Large Scale Collaborative Filtering Algorithms

Chih-Chao Ma
Department of Computer Science
National Taiwan University
Outline

- Introduction
- Singular Value Decomposition
- Post-Processing
- Experiments
- Conclusions
Outline

- Introduction
- Singular Value Decomposition
- Post-Processing
- Experiments
- Conclusions
Introduction

- Recommendation systems give people advices
  - Web shopping
  - News (Yahoo) and videos (Youtube)
- Collaborative Filtering
  - Make predictions by taste information
  - e.g. the ratings of products given by users
Preferences for \( m \) objects from \( n \) users

- \( V \) is the \textbf{sparse} matrix of the scores
  - Some users do not score some objects
    - \( V \in \mathbb{R}^{n \times m} \) with an indicator \( I \in \{0, 1\}^{n \times m} \)

- Predict the unknown scores in the matrix
  - Represented by another sparse matrix \( A \in \mathbb{R}^{n \times m} \)
Evaluation Criteria

- Performance are measured by RMSE (Root Mean Square Error)
  
  \[
  P \in \mathbb{R}^{n \times m} \text{ is the prediction matrix with indicator } J
  \]
  
  \[
  \text{RMSE}(P, A) = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{m} J_{ij}(A_{ij} - P_{ij})^2}{\sum_{i=1}^{n} \sum_{j=1}^{m} J_{ij}}}
  \]

- Give advices to users equally
  Regardless of the numbers of scores
  Uniform distribution on users for test data
The Netflix Prize

A contest held by Netflix
17,770 movies and 480,189 users
100,480,507 training scores (values given)
2,817,131 test scores (values unknown)
Select the 9 most recent scores for each user
Divided into probe, quiz, and test sets
Probe set is used for offline validation
Quiz and test sets are the test data
Related Work

- **KNN (K-Nearest Neighbors)**
  Define the similarity between users or objects
  Predict unknown score by other “similar” ones

- **SVD (Singular Value Decomposition)**
  Find the features of users and objects
  Predict the scores by a predefined function

- **Regression**
Outline

- Introduction
- Singular Value Decomposition
- Post-Processing
- Experiments
- Conclusions
Formulation

- User and object features
  \( V \in \mathbb{R}^{n \times m} \) is the training matrix
  find feature matrices \( U \in \mathbb{R}^{f \times n} \) and \( M \in \mathbb{R}^{f \times m} \)
  \( f \) is the dimension of SVD

- Predict scores by a function \( p(U_i, M_j) \)

- Objective function

\[
E = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} I_{ij} (V_{ij} - p(U_i, M_j))^2 + k_u \sum_{i=1}^{n} \| U_i \|^2 + k_m \sum_{j=1}^{m} \| M_j \|^2
\]
**Prediction Function**

- Dot product: \( p(U_i, M_j) = U_i^T M_j \)
- Matrix factorization problem: \( U^T M \approx V \)
- The scores often have a range \([a, b]\).
  \[
p(U_i, M_j) = a + U_i^T M_j
\]

- Linear model
  \[
  - \frac{\partial E}{\partial U_i} = \sum_{j=1}^{m} l_{ij} ((V_{ij} - p(U_i, M_j)) M_j) - k_u U_i
  \]
  \[
  - \frac{\partial E}{\partial M_j} = \sum_{i=1}^{n} l_{ij} ((V_{ij} - p(U_i, M_j)) U_i) - k_m M_j
  \]
Learning Types

- **Batch learning** optimizes all variables at a time.
  Look through all training scores

- **Incomplete incremental learning**
  Consider one user (or one object) at a time

- **Complete incremental learning**
  Consider one score at a time

Score $V_{ij} \rightarrow$ feature vectors $U_i, M_j$

$$E_{ij} = \frac{1}{2} (V_{ij} - p(U_i, M_j))^2 + \frac{k_u}{2} \|U_i\|^2 + \frac{k_m}{2} \|M_j\|^2$$
Variants

- Add per-user biases $\alpha$ and per-object biases $\beta$
  \[
p(U_i, M_j, \alpha_i, \beta_j) = a + U_i^T M_j + \alpha_i + \beta_j
  \]
  Updated by also gradient descent

- Add Constraints on feature vectors
  **Constrained SVD** [Salakhutdinov and Mnih, 2007]
  Import a constraint matrix $W \in \mathbb{R}^{f \times m}$

  \[
  U_i = Y_i + \frac{\sum_{k=1}^{m} I_{ik} W_k}{\sum_{k=1}^{m} I_{ik}}
  \]

  Inappropriate for complete incremental learning
Compound SVD

- Combine both biases and constraints
- Optimize three matrices \( Y, M, W \) and biases \( \alpha, \beta \)
- Update \( Y, M, \alpha, \beta \) by complete incremental learning
- Update \( W \) by incomplete incremental learning

\[
- \nabla Y_i = (V_{ij} - p(U_i, M_j, \alpha_i, \beta_j)) M_j - k_Y Y_i
\]

\[
- \nabla W_k = I_{ik} \sum_{j=1}^{m} l_{ij} \left( \frac{(V_{ij} - p(U_i, M_j, \alpha_i, \beta_j)) M_j}{\sum_{k=1}^{m} I_{ik}} \right) - I_{ik} k_W W_k
\]
Outline

- Introduction
- Singular Value Decomposition
- Post-Processing
- Experiments
- Conclusions
Post-Processing

- An (SVD) algorithm predicts scores with errors
  - Estimate the test errors by training errors
- \( R \in \mathbb{R}^{n \times m} \) is the residual matrix of training data
  \[
  R_{ij} = V_{ij} - p(U_i, M_j, \alpha_i, \beta_j) \text{ if } V_{ij} \text{ exists}
  \]
- Update the biases by residuals
  - \( \bar{R}_j \) is the average residual of object \( j \)

  \[
  \begin{align*}
  \beta_j &\leftarrow \beta_j - \bar{R}_j \\
  R_{ij} &\leftarrow R_{ij} - \bar{R}_j \text{ if } l_{ij} = 1
  \end{align*}
  \]
Regression

- Build a model for each user
- Ridge regression

Features $X \in \mathbb{R}^{t \times f}$, target values $Y \in \mathbb{R}^{t \times 1}$

Involve a kernel function $K(x_i, x_j)$

$$\hat{y} = K(x, X)(K(X, X) + \lambda I_t)^{-1}Y$$

$$K(x_i, x_j) = \begin{cases} (x_i x_j^T)^p & \text{if } x_i x_j^T \geq 0 \\ 0 & \text{if } x_i x_j^T < 0 \end{cases}$$

$p = 5 - 20$ works well in experiments

Only trust a neighbor with high similarity
Acceleration

- The computation of \((K(X, X) + \lambda l_t)^{-1}\) is expensive.
- Kernel function and inversion on a \(t \times t\) matrix.
- Use a threshold on \(t\).
- Acceleration under the polynomial kernel.
- Replace \(K(X, X)\) with \(l_t\).

\[
(K(X, X) + \lambda l_t)^{-1} = \left(\frac{1}{1 + \lambda}\right) l_t
\]

Still use the kernel function in prediction.
The simplified algorithm is like a weighted sum

\[ \hat{y} = K(x, X)(\frac{1}{1 + \lambda})l_t Y \]

\[ = \sum_{a=1}^{t} \frac{K(x, x_a)}{1 + \lambda} y_a \]

Weight average is more reliable

Modify the form again

\[ \hat{y} = \frac{\sum_{a=1}^{t} K(x, x_a)y_a}{\sum_{a=1}^{t} K(x, x_a) + \lambda} \]
Outline

- Introduction
- Singular Value Decomposition
- Post-Processing
- Experiments
- Conclusions
Data Sets

- **Movielens**
  6,040 users and 3,706 movies
  1,000,209 scores (density = 4.61%)
  Select 3 scores of each user as test data

- **Netflix**
  480,189 users and 17,770 movies
  100,480,507 scores (density = 1.18%)
  Use probe set (1,408,395 scores) as test data
Algorithms in Experiments

- Algorithms used for comparison
  - **AVGB**: a simple baseline predictor \( P_{ij} = \mu_j + b_i \)
  - **SVDNR**: SVD without regularization terms
  - **SVD**: SVD with complete incremental learning
  - **SVDUSER**: SVD with incomplete incremental learning in the order of users
  - **CSVD**: The compound SVD algorithm

<table>
<thead>
<tr>
<th>Dataset</th>
<th>AVGB</th>
<th>SVDNR</th>
<th>SVD</th>
<th>CSVD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movielens</td>
<td>0.9313</td>
<td>0.8796</td>
<td>0.8713</td>
<td>0.8706</td>
</tr>
<tr>
<td>Netflix</td>
<td>0.9879</td>
<td>0.9280</td>
<td>0.9229</td>
<td>0.9178</td>
</tr>
</tbody>
</table>
Performance versus Time
Post-Processing Algorithms

- Start from the best algorithm **CSVD**
  - **CSVD**: Compound SVD without post-processing
  - **SHIFT**: Update biases by training residuals
  - **KNN**: $K$-nearest neighbor on residuals
  - **KRR**: Kernel ridge regression on residuals
  - **WAVG**: Weighted average on residuals

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CSVD</th>
<th>SHIFT</th>
<th>KNN</th>
<th>KRR</th>
<th>WAVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movielens</td>
<td>0.8706</td>
<td>0.8677</td>
<td>0.8644</td>
<td>0.8635</td>
<td>0.8621</td>
</tr>
<tr>
<td>Netflix</td>
<td>0.9178</td>
<td>0.9175</td>
<td>0.9139</td>
<td>0.9101</td>
<td>0.9097</td>
</tr>
</tbody>
</table>
Experiments

Competition for Netflix Prize

- Use compound SVD with dimension $f = 256$
  - Use the probe set for validation
- Apply the weighted average algorithm
- RMSE = 0.8868, 35th place when submitted
  - 6.79% better than baseline RMSE 0.9514
    (10% for the Grand Prize)
- Ordinary SVD gives results in 0.91 – 0.93
Outline

- Introduction
- Singular Value Decomposition
- Post-Processing
- Experiments
- Conclusions
Conclusions

- Complete incremental learning for SVD
  Update after looking a single score
- **Compound SVD** outperforms the original SVD
  Combination of biases and constraints
- Post-processing algorithms on the residuals
  KNN, Regression, Weighted average