td>
</tr>
</tbody>
</table>

**Grand Prize - RMSE <= 0.8563**

**Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Team Name</th>
<th>Best Score</th>
<th>% Improvement</th>
<th>Last Submit Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>xiangliang</td>
<td>0.6633</td>
<td>9.26</td>
<td>2009-07-21 02:04:40</td>
</tr>
<tr>
<td>14</td>
<td>Gravity</td>
<td>0.6634</td>
<td>9.25</td>
<td>2009-07-26 15:58:34</td>
</tr>
<tr>
<td>15</td>
<td>Cest</td>
<td>0.6642</td>
<td>9.17</td>
<td>2009-07-25 17:42:38</td>
</tr>
<tr>
<td>16</td>
<td>Invisible Ideas</td>
<td>0.6644</td>
<td>9.14</td>
<td>2009-07-20 03:26:12</td>
</tr>
<tr>
<td>17</td>
<td>Just a guy in a garage</td>
<td>0.6650</td>
<td>9.08</td>
<td>2009-07-22 14:10:42</td>
</tr>
<tr>
<td>18</td>
<td>Craig Carmichael</td>
<td>0.6656</td>
<td>9.02</td>
<td>2009-07-25 16:00:54</td>
</tr>
<tr>
<td>19</td>
<td>J Dennis Su</td>
<td>0.6658</td>
<td>9.00</td>
<td>2009-03-11 09:41:54</td>
</tr>
<tr>
<td>20</td>
<td>armehill</td>
<td>0.6659</td>
<td>8.99</td>
<td>2009-04-16 06:29:35</td>
</tr>
</tbody>
</table>

**Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell**
<table>
<thead>
<tr>
<th>Rank</th>
<th>Team Name</th>
<th>Best Score</th>
<th>% Improvement</th>
<th>Last Submit Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The Ensemble</td>
<td>0.8553</td>
<td>10.10</td>
<td>2009-07-26 18:36:22</td>
</tr>
<tr>
<td>2</td>
<td>BellKor's Pragmatic Chaos</td>
<td>0.8554</td>
<td>10.09</td>
<td>2009-07-26 18:16:28</td>
</tr>
<tr>
<td>3</td>
<td>Grand Prize Team</td>
<td>0.8571</td>
<td>9.91</td>
<td>2009-07-24 13:07:49</td>
</tr>
<tr>
<td>4</td>
<td>Opera Solutions and Vandelay United</td>
<td>0.8573</td>
<td>9.89</td>
<td>2009-07-25 20:05:52</td>
</tr>
<tr>
<td>5</td>
<td>Vandelay Industries I</td>
<td>0.8579</td>
<td>9.83</td>
<td>2009-07-26 02:49:53</td>
</tr>
<tr>
<td>6</td>
<td>PragmaticTheory</td>
<td>0.8582</td>
<td>9.80</td>
<td>2009-07-12 15:09:53</td>
</tr>
<tr>
<td>7</td>
<td>BellKor in BigChaos</td>
<td>0.8590</td>
<td>9.71</td>
<td>2009-07-26 12:57:25</td>
</tr>
<tr>
<td>8</td>
<td>Face</td>
<td>0.8603</td>
<td>9.58</td>
<td>2009-07-24 17:16:43</td>
</tr>
<tr>
<td>9</td>
<td>Opera Solutions</td>
<td>0.8612</td>
<td>9.48</td>
<td>2009-07-26 17:19:11</td>
</tr>
<tr>
<td>10</td>
<td>BellKor</td>
<td>0.8612</td>
<td>9.48</td>
<td>2009-07-26 17:19:11</td>
</tr>
<tr>
<td>11</td>
<td>BigChaos</td>
<td>0.8613</td>
<td>9.47</td>
<td>2009-06-23 23:06:52</td>
</tr>
<tr>
<td>12</td>
<td>Feeds2</td>
<td>0.8613</td>
<td>9.47</td>
<td>2009-07-24 20:06:46</td>
</tr>
<tr>
<td>13</td>
<td>xiangliang</td>
<td>0.8633</td>
<td>9.26</td>
<td>2009-07-21 02:04:40</td>
</tr>
<tr>
<td>14</td>
<td>Gravity</td>
<td>0.8634</td>
<td>9.25</td>
<td>2009-07-26 15:58:34</td>
</tr>
<tr>
<td>15</td>
<td>Ces</td>
<td>0.8642</td>
<td>9.17</td>
<td>2009-07-25 17:42:38</td>
</tr>
<tr>
<td>16</td>
<td>Invisible Ideas</td>
<td>0.8644</td>
<td>9.14</td>
<td>2009-07-20 03:26:12</td>
</tr>
<tr>
<td>17</td>
<td>Just a guy in a garage</td>
<td>0.8660</td>
<td>9.08</td>
<td>2009-07-22 14:10:42</td>
</tr>
<tr>
<td>18</td>
<td>Craig Carmichael</td>
<td>0.8666</td>
<td>9.02</td>
<td>2009-07-25 16:00:54</td>
</tr>
<tr>
<td>19</td>
<td>J Dennis Su</td>
<td>0.8658</td>
<td>9.00</td>
<td>2009-03-11 09:41:54</td>
</tr>
<tr>
<td>20</td>
<td>arnemilli</td>
<td>0.8659</td>
<td>8.99</td>
<td>2009-04-16 06:29:35</td>
</tr>
</tbody>
</table>
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 Scoring and Judging

- Leaders on test set are contacted and submit their code and documentation (mid-August)

- Judges review documentation and inform winners that they have won $1 million prize (late August)

- Considerable speculation in press and blogs about which team has actually won

- News conference scheduled for Sept 21st in New York to announce winner and present $1 million check
### Leaderboard

Showing Test Score. [Click here to show quiz score](#)

Display top 20 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>0.8563</td>
<td>9.90</td>
<td>2009-07-10 21:24:40</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</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>2009-04-07 12:33:59</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>0.8624</td>
<td>9.46</td>
<td></td>
</tr>
</tbody>
</table>

**Grand Prize** - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos

**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               | 0.8666 | 9.00 | 2009-03-21 16:20:50 |

**Progress Prize 2007** - RMSE = 0.8723 - Winning Team: KorBell
## Netflix Prize Leaderboard

### Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos

<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>0.8562</td>
<td>9.94</td>
<td>2009-07-18 24:42:46</td>
</tr>
<tr>
<td>4</td>
<td>Opera Solutions and Vandyel 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>
</tbody>
</table>

### Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos

<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>13</td>
<td>xianglang</td>
<td>0.8642</td>
<td>9.27</td>
<td>2009-07-15 14:53:22</td>
</tr>
<tr>
<td>14</td>
<td>Gravity</td>
<td>0.8643</td>
<td>9.26</td>
<td>2009-04-22 18:31:32</td>
</tr>
<tr>
<td>15</td>
<td>Ces</td>
<td>0.8651</td>
<td>9.18</td>
<td>2009-06-21 19:24:53</td>
</tr>
<tr>
<td>16</td>
<td>Invisible Ideas</td>
<td>0.8653</td>
<td>9.15</td>
<td>2009-07-15 15:53:04</td>
</tr>
<tr>
<td>17</td>
<td>Just a guy in a garage</td>
<td>0.8662</td>
<td>9.06</td>
<td>2009-05-24 10:02:54</td>
</tr>
<tr>
<td>18</td>
<td>J Dennis Su</td>
<td>0.8666</td>
<td>9.02</td>
<td>2009-03-07 17:16:17</td>
</tr>
<tr>
<td>19</td>
<td>Craig Carmichael</td>
<td>0.8666</td>
<td>9.02</td>
<td>2009-07-25 16:00:54</td>
</tr>
<tr>
<td>20</td>
<td>acmehill</td>
<td>0.8666</td>
<td>9.00</td>
<td>2009-03-21 16:20:50</td>
</tr>
</tbody>
</table>

### Progress Prize 2007 - RMSE = 0.8723 - Winning Team: KorBell

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Note: The % Improvement column is calculated as 100 * (Best Test Score - RMSE) / RMSE.
Million Dollars Awarded Sept 21st 2009
Lessons Learned
Lessons Learned

• Scale 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
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.

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....”
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**

<table>
<thead>
<tr>
<th>Forum</th>
<th>Topics</th>
<th>Posts</th>
<th>Last post</th>
</tr>
</thead>
<tbody>
<tr>
<td>Important Announcements</td>
<td>5</td>
<td>151</td>
<td>Today 04:29:38 by Yehudakoren</td>
</tr>
<tr>
<td>Registration Problems</td>
<td>1</td>
<td>1</td>
<td>2006-10-05 08:37:53 by prizemaster</td>
</tr>
<tr>
<td>Administrivia</td>
<td>3</td>
<td>43</td>
<td>2009-06-22 09:23:04 by dale5351</td>
</tr>
<tr>
<td>Prize and Forum FAQ</td>
<td>15</td>
<td>18</td>
<td>2009-03-24 10:18:36 by prizemaster</td>
</tr>
<tr>
<td>Request for new Category or Forum</td>
<td>18</td>
<td>40</td>
<td>2008-04-29 20:50:19 by filmmakershelpl</td>
</tr>
</tbody>
</table>

**Awarded Prizes**

<table>
<thead>
<tr>
<th>Forum</th>
<th>Topics</th>
<th>Posts</th>
<th>Last post</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grand Prize</td>
<td>1</td>
<td>14</td>
<td>2009-10-09 12:18:23 by statistician</td>
</tr>
<tr>
<td>Progress Prize 2008</td>
<td>2</td>
<td>17</td>
<td>2009-03-18 02:40:53 by CS1</td>
</tr>
<tr>
<td>Progress Prize 2007</td>
<td>5</td>
<td>29</td>
<td>2008-10-06 06:51:51 by dinc3r</td>
</tr>
</tbody>
</table>

**Questions (and answers)**

<table>
<thead>
<tr>
<th>Forum</th>
<th>Topics</th>
<th>Posts</th>
<th>Last post</th>
</tr>
</thead>
</table>
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
Why Collaboration?

Technical Reasons

- Realization that combining many different models and techniques always produced small but systematic improvements
  (Statistical theory supports this....)

- “Teaming” was strategically attractive

- Particularly for the “end-game” (summer 2009), teaming was quite critical in terms of who won the competition
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?
Did Netflix Violate Subscribers’ Privacy? Lawsuit Says Yes

By Ashby Jones

A potentially very interesting lawsuit was filed in San Jose, Calif., on Thursday concerning a closeted lesbian’s privacy and Netflix, the movie rental company.

The woman, an Ohio resident who, for purposes of the lawsuit has requested anonymity, sued Netflix alongside others hoping to sue on behalf of a potential class, claiming that the company divulged personal information which allowed thousands to discern her sexual preference. Click here for the story, from Wired; here for the complaint.

Here’s the backstory. In 2006, Netflix initiated a contest in which it would pay $1 million to the first person or team of people to make certain improvements to its recommendation system —
Links to additional information

- Netflix prize page (FAQs, rules, forum, etc)
  
  http://www.netflixprize.com/

- Page with links to articles, blogs, etc
  