Abstract

Collaborative Filtering is effective to provide customers with personalized recommendations by analyzing the purchase patterns. Matrix factorization, e.g. Singular Value Decomposition, is another successful technique in recommendation system.

We implemented Singular Value Decomposition algorithm to achieve the least total squared errors. Based on the result, item-feature Collaborative Filtering was further designed and implemented in P-Tree data structure. The purpose of item-feature Collaborative Filtering is to achieve the local optimization. Our experiment on Netflix Prize data suggests Singular Value Decomposition is good at global optimization and item-feature Collaborative Filtering is good at local optimization. Blending of two algorithms achieves better RMSE score.

1 INTRODUCTION

With the increasing global competition in the marketing field, the cost of acquiring a new customer is between 3 and 15 times (depending on the type of business) the cost of retaining an existing customer. Businesses are moving from customer acquisition to customer retention. By analyzing the buyer’s purchase history, recommendation system identifies customer’s behavior, preference, needs, purchase patterns, etc. These information are helpful for seller to set prices, tailor promotions, recommend the most likely purchased items.

Item-based Collaborative Filtering algorithm, one of the most successful techniques in recommendation system, assumes that a buyer is likely to purchase similar items. The key of item-based Collaborative Filtering algorithm is the definition of item similarity.

Another successful technique in recommendation system is Matrix factorization, including Non-Negative Matrix Factorization and Singular Value Decomposition (SVD), which have numerous variants.

This paper is an extension to our previous experimental work on item-based P-Tree Collaborative Filtering algorithm [14] [15]. The rest of the paper is organized as follows. The next section describes P-Tree algorithm, which is for efficient data processing. Section 3 provides an introduction to SVD algorithms. Section 4 describe SVD item-feature Collaborative Filtering algorithms. In section 5 we present the experimental results of SVD item-feature filtering on Netflix Prize data set. The final section gives conclusions and directions for future research work.

2 P-TREE ALGORITHM

Tremendous volumes of data causes the cardinality problem for conventional item-based Collaborative Filtering algorithm. For fast and efficient data processing, we transform the data into P-Tree [12], the lossless, compressed, and data-mining-ready vertical data structures.

P-trees are used for fast computation of counts and for masking specific phenomena. This vertical data representation consists of set structures representing the data column-by-column rather than row-by-row (horizontal relational data). Predicate-trees are one choice of vertical data representation, which can be used for data mining instead of the more common sets of relational records. This data structure has been successfully applied in data mining applications ranging from Classification and Clustering with K-Nearest Neighbor, to Classification with Decision Tree Induction, to Association Rule Mining [16][17][18][19][20]. A basic P-tree represents one attribute bit that is reorganized into a tree structure by recursive sub-division, while recording the predicate true value for each division. Each level of the tree contains truth-bits that
represent sub-trees and can then be used for phenomena masking and fast computation of counts. This construction is continued recursively down each tree path until downward closure is reached. For example, if the predicate is "purely 1 bits", downward closure is reached when purity is reached (either purely 1 bits or purely 0 bits). In this case, a tree branch is terminated when a sub-division is reached that is entirely pure (which may or may not be at the leaf level). These basic P-trees and their complements are combined using Boolean algebra operators such as AND(&) OR(|) and NOT(¬) to produce mask P-trees for individual values, individual tuples, value intervals, tuple rectangles, or any other attribute pattern. The root count of any P-tree will indicate the occurrence count of that pattern. The P-tree data structure provides a structure for counting patterns in an efficient, highly scalable manner.

3 SINGULAR VALUE DECOMPOSITION

SVD is an important factorization of a rectangular real or complex matrix. It is widely used in signal processing and statistics. SVD algorithm was first proposed for collaborative filtering by Simon Funk in Netflix [13]. The method assumes that the user rating on item depends on a predefined function. In SVD the problem is defined as,

\[ \hat{R} = U^T \cdot I \]

where \( U \) is the user feature matrix and \( I \) is the item feature matrix. \( \hat{R} \) is the prediction matrix of \( R \), which is the rating matrix.

According to the definition, the prediction for user \( u \) on item \( i \) is then made by,

\[ \hat{r}_{u,i} = U_u^T \cdot I_i = \sum_f U_u(f) \cdot I_i(f) = \sum_f u_f \cdot i_f \]

where \( U_u \) is the \( u \)-th row in user matrix \( U \), which has a number of \( f \) features of user \( u \). Similarly, \( I_i \) is the \( f \)-dimensional item feature vector of item \( i \).

The parameters in the feature matrices \( U \) and \( I \) are optimized by minimizing the total squared errors of \( \|R - \hat{R}\| \) on the observed set \( T \). However the algorithm may suffer from overfitting. Early stopping is often required to prevent overfitting.

\[ \epsilon_{u,i} = r_{u,i} - \hat{r}_{u,i} \]

where \( \epsilon_{u,i} \) is the residual error from the current prediction.

\[ u_f += lrate(\epsilon_{u,i} \cdot i_f - \lambda \cdot u_f) \]

\[ i_f += lrate(\epsilon_{u,i} \cdot u_f - \lambda \cdot i_f) \]

where \( lrate \) is the learning rate, \( \lambda \) is the parameter to prevent overfitting.

Simon proposed a regularized SVD with gradient descent learning schema. His model trained factors one by one. Other efficient variants were published by Bell and Koren [], where they used alternation least squares for weight updates. Biased SVD was proposed by Paterek [].

SVD algorithm is illustrated in Algorithm 1.

```
Input: UserSet U, ItemSet I, observed rating set T, factor number f, iteration times Iter
Output: SVD model parameters

foreach (u, i) \( \in T \) do
    \( \epsilon_{u,i} = r_{u,i} - \hat{r}_{u,i}; \)
    for \( k = 0; k < f; + + k \) do
        \( u_f = u_f + lrate(\epsilon_{u,i} \cdot i_f - \lambda \cdot u_f); \)
        \( i_f = i_f + lrate(\epsilon_{u,i} \cdot u_f - \lambda \cdot i_f); \)
    end
    \( \lambda = 0.9 * \lambda; \)
end
```

Algorithm 1: Singular Value Decomposition

4 SVD ITEM-FEATURE COLLABORATIVE FILTERING ALGORITHM

Item-based P-Tree Collaborative Filtering algorithm is illustrated in Algorithm 2. The raw horizontal data is transformed to vertical P-Tree structure at the first time. P-Tree is then saved in binary file on the disk and can be loaded later. To predict how user \( u \) rates on item \( i \), we build the item-based similarity matrix and identify the top \( K \) most similar items for item \( i \). The prediction is then made based on \( u \)'s ratings on these neighbor items.

4.1 SVD Item-feature Similarity

The similarity between item pair can be measured by Cosine, Pearson, adjusted Cosine, etc. To fully utilize the results of SVD, we define SVD item-feature similarity of item \( i \) and \( j \) as the similarity of corresponding item vector \( I_i \) and \( I_j \),
4.3.1 Weighted Average

The rating of user \( u \) on item \( i \) is predicted as the average of weighted ratings of items in \( I_K \) by user \( u \). The weight function is the item similarity between item \( i \) and neighbor item \( j \).

\[
r_{u,i} = \frac{\sum_{j \in I_K} r_{u,j} \times \text{sim}(i,j)}{\sum_{j \in I_K} \text{sim}(i,j)}
\]

4.3.2 Item Effects

In weighted average prediction, the item variance is not considered. However, the item variance exists and has a significant impact. A popular item might make the user give a high rating because of conformity, the psychological phenomenon in which an individual’s behavior is influenced by other people. In order to remove the item effect in the prediction, we replace \( r_{u,j} \) in weighted average prediction with \( r_{u,j} - \bar{r}_j + \bar{r}_i \),

\[
r_{u,i} = \frac{\sum_{j \in I_K} (r_{u,j} - \bar{r}_j + \bar{r}_i) \times \text{sim}(i,j)}{\sum_{j \in I_K} \text{sim}(i,j)}
\]

4.3.3 Linear Regression

The raw prediction results could be further regressed by the linear model,

\[
r_{u,i} = \alpha r_{u,i} + \beta + \epsilon
\]

The regression model parameter \( \alpha \) and \( \beta \) are determined by solving the following least squares problem,

\[
\min \sum_{(u,i) \in Q} (r_{u,i} - \hat{r}_{u,i})^2
\]

5 EXPERIMENT

5.1 Data Set and Quality Evaluation

The training data set of Netflix Prize [11] consists of 7 years of 100,480,507 ratings which were rated by 480,189 randomly-chosen, anonymous Netflix customers on 17,770 movies. A total of 2,817,131 ratings are provided as the test data set, half as quiz and the other half as test.

We evaluate the Root Mean Squared Error (RMSE) score on the Netflix test data set.
Table 1: RMSE on SVD

<table>
<thead>
<tr>
<th>Factor number</th>
<th>Iterations</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>100</td>
<td>0.9291</td>
</tr>
<tr>
<td>40</td>
<td>200</td>
<td>0.9181</td>
</tr>
<tr>
<td>80</td>
<td>100</td>
<td>0.9232</td>
</tr>
<tr>
<td>80</td>
<td>200</td>
<td>0.9118</td>
</tr>
</tbody>
</table>

Table 1: RMSE on SVD

Table 2: RMSE on Neighbor Size

<table>
<thead>
<tr>
<th>K</th>
<th>SVD IF</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.9865</td>
</tr>
<tr>
<td>20</td>
<td>0.9900</td>
</tr>
<tr>
<td>30</td>
<td>0.9972</td>
</tr>
<tr>
<td>40</td>
<td>1.0031</td>
</tr>
<tr>
<td>50</td>
<td>1.0078</td>
</tr>
</tbody>
</table>

Table 2: RMSE on Neighbor Size

5.2 Experimental Results

5.2.1 Experiment on SVD Algorithm

Limited experiments with different factor numbers and iterations were tested. The detailed RMSE scores of SVD algorithm is shown in table 1. We used the item feature matrix of SVD with features=40 and iterations=200 to build the item-feature similarity matrix.

5.2.2 Experiment on Neighborhood Size

The size of the neighborhood of Collaborative Filtering algorithm has a significant impact on the prediction quality. The detailed RMSE scores of SVD item-feature Collaborative Filtering algorithms with 10, 20, 30, 40 and 50 neighbors on Netflix test data are shown in table 2.

From the experimental results, it can be observed that the optimal size of neighborhood is 10. As the neighborhood size increases and more non-relevant items are included, the RMSE score drops and the prediction quality deteriorates.

5.2.3 Experiment on Similarity Correction

We applied the similarity corrections described in Section 4.2 to SVD item-feature Collaborative Filtering algorithm. It is observed that SVD item-feature similarity algorithm gets better RMSE score when similarity corrections are included. Detailed RMSE score is shown in Table 3.

5.2.4 Experiment on Item Effects

We tried the experiments on RMSE score of SVD item-feature similarity with item effects. The result shows conformity does exist and item effects have a significant impact on the prediction accuracy. Table 4 shows the detailed RMSE improvement. There is about 5% for SVD item-feature similarity.

Table 3 shows the best RMSE scores of SVD item-feature similarity algorithms with similarity correction and item effects. SVD item-feature similarity Collaborative Filtering algorithm achieves the best results by including similarity corrections and item effects.

5.2.5 Experiment on Regression

The raw prediction results from SVD Item-feature similarity functions are regressed by the linear model and Table 6 shows the results and regression parameter $\alpha$ and $\beta$.

5.2.6 Combinatorial Optimization Algorithms

A key observation with combinatorial method is that usually it minimizes the RMSE score from the individual predictor. Combination on SVD algorithm and SVD item-feature Collaborative Filtering algorithm was further implemented and the final RMSE

Table 5: Similarity Correction & Item Effects

<table>
<thead>
<tr>
<th>Correction</th>
<th>SVD IF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item Effects Included</td>
<td>0.9251</td>
</tr>
</tbody>
</table>

Table 5: Similarity Correction & Item Effects
<table>
<thead>
<tr>
<th></th>
<th>SVD</th>
<th>SVD IF</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>0.9125</td>
<td>0.9608</td>
</tr>
<tr>
<td>β</td>
<td>0.3311</td>
<td>0.0664</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.9171</td>
<td>0.9212</td>
</tr>
</tbody>
</table>

Table 6: Regression

<table>
<thead>
<tr>
<th></th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD</td>
<td>0.5226</td>
</tr>
<tr>
<td>SVD IF</td>
<td>0.4509</td>
</tr>
</tbody>
</table>

Table 7: Blending Weights

score reaches 0.9090, which is better than any individual algorithm. Table 7 shows the weights for each algorithm.

The experiment suggests blending two algorithms improves the prediction accuracy even though SVD item-feature similarity function comes from the result of SVD. Singular Value Decomposition is good at global optimization and Collaborative Filtering algorithm is good at local optimization. The combinatorial method of SVD algorithm and item-feature Collaborative Filtering algorithm has better RMSE score over any individual predictor.

6 CONCLUSION

In this paper we show Singular Value Decomposition and SVD item-feature Collaborative Filtering algorithm on Netflix Prize data. The experiments were implemented in P-Tree. The experiment results show support based similarity corrections and item effects significantly improve the prediction accuracy of SVD item-feature algorithm. The combinatorial method of two algorithms out-performs the result of any individual predictor.

7 ACKNOWLEDGMENTS

We would like to thank all DataSURG members for their hard work on P-Tree API. We also thank the Center of High Performance Computing at North Dakota State University, which provides the computing facility in our experimentation.

References


[18] A. Perera, T. Abidin, G. Hamer and W. Perrizo, Vertical Set Square Distance Based Clustering without Prior Knowledge of K, 14th International Conference on Intelligent and Adaptive Systems and Software Engineering (IASSE 05), Toronto, Canada, 2004
