Recommendation Systems: Part II

Prof. Srijan Kumar
http://cc.gatech.edu/~srijan
Announcements

• **Project:**
  – Final report rubric: released
  – Final presentation: details forthcoming
Recommendation Systems

- Content-based
- Collaborative Filtering
- Latent Factor Models
- Case Study: Netflix Challenge
- Deep Recommender Systems

Slide reference: Mining Massive Dataset
http://mmds.org/
Latent Factor Models

- These models learn **latent factors** to represent users and items from the rating matrix
  - Latent factors are not directly observable
  - These are derived from the data
- **Recall:** Network embeddings
- **Methods:**
  - Singular value decomposition (SVD)
  - Principal Component Analysis (PCA)
  - Eigendecomposition
Latent Factors: Example

• **Embedding axes** are a type of latent factors

• **In a user-movie rating matrix:**

• **Movie latent factors** can represent axes:
  – Comedy vs drama
  – Degree of action
  – Appropriateness to children

• **User latent factors** will measure a user’s affinity towards corresponding movie factors
Latent Factors: Example

- **Geared towards females**
  - The Princess Diaries
  - Sense and Sensibility

- **Geared towards males**
  - The Lion King
  - Braveheart

- **Factor 1**
  - Ocean's 11
  - Lethal Weapon

- **Factor 2**
  - Funny
  - Independence Day
  - Dumb and Dumber

- **Serious**
  - Amadeus

Srijan Kumar, Georgia Tech, CSE6240 Spring 2020: Web Search and Text Mining
SVD

- **SVD**: SVD decomposes an input matrix into multiple factor matrices
  - \( A = U \Sigma V^T \)
  - Where,
  - \( A \): Input data matrix
  - \( U \): Left singular vecs
  - \( V \): Right singular vecs
  - \( \Sigma \): Singular values
SVD

• SVD gives **minimum reconstruction error** (Sum of Squared Errors):

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

• **SSE** and **RMSE** are monotonically related:
  – \( RMSE = \frac{1}{c} \sqrt{SSE} \)  \( \Rightarrow \) SVD is minimizing RMSE

• **Complication:** The sum in SVD error term is over all entries. But our \( R \) has missing entries.
  – **Solution:** no-rating in interpreted as zero-rating.
SVD on Rating Matrix

- “SVD” on rating data: \( R \approx Q \cdot P^T \)
- Each row of \( Q \) represents an item
- Each column of \( P \) represents a user
Ratings as Products of Factors

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

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

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

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

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

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

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

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

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

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

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

$p_x = \text{column } x \text{ of } P^\top$
Latent Factor Models: Example

Movies plotted in two dimensions. Dimensions have meaning.

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

- Funny
  - The Lion King
  - Independence Day

- Serious
  - Amadeus
  - Ocean’s 11

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

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

Users fall in the same space, showing their preferences.

Geared towards females

Geared towards males

Serious
Amadeus

Funny
Dumb and Dumber

Factor 1

Factor 2

The Princess Diaries

The Lion King

Ocean’s 11

Independence Day

Dumb and Dumber

Lethal Weapon

Braveheart
SVD: Problems

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

- This is a classical example of **overfitting:**
  - With too much freedom (too many free parameters) the model starts fitting noise
  - Model fits too well the training data and thus not generalizing well to unseen test data
Preventing Overfitting

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

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

\(\lambda_1, \lambda_2 \ldots\) user set regularization parameters

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

The Color Purple
Sense and Sensibility
The Princess Diaries

The Lion King
Independence Day
Dumb and Dumber

Factor 1
Factor 2

Geared towards females
serious
funny
Geared towards males
**Modeling Biases and Interactions**

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

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

\[
\mu = \text{overall mean rating}
\]
\[
b_x = \text{bias of user } x
\]
\[
b_i = \text{bias of movie } i
\]
Baseline Predictor

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

  – Rating scale of user $x$
  – Values of other ratings user gave recently (day-specific mood, anchoring, multi-user accounts)

  +

  – (Recent) popularity of movie $i$
  – Selection bias; related to number of ratings user gave on the same day (“frequency”)
Putting It All Together

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

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

**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:**

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

- **goodness of fit**

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

- **regularization**

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

- **Stochastic gradient decent to find parameters**

  - Note: Both biases \( b_x, b_i \) as well as interactions \( q_i, p_x \) are treated as parameters (we estimate them)
Recommendation Systems

- Content-based
- Collaborative Filtering
- Latent Factor Models

Case Study: Netflix Challenge
- Deep Recommender Systems
## Case Study: Netflix Prize

![Netflix Prize Leaderboard](image)

<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>9.80%</td>
<td>2009-06-25 22:15:51</td>
</tr>
<tr>
<td>3</td>
<td>BellKor in BigChaos</td>
<td>0.8590</td>
<td>9.71%</td>
<td>2009-05-13 08:14:09</td>
</tr>
<tr>
<td>4</td>
<td>Grand Prize Team</td>
<td>0.8593</td>
<td>9.68%</td>
<td>2009-06-12 08:20:24</td>
</tr>
<tr>
<td>5</td>
<td>Dace</td>
<td>0.8604</td>
<td>9.56%</td>
<td>2009-04-22 05:57:03</td>
</tr>
<tr>
<td>6</td>
<td>BigChaos</td>
<td>0.8613</td>
<td>9.47%</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 18:31:32</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>xvector</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>2009-06-28 22:51:55</td>
</tr>
<tr>
<td>15</td>
<td>Ces</td>
<td>0.8642</td>
<td>9.17%</td>
<td>2009-06-26 14:34:14</td>
</tr>
</tbody>
</table>

**Grand Prize - RMSE <= 0.8563**

**Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos**
Case Study: The Netflix Prize

**Competition**
- **Task:** Reduce RMSE
- 2,700+ teams
- **$1 million** prize for 10% improvement on Netflix

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

- **Basic Collaborative filtering:** 0.94
- **Collaborative filtering++:** 0.91
- **Latent factors:** 0.90
- **Latent factors + Biases:** 0.89
Case Study: The Netflix Prize

- **Training data**
  - 100 million ratings, 480,000 users, 17,770 movies
  - 6 years of data: 2000-2005

- **Test data**
  - Last few ratings of each user (2.8 million)
  - **Evaluation criterion**: Root Mean Square Error (RMSE)
    \[
    \text{RMSE} = \frac{1}{|R|} \sqrt{\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2}
    \]
  - Netflix’s system RMSE: 0.9514
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
Modeling Local & Global Effects

• **Global:**
  – Mean movie rating: **3.7 stars**
  – *The Sixth Sense* is **0.5** stars above avg.
  – Joe rates **0.2** stars below avg.
    ⇒ **Baseline estimation:**
    Joe will rate *The Sixth Sense* 4 stars

• **Local neighborhood (CF/NN):**
  – Joe didn’t like related movie *Signs*
  – ⇒ **Final estimate:**
    Joe will rate *The Sixth Sense* **3.8 stars**
Interpolation Weights

• **So far:** \( \hat{r}_{xi} = b_{xi} + \sum_{j \in N(i; x)} w_{ij} (r_{xj} - b_{xj}) \)
  
  – Weights \( w_{ij} \) derived based on their role; no use of an arbitrary similarity measure (\( w_{ij} \neq s_{ij} \))
  
  – Explicitly account for interrelationships among the neighboring movies

• **Next: Latent factor model**
  
  – Extract “regional” correlations
Temporal Biases Of Users

• Sudden rise in the average movie rating (early 2004)
  – Improvements in Netflix
  – GUI improvements
  – Meaning of rating changed

• Movie age
  – Users prefer new movies without any reasons
  – Older movies are just inherently better than newer ones

Y. Koren, Collaborative filtering with temporal dynamics, KDD ’09
Temporal Biases & Factors

• **Original model:**
  \[ r_{xi} = \mu + b_x + b_i + q_i \cdot p_x \]

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

• **Add temporal dependence to factors**
  – \( p_x(t) = \) user preference vector on day \( t \)
Adding Temporal Effects

- **RMSE**
- **Millions of parameters**
- **CF (no time bias)**
- **Basic Latent Factors**
- **CF (time bias)**
- **Latent Factors w/ Biases**
- **+ Linear time factors**
- **+ Per-day user biases**
- **+ CF**
Case Study: The Netflix Prize

- New update: Added time
- But still no prize!
- What to do next?

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
Grand Prize: 0.8563

New update: Added time
But still no prize!
What to do next?
The big picture

Solution of BellKor's Pragmatic Chaos

All developed CF models

- BRISMF
- MF1
- NSVDD
- SVD-Time
- SBRAMF
- Split RBM
- BK3
- BK4
- BK1
- BK5-SVD++
- GTE
- Movie KNN
- V
- Baseline
- 1/2/3
- DRBM
- SVD++
- ISVD
- ISVD2
- MF2
- Integrated M.
- RBM
- User KNN
- Classif.
- ModeKNN 1...5
- Asym. 1/2/3

Latent User and Movie Features

approx. 500 predictors

Probe Blending

200 blends

Linear Blend 10.09 % improvement

Probe Blending

30 blends
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 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….
<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>0.8582</td>
<td></td>
<td>2009-07-10 12:41:01</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>9.47</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>
<tr>
<td></td>
<td><strong>Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<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>
<tr>
<td></td>
<td><strong>Progress Prize 2007 - RMSE = 0.8723 - Winning Team: KorBell</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
$1M Awarded Sept 21\textsuperscript{st} 2009
Recommendation Systems

- Content-based
- Collaborative Filtering
- Latent Factor Models
- Case Study: Netflix Challenge
- Deep Recommender Systems
Deep Recommender Systems

• How can deep learning advance recommendation systems?
• Simple way for content-based models: Use CNNs, LSTMs for generate image and text features of items
Deep Recommender Systems

• But how can DL be used for tasks and methods at the core of recommendation systems?
  – For collaborative filtering?
  – For latent factor models?
  – For temporal dynamics?
  – Some new techniques?
Why Deep Learning Techniques

Pros:

- Capture non-linearity well
- Non-manual representation learning
- Efficient sequence modeling
- Somewhat flexible and easy to retrain

Cons:

- Lack of interpretability
- Large data requirements
- Extensive hyper-parameter tuning
Applicable DL Techniques

Deep Learning methods:
• MLPs and AutoEncoders
• CNNs
• RNNs
• Adversarial Networks
• Attention models
• Deep reinforcement learning

How to use these methods to improve recommender systems?
Several Methods

• Neural Collaborative Filtering
• Recurrent Recommender Systems
• LatentCross
• Dynamic User Model: JODIE
Neural Collaborative Filtering

• Neural extensions of traditional recommender system

• **Input**: rating matrix, user profile and item features (optional)
  – If user/item features are unavailable, we can use one-hot vectors

• **Output**: User and item embeddings

• Traditional matrix factorization is a special case of NCF

• Reference: *Neural Collaborative Filtering*, He et al., WWW 2017
NCF Setup

- User feature vector: $s_{\text{user}}^u$
- Item feature vector: $s_{\text{item}}^i$
- User embedding matrix: $U$
- Item embedding matrix: $I$
- Neural network: $f$
- Neural network parameters: $\Theta$
- Predicted rating:

$$\hat{r}_{ui} = f(U^T \cdot s_{\text{user}}^u, V^T \cdot s_{\text{item}}^i | U, V, \Theta)$$
NCF Model Architecture

- Multiple layers of fully connected layers form the Neural CF layer.
- Output is a rating score $\hat{r}_{ui}$
- Real rating score is $r_{ui}$
NCF model: Loss function

- **Train** on the difference between predicted rating and the real rating
- **Use negative sampling** to reduce the negative data points
- **Loss =** cross-entropy loss

\[
\mathcal{L} = - \sum_{(u, i) \in \mathcal{O} \cup \mathcal{O}^-} r_{ui} \log \hat{r}_{ui} + (1 - r_{ui}) \log (1 - \hat{r}_{ui})
\]