Recommender Systems

from IERG 4030 by Prof. Wing C. Lau
Acknowledgements

- The slides used in this chapter are adapted from:
  - CS246 Mining Massive Data-sets, by Jure Leskovec, Stanford University.
  - Some slides and plots borrowed from Yehuda Koren, Robert Bell and Padhraic Smyth with the author’s permission. All copyrights belong to the original author of the material.
Content-based Systems & Collaborative Filtering
High Dimensional Data

- High Dimensional Data
  - Locality sensitive hashing
  - Clustering
  - Dimensionality reduction
- Graph Data
  - Community Detection
  - Spam Detection
- Infinite Data
  - Filtering
  - Streams
  - Web advertising
  - Queries on streams
- Machine Learning
  - Decision Trees
  - Perceptron, kNN
- Apps
  - Recommender systems
  - Association Rules
  - Duplicate document detection
Example: Recommender Systems

- **Customer X**
  - Buys Metallica CD
  - Buys Megadeth CD

- **Customer Y**
  - Does search on Metallica
  - Recommender system suggests Megadeth from data collected about customer X
Recommendations

Examples:
- amazon.com
- Pandora
- StumbleUpon
- del.icio.us
- movielens
  helping you find the right movies
- last.fm
  the social music revolution
- Google
- News
- YouTube
- XBOX
- LIVE
The Long Tail

Sources: Erik Brynjolfsson and Jeffrey Hu, MIT, and Michael Smith, Carnegie Mellon; Barnes & Noble; Netflix; RealNetworks.
Physical vs. Online

Read http://www.wired.com/wired/archive/12.10/tail.html to learn more!
Formal Model

- \( X = \) set of Customers
- \( S = \) set of Items

Utility function \( u: X \times S \rightarrow R \)

- \( R = \) set of ratings
- \( R \) is a totally ordered set
- e.g., 0-5 stars, real number in \([0,1]\)
## Utility Matrix

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>0.5</td>
<td>0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td>0.2</td>
<td>1</td>
<td></td>
<td>0.4</td>
</tr>
<tr>
<td>David</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Key Problems

○ (1) Gathering “known” ratings for matrix
  ● How to collect the data in the utility matrix

○ (2) Extrapolate unknown ratings from the known ones
  ● Mainly interested in high unknown ratings
    • We are not interested in knowing what you don’t like but what you like

○ (3) Evaluating extrapolation methods
  ● How to measure success/performance of recommendation methods
(1) Gathering Ratings

- **Explicit**
  - Ask people to rate items
  - Doesn’t work well in practice – people can’t be bothered

- **Implicit**
  - Learn ratings from user actions
    - E.g., purchase implies high rating
  - What about low ratings?
Key problem: matrix $U$ is sparse

- Most people have not rated most items
- Cold start:
  - New items have no ratings
  - New users have no history

Three approaches to recommender systems:

1) Content-based
2) Collaborative Filtering
   - Memory-based
     - User-based Collaborative Filtering
     - Item-based Collaborative Filtering
   - Latent factor based
Content-based Recommender Systems
Content-based Recommendations

- **Main idea:** Recommend items to customer $x$ similar to previous items rated highly by $x$

---

**Example:**

- **Movie recommendations**
  - Recommend movies with same actor(s), director, genre, …

- **Websites, blogs, news**
  - Recommend other sites with “similar” content
Plan of Action

Item profiles

likes

recommend

match

Red
Circles
Triangles

User profile

11/6/2019
Item Profiles

- For each item, create an item profile

- Profile is a set (vector) of features
  - **Movies**: author, title, actor, director,…
  - **Text**: Set of “important” words in document

- How to pick important features?
  - Usual heuristic from text mining is **TF-IDF**
    (Term frequency * Inverse Doc Frequency)
    - Term … Feature
    - Document … Item
Sidenote: TF-IDF

\( f_{ij} = \text{frequency of term (feature) } i \text{ in doc (item) } j \)

\[
TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}
\]

\( n_i = \text{number of docs that mention term } i \)

\( N = \text{total number of docs} \)

\[
IDF_i = \log \frac{N}{n_i}
\]

**Note:** we normalize TF to discount for “longer” documents

TF-IDF score: \( w_{ij} = TF_{ij} \times IDF_i \)

**Doc profile** = set of words with highest TF-IDF scores, together with their scores
User Profiles and Prediction

- **User profile possibilities:**
  - Weighted average of rated item profiles
  - **Variation:** weight by difference from average rating for item
  - ...

- **Prediction heuristic:**
  - Given user profile $\mathbf{x}$ and item profile $\mathbf{i}$, estimate
    \[ u(x, i) = \cos(x, i) = \frac{x \cdot i}{||x|| \cdot ||i||} \]
Pros: Content-based Approach

- **+: No need for data on other users**
  - No cold-start or sparsity problems
- **+: Able to recommend to users with unique tastes**
- **+: Able to recommend new & unpopular items**
  - No first-rater problem
- **+: Able to provide explanations**
  - Can provide explanations of recommended items by listing content-features that caused an item to be recommended
Cons: Content-based Approach

- Finding the appropriate features is hard
  - E.g., images, movies, music

- Overspecialization
  - Never recommends items outside user’s content profile
  - People might have multiple interests
  - Unable to exploit quality judgments of other users

- Recommendations for new users
  - How to build a user profile?
Collaborative Filtering
Collaborative filtering

- Recommend items based on past transactions of users
- Analyze relations between users and/or items
- Specific data characteristics are irrelevant
  - Domain-free: user/item attributes are not necessary
  - Can identify elusive aspects

![Amazon recommendation example](amazon.com)

*Customers who bought items in your Recent History also bought:*
Collaborative Filtering (CF)

Memory-based
(e.g., k-nearest neighbors)

Model-based
(e.g., matrix factorization)

Figure 1. The user-oriented neighborhood method. Joe likes the three movies on the left. To make a prediction for him, the system finds similar users who also liked those movies, and then determines which other movies they liked. In this case, all three liked Saving Private Ryan, so that is the first recommendation. Two of them liked Dune, so that is next, and so on.

Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

http://research.yahoo.com/pub/2859
Example of Memory-based Collaborative Filtering: User-User Collaborative Filtering

1. Consider user \( x \)

2. Find set \( N \) of other users whose ratings are “similar” to \( x \)’s ratings, e.g. using K-nearest neighbors (KNN)

3. Recommend items to \( x \) based on the weighted ratings of items by users in \( N \)
Similar Users

- Let $r_x$ be the vector of user $x$’s ratings

- **Jaccard similarity measure**
  - **Problem:** Ignores the value of the rating

- **Cosine similarity measure**
  - $\text{sim}(x, y) = \cos(r_x, r_y) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$
  - **Problem:** Treats missing ratings as “negative”

- **Pearson correlation coefficient**
  - $\bar{r}_x \cdot \bar{r}_y \ldots \text{avg. rating of } x, y$
  - $I_{xy}$ is the set of items rated by both user $x$ and user $y$.
### Similarity Metric

- **Intuitively we want:** $\text{sim}(A, B) > \text{sim}(A, C)$
- **Jaccard similarity:** $\frac{1}{5} < \frac{2}{4}$
- **Cosine similarity:** $0.386 > 0.322$
  - Considers missing ratings as “negative”
  - **Solution:** subtract the (row) mean

#### Example

<table>
<thead>
<tr>
<th></th>
<th>HP1</th>
<th>HP2</th>
<th>HP3</th>
<th>TW</th>
<th>SW1</th>
<th>SW2</th>
<th>SW3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>5</td>
<td></td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

### Cosine Sim:

- **sim A,B vs. A,C:**
  - $0.092 > -0.559$

Notice cosine sim. is correlation when data is centered at 0.
Rating Predictions

- Let $r_x$ be the vector of user $x$’s ratings
- Let $N$ be the set of $k$ users most similar to $x$ who have rated item $i$

Prediction for item $i$ of user $x$:

- $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
- $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$

Shorthand: $s_{xy} = sim(x, y)$

- Other options?
- Many other tricks possible…
Another type of Memory-based Collaborative Filtering: Item-Item based Collaborative Filtering

- So far: User-user collaborative filtering

- Another view: Item-item
  
  - For item $i$, find other similar items
  
  - Estimate rating for item $i$ based on the target user’s ratings on items similar to item $i$
  
  - Can use same similarity metrics and prediction functions as in user-user model

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

- $s_{ij}$… similarity of items $i$ and $j$
- $r_{xj}$… rating of user $x$ on item $j$
- $N(i;x)$… set items rated by $x$ similar to $i$
Item-Item CF ($|N| = 2$)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- unknown rating
- rating between 1 to 5
### Item-Item CF (|N|=2)

#### Users

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>?</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Movies

- Estimate rating of movie 1 by user 5
### Item-Item CF ($|N|=2$)

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

Here we use Pearson correlation as similarity:
1) Subtract mean rating $m_i$ from each movie $i$
   $$m_i = \frac{(1+3+5+5+4)}{5} = 3.6$$
2) Compute cosine similarities between rows

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Sim(1,m)**
- 1.00
- -0.18
- 0.41
- -0.10
- -0.31
- 0.59

**Row 1:** $[-2.6, 0, -0.6, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]$
Compute similarity weights:
$s_{13}=0.41$, $s_{16}=0.59$
Item-Item CF (|N|=2)

Predict by taking weighted average:

\[ r_{15} = \frac{(0.41 \times 2 + 0.59 \times 3)}{(0.41 + 0.59)} = 2.6 \]
Common Practice for Item-Item Collaborative Filtering

- Define similarity $s_{ij}$ of items $i$ and $j$
- Select $K$ nearest neighbors (KNN): $N(i; x)$
  - Set of Items most similar to item $i$, that were rated by $x$
- Estimate rating $r_{xi}$ as the weighted average:

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

$N(i;x) = $ set of items similar to item $i$ that were rated by $x$
$s_{ij} = $ similarity of items $i$ and $j$
$r_{xj} = $ rating of user $x$ on item $j$
### Item-Item vs. User-User

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td></td>
<td>0.5</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td>0.9</td>
<td>1</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>David</td>
<td></td>
<td>1</td>
<td>0.4</td>
<td></td>
</tr>
</tbody>
</table>

- **In practice, it has been observed that item-item often works better than user-user**
- **Why?** Items are simpler, users have multiple tastes
Pros/Cons of Collaborative Filtering

- **+ Works for any kind of item**
  - No feature selection needed

- **- Cold Start:**
  - Need enough users in the system to find a match

- **- Sparsity:**
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items

- **- First rater:**
  - Cannot recommend an item that has not been previously rated
  - New items, Esoteric items

- **- Popularity bias:**
  - Cannot recommend items to someone with unique taste
  - Tends to recommend popular items
Hybrid Methods

- **Implement two or more different recommenders and combine predictions**
  - Perhaps using a linear model

- **Add content-based methods to collaborative filtering**
  - Item profiles for new item problem
  - Demographics to deal with new user problem
Remarks & Practical Tips

- Evaluation
- Error metrics
- Complexity / Speed
## Evaluation

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

<table>
<thead>
<tr>
<th></th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
## Evaluation

<table>
<thead>
<tr>
<th>users</th>
<th>movies</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<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>

Test Data Set
Evaluating Predictions

- **Compare predictions with known ratings**
  - **Root-mean-square error (RMSE)**
    - where \( \hat{x} \) is predicted, \( x \) is the true rating of \( x \) on \( i \)
  - **Precision at top 10**: 
    - % of those in top 10
  - **Rank Correlation**:
    - Spearman’s *correlation* between system’s and user’s complete rankings

- **Another approach: 0/1 model**
  - **Coverage**:
    - Number of items/users for which system can make predictions
  - **Precision** = \( \frac{TP}{TP + FP} \)
  - **Accuracy** = \( \frac{TP+TN}{TP + FP + TN + FN} \)
  - **Receiver Operating characteristic (ROC) Curve**
    - Y-axis: True Positive Rates (TPR) ; X-axis False Positive Rates (FPR)
    - TPR (aka Recall) = \( \frac{TP}{P} = \frac{TP}{TP+FN} \) ;
    - FPR = \( \frac{FP}{N} = \frac{FP}{FP + TN} \)
    - See [https://en.wikipedia.org/wiki/Precision_and_recall](https://en.wikipedia.org/wiki/Precision_and_recall)
Problems with Error Measures

- **Narrow focus on accuracy sometimes misses the point**
  - Prediction Diversity
  - Prediction Context
  - Order of predictions

- **In practice, we care only to predict high ratings:**
  - RMSE might penalize a method that does well for high ratings and badly for others
Collaborative Filtering: Complexity

- Expensive step is finding $k$ most similar customers: $O(|X|)$
  - Recall that $X =$ set of customers in the system

- Too expensive to do at runtime
  - Could pre-compute using clustering as approx.

- Naïve pre-computation takes time $O(N \cdot |C|)$
  - $|C| =$ # of clusters = $k$ in the k-means ; $N =$ # of data points ;

- We already know how to do this!
  - Near-neighbor search in high dimensions (LSH)
  - Clustering
  - Dimensionality reduction
Tip: Add Data

- **Leverage all the data**
  - Don’t try to reduce data size in an effort to make fancy algorithms work
  - Simple methods on large data do best

- **Add more data**
  - e.g., add IMDB data on genres

- **More data beats better algorithms**

http://anand.typepad.com/datawocky/2008/03/more-data-usual.html
Recommender Systems: Latent Factor Models
Collaborative Filtering via Latent Factor Models (e.g., SVD)

- **Serious**
  - Amadeus
- **Funny**
  - Dumb and Dumber
- **Geared towards females**
  - The Princess Diaries
  -感性
- **Geared towards males**
  - Braveheart
  - 动作
- **Ocean's 11**
- **The Lion King**
- **Independence Day**
The Netflix Utility Matrix $R$

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

480,000 users

17,700 movies
Utility Matrix $R$: Evaluation

Matrix $R$

480,000 users

17,700 movies

Training Data Set

Test Data Set

SSE = $\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2$

True rating of user $x$ on item $i$

Predicted rating

RECSYS 52
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 will want the reconstruction error to be small on known ratings and we don’t care about the values on the missing ones

SVD: \( A = U \Sigma V^T \)
Ratings as Products of Factors

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

\[
\hat{r}_{xi} = q_i \cdot p_x^T = \sum_f q_{if} \cdot p_{xf}
\]

$q_i =$ 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 $i$?

\[
\hat{r}_{xi} = q_i \cdot p^T_x = \sum_f q_{if} \cdot p_{xf}
\]

$\hat{r}_{xi}$ = 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 $i$?

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

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

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

- Geared towards females
  - The Color Purple
  - Sense and Sensibility
- Serious
  - Amadeus
- Geared towards males
  - Braveheart
  - Lethal Weapon
- Funny
  - The Lion King
  - Ocean's 11
  - Independence Day
  - Dumb and Dumber
- Serious
  - Braveheart
  - Lethal Weapon
  - Independence Day
  - Dumb and Dumber
Recap: SVD

- **Remember SVD:**
  - \( A \): Input data matrix
  - \( U \): Left singular vecs
  - \( V \): Right singular vecs
  - \( \Sigma \): Singular values

- **SVD gives minimum reconstruction error (SSE!):**
  \[
  \min_{U,V,\Sigma} \sum_{ij} (A_{ij} - [U\Sigma V^T]_{ij})^2
  \]

So in our case, “SVD” on Netflix data: \( R \approx Q \cdot P^T \)

- \( A = R, \quad Q = U, \quad P^T = \Sigma V^T \)

- But, we are not done yet! \( R \) has missing entries!
Latent Factor Models

- SVD isn’t defined when entries are missing!
- Use specialized methods to find $P$, $Q$
  - $\min_{P,Q} \sum_{(i,x) \in \mathcal{R}} (r_{xi} - q_i \cdot p_x^T)^2$
  - Note:
    - We don’t require cols of $P$, $Q$ to be orthogonal/unit length
    - $P$, $Q$ map users/movies to a latent space
    - The most popular model among Netflix contestants
Dealing with Missing Entries

- Want to minimize SSE for unseen test data
- Idea: **Minimize SSE on training data**
  - Want large $f$ (# of factors) to capture all the signals
  - But, SSE on test data begins to rise for $f > 2$
- **Regularization is needed to avoid Overfitting!**
  - Allow rich model where there are sufficient data
  - Shrink aggressively where data are scarce

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

$\lambda$... regularization parameter

"error"  "length"
Recommendations via Latent Factor Models (e.g., SVD++ by the [Bellkor Team])

- The Color Purple
- The Princess Diaries
- Sense and Sensibility
- The Lion King
- Independence Day
- Amadeus
- Braveheart
- Ocean's 11
- Lethal Weapon
- The Princess Diaries
- Dumb and Dumber

Geared towards females

Geared towards males

serious

escapist
Dealing with Missing Entries

- Want to minimize SSE for unseen test data
- **Idea**: Minimize SSE on training data
  - Want large $f$ (# of factors) to capture all the signals
  - But, SSE on test data begins to rise for $f > 2$
- Regularization is needed!
  - Allow rich model where there are sufficient data
  - Shrink aggressively where data are scarce

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

$\lambda$… regularization parameter

“error”

“length”
The Effect of Regularization

\[
\min \sum_{r_{xi} - q_i p_x^T}^2 + \lambda \left[ \sum_x \|p_x\|^2 + \sum_i \|q_i\|^2 \right]
\]

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

RECSYS 64
The Effect of Regularization

\[
\min_{P,Q} \sum_{i \in \text{training}} (r_{xi} - q_i^T p_i)^2 + \lambda \left[ \sum_i ||p_i||^2 + \sum_i ||q_i||^2 \right]
\]

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

\[
\min_{P,Q} \sum_{\text{training}} (r_{xi} - q_i p_i^T)^2 + \lambda \left[ \sum_i \|p_i\|^2 + \sum_i \|q_i\|^2 \right]
\]

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

\[
\min_{P,Q} \sum_{\text{training}} (r_{xi} - q_i p_i^T)^2 + \lambda \left[ \sum_x \|p_i\|^2 + \sum_i \|q_i\|^2 \right]
\]

\[
\min_{\text{factors}} \text{“error”} + \lambda \text{“length”}
\]
Use Gradient Descent to search for the optimal settings

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

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

- **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 $Q$:
      $$\nabla Q = [\nabla q_{if}] \quad \text{and} \quad \nabla q_{if} = \sum_{x,i} -2(r_{xi} - q_i p_x^T)p_{xf} + 2\lambda q_{if}$$
    - Here $q_{if}$ is entry $f$ of row $q_i$ of matrix $Q$
  - **Observation**: Computing gradients is slow!
Degression to the lecture notes of Regression and Gradient Descent by Andrew Ng’s Machine Learning course from Coursera
(Batch) Gradient Descent

○ **Want to find matrices $P$ and $Q$:**

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

○ **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 $Q$:
    $$
    \nabla Q = [\nabla q_{if}] \quad \text{and} \quad \nabla q_{if} = \sum_{x,i} -2 (r_{xi} - q_i p_x^T) p_{xf} + 2\lambda q_{if}
    $$
    - Here $q_{if}$ is entry $f$ of row $q_i$ of matrix $Q$
- **Observation:** Computing gradients is slow!

---

*How to compute gradient of a matrix? Compute gradient of every element independently!*
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 \( Q \)
  - \( Q = Q - \sum x,i \nabla Q(r_{xi}) \)
  - **Idea:** Instead of evaluating gradient over all ratings evaluate it for each individual rating and make a step

- **GD:** \( Q \leftarrow Q - \eta \sum r_{xi} \nabla Q(r_{xi}) \)

- **SGD:** \( Q \leftarrow Q - \eta Q(r_{xi}) \)
  - **Faster convergence!**
    - Need more steps but each step is computed much faster
SGD vs. GD

- **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!
Stochastic Gradient Descent

- **Stochastic gradient descent:**
  - Initialize $P$ and $Q$ (using SVD, pretend missing ratings are 0)
  - Then iterate over the ratings (multiple times if necessary) and update factors:
    For each $r_{xi}$:
    - $\varepsilon_{xi} = r_{xi} - q_i \cdot p_x^T$ (derivative of the “error”)
    - $q_i \leftarrow q_i + \eta (\varepsilon_{xi} p_x - \lambda q_i)$ (update equation)
    - $p_x \leftarrow p_x + \eta (\varepsilon_{xi} q_i - \lambda p_x)$ (update equation)

- **2 for loops:**
  - For until convergence:
    - For each $r_{xi}$
      - Compute gradient, do a “step”
Summary: Recommendations via Optimization

- **Goal:** Make good recommendations
  - Quantify goodness using **SSE:**
    So, **Lower SSE means better recommendations**
  - We want to make good recommendations on items that some user has not yet seen.
  - Let’s set values *for P and Q* such that they work well on known (user, item) ratings
  - And hope these values for *P and Q* will predict well the unknown ratings
- This is the a case where we apply **Optimization methods**
Backup Slides
The Netflix Challenge: 2006-09
We Know What You Ought To Be Watching This Summer
Welcome!

The Netflix Prize seeks to substantially improve the accuracy of predictions about how much someone is going to love a movie based on their movie preferences. Improve it enough and you win one (or more) Prizes. Winning the Netflix Prize improves our ability to connect people to the movies they love.

Read the Rules to see what is required to win the Prizes. If you are interested in joining the quest, you should register a team.

You should also read the frequently-asked questions about the Prize. And check out how various teams are doing on the Leaderboard.

Good luck and thanks for helping!
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
## Movie rating data

### Training data

<table>
<thead>
<tr>
<th>user</th>
<th>movie</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>213</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>345</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>123</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>768</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>76</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>45</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>568</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>342</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>234</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>76</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>56</td>
<td>4</td>
</tr>
</tbody>
</table>

### Test data

<table>
<thead>
<tr>
<th>user</th>
<th>movie</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>62</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>96</td>
<td>?</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>?</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>?</td>
</tr>
<tr>
<td>3</td>
<td>47</td>
<td>?</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>?</td>
</tr>
<tr>
<td>4</td>
<td>41</td>
<td>?</td>
</tr>
<tr>
<td>4</td>
<td>28</td>
<td>?</td>
</tr>
<tr>
<td>5</td>
<td>93</td>
<td>?</td>
</tr>
<tr>
<td>5</td>
<td>74</td>
<td>?</td>
</tr>
<tr>
<td>6</td>
<td>69</td>
<td>?</td>
</tr>
<tr>
<td>6</td>
<td>83</td>
<td>?</td>
</tr>
</tbody>
</table>
Overall rating distribution

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

<table>
<thead>
<tr>
<th>Title</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>
<tr>
<td>Forrest Gump</td>
<td>4.299</td>
<td>180736</td>
</tr>
<tr>
<td>Lord of the Rings: The Fellowship of the ring</td>
<td>4.433</td>
<td>147932</td>
</tr>
<tr>
<td>The Sixth Sense</td>
<td>4.325</td>
<td>149199</td>
</tr>
<tr>
<td>Indiana Jones and the Last Crusade</td>
<td>4.333</td>
<td>144027</td>
</tr>
</tbody>
</table>
Challenges

• Size of data
  – Scalability
  – Keeping data in memory

• Missing data
  – 99 percent missing
  – Very imbalanced

• Avoiding overfitting

• Test and training data differ significantly
The BellKor recommender system

- Use an ensemble of complementing predictors
- Two, half tuned models worth more than a single, fully tuned model
Extending Latent Factor Model to Include Biases
Modeling Biases and Interactions

- **μ** = overall mean rating
- **b_x** = bias of user \( x \)
- **b_i** = bias of movie \( i \)

**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
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

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

- Overall mean rating
- Bias for user \( x \)
- Bias for movie \( i \)
- User-Movie interaction

\textbf{Example:}

- Mean rating: \( \mu = 3.7 \)
- 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:
  \[ = 3.7 - 1 + 0.5 = 3.2 \]
Fitting the New Model

- **Solve:**

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

goodness of fit

\[
+ \lambda \left( \sum_i \| q_i \|^2 + \sum_x \| p_x \|^2 + \sum_x \| b_x \|^2 + \sum_i \| b_i \|^2 \right)
\]

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

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

- CF (no time bias)
- Basic Latent Factors
- Latent Factors w/ Biases

RMSE vs. Millions of parameters
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

Grand Prize: 0.8563
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:**
  \[ r_{xi} = \mu + b_x + b_i + q_i \cdot p_x^T \]

- **Add time dependence to biases:**
  \[ r_{xi} = \mu + b_x(t) + b_i(t) + q_i \cdot p_x^T \]
  - Make parameters \( b_u \) and \( b_i \) to depend on time
  - (1) Parameterize time-dependence by linear trends
    (2) Each bin corresponds to 10 consecutive weeks

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

\[ b_i(t) = b_i + b_{i,Bin}(t) \]

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

![Graph showing the impact of adding temporal effects on RMSE with varying millions of parameters. The graph compares different models:
- CF (no time bias)
- Basic Latent Factors
- CF (time bias)
- Latent Factors w/ Biases
- + Linear time factors
- + Per-day user biases
- + CF

The x-axis represents millions of parameters, and the y-axis represents the RMSE. The graph illustrates how adding temporal effects reduces the RMSE as the number of parameters increases.]
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
Solution of BellKor's Pragmatic Chaos

All developed CF models
- BRISMF
- MF1 NSVDD
- SVD-Time
- SVD-AUF
- Movie KNN
- V KNN + time
- NSVD1
- Baseline
- DRBM
- SVD++
- ISVD2
- User KNN
- Classif. ModeKNN
- 1...5 Asym. 1/2/3

Latent User and Movie Features

Probe Blending
- 200 blends

Probe Blending
- 30 blends

Linear Blend 
10.09 % improvement

approx. 500 predictors
Effect of ensemble size

Error - RMSE

#Predictors
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 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**
  - 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

<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>6.00</td>
<td>2009-07-26 21:24:36</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>2009-04-07 12:33:59</td>
</tr>
<tr>
<td>11</td>
<td>Opera Solutions</td>
<td>0.8623</td>
<td>9.46</td>
<td>2009-07-24 00:34:07</td>
</tr>
<tr>
<td>12</td>
<td>BellKor</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>xiangliang</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>
</table>

---

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

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

---

Click here to show quiz score.
Million $ Awarded Sept 21st 2009
Further reading

- Y. Koren, Collaborative filtering with temporal dynamics, KDD ’09
- http://www.the-ensemble.com/