Abstract

Recommendation systems provide customers with personalized recommendations by analyzing the purchase history. Item-based Collaborative Filtering (CF) algorithms recommend items which are similar to what the customers purchased before. Item-based algorithms are widely employed over user-based algorithms due to less computational complexity and better accuracy.

We implement several types of item-based CF algorithms with P-Tree data structure. Similarity corrections and item effects are further included in the algorithms to exclude support and item variance. Our experiment on Netflix Prize data suggests support based similarity corrections and item effects have a significant impact on the prediction accuracy. Pearson and SVD item-feature similarity algorithms with support based similarity correction and item effects achieve better RMSE scores.

1 INTRODUCTION

With the increasing global competition in the marketing field, e-commerce is moving from customer acquisition to customer retention. Recommendation systems provide personalized recommendations and hence increases customer loyalty, which eventually leads to business success. Through analyzing the customer’s purchase history, the Collaborative Filtering algorithm identifies customer’s preference and recommends the most likely purchased items.

Either user-based or item-based Collaborative Filtering (CF) algorithms have been successfully deployed in e-commerce [1]. The underlying assumption of the user-based Collaborative Filtering algorithms, or social filtering methods, is that if two buyers purchased the same items in the past, they should have the same preferences, and one buyer is likely to buy the products that have already been purchased by the other. However, the computation complexity of the user-based CF algorithm is linear to the numbers of users in the data set. The limitation of computational resources and overwhelming transactional data exceed the capability of current user-based similarity algorithms. Item-based filtering algorithm assumes that a buyer is likely to buy similar items. This assumption is quite true, especially in movie renting or purchasing context, since most customers have a stable preference on the movie genre, actor, director, etc. Compared to the user-based algorithm, the item-base CF algorithm has less scalability concerns and a better prediction accuracy [2][3][4].

This paper is an extension to our previous experimental work on item-based P-Tree Collaborative Filtering algorithm [13]. The rest of the paper is organized as follows. The next section describes P-Tree algorithm, which is for fast and efficient data processing. Section 3 provides an introduction to item-based Collaborative Filtering algorithm. In section 4 we present the experimental results of item-based 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 cause 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 [14][15][16][17][18]. 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 used in association rule mining 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 ITEM-BASED COLLABORATIVE FILTERING ALGORITHM

Item-based P-Tree Collaborative Filtering algorithm is illustrated in Algorithm 1. The raw horizontal data is transformed to vertical P-Tree structure for 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.

3.1 Item-based Similarity

In this section, we present several similarity functions to compute the similarity between items. These similarity functions between item $i$ and $j$ are normalized by the co-support users who rate both item $i$ and $j$, instead of users who rate either item $i$ or $j$. The purpose is to alleviate the sparsity in the data set where most users purchased or rated less than 1% of the items in the item set. In the following subsections, $U$ denotes the co-support between movies.

**Algorithm 1: Item-based P-Tree CF Algorithm**

3.1.1 Cosine Similarity

In Cosine similarity, items are treated as vectors and the similarity between items are computed by the Cosine of the angle between corresponding vectors. Cosine similarity of item $i$ and $j$ is given as,

$$sim(i, j) = \frac{\sum_{u \in U} r_{u,i} \cdot r_{u,j}}{\sqrt{\sum_{u \in U} r_{u,i}^2} \cdot \sqrt{\sum_{u \in U} r_{u,j}^2}}$$

3.1.2 Pearson Correlation

As the most popular similarity measurement, Pearson correlation of item $i$ and $j$ is given as,

$$sim(i, j) = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i) \cdot (r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2} \cdot \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}}$$

where $\bar{r}_i$ and $\bar{r}_j$ is the average rating of the $i$-th and $j$-th item respectively.

3.1.3 Adjusted Cosine Similarity

In Cosine and Pearson similarity the user variance is not considered, but it might have a significant impact. Because some users are easy to give high ratings
while other critical customers are reluctant to do so even though they like the item. To eliminate the user rating variance, Adjusted Cosine similarity of item \( i \) and \( j \) is given as,

\[
sim(i, j) = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u) * (r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_u)^2}}
\]

where \( \bar{r}_u \) denotes the average rating of user \( u \) for all the items he rated.

### 3.1.4 SVD Item-Feature Similarity

Regularized Singular Feature Value Decomposition (SVD) algorithm was proposed by Simon Funk. The prediction for user \( u \) on item \( i \) is made by,

\[
r_{u,i} = U_u^T \bullet I_i
\]

where \( U \) and \( I \) are \( M \)-dimensional vectors of users and items. We define SVD item-feature similarity of item \( i \) and \( j \) as the similarity of corresponding item vector \( V_i \) and \( V_j \),

\[
sim(i, j) = \frac{\sum_{m=1}^{M} (V_{i,m} - \bar{V}_i) * (V_{j,m} - \bar{V}_j)}{\sqrt{\sum_{m=1}^{M} (V_{i,m} - \bar{V}_i)^2} \sqrt{\sum_{m=1}^{M} (V_{j,m} - \bar{V}_j)^2}}
\]

### 3.2 Similarity Correction

It is obvious that two items are not thought similar when only a few customers purchased or rated both. We suggest including the co-support of items in computation of item similarity. We apply the support correction to Cosine, Pearson, Adjusted Cosine and SVD item-feature similarity functions. The new similarity function with support correction is defined as,

\[
\log(N_{ij}) \ast \sim(i, j)
\]

where \( N_{ij} \) is the number of users who rate both item \( i \) and \( j \).

### 3.3 Prediction

Once the item similarity is calculated, we predict the rating of user \( u \) on item \( i \) from his previous ratings on the top \( K \) neighbor items to \( i \), which is denoted as \( I_K \).

#### 3.3.1 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 \( j \).

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

#### 3.3.2 Item Effects

In weighted average prediction, the item variance is not considered. However, similar to the user variance described in section 3.1.3, the item variance exists and has 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 \( r_{u,j} - \bar{r}_j + \bar{r}_i \),

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

#### 3.3.3 Linear Regression

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

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

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

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

### 4 EXPERIMENT

#### 4.1 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 Square Deviation (RMSE) score on the Netflix test data set.
4.2 Experimental Results

4.2.1 Experiment on Neighborhood Size

From our previous work, we knew the size of the neighborhood of Collaborative Filtering algorithm has a significant impact on the prediction quality. Figure 1 shows the RMSE scores of Cosine, Pearson, Adjusted Cosine and SVD item-feature item similarity algorithms with 10, 20, 30, 40 and 50 neighbors on Netflix data.

From the experimental results, it can be observed that the optimal size of neighborhood ranges from 20 to 30 for all the similarity methods. The accuracy of the prediction improves when the neighborhood size increases from 10 to 30 since more similar movies are included. As the neighborhood size increases and more non-relevant movies are included, the RMSE score drops and the prediction quality deteriorates. The detailed RMSE scores on Netflix test data are shown in table 1.

\[
\begin{array}{|c|c|c|c|c|}
\hline
K & \text{Cosine} & \text{Pearson} & \text{Adj.Cos} & \text{SVD IF} \\
\hline
10 & 1.0742 & 1.0092 & 0.9786 & 0.9865 \\
20 & 1.0629 & 1.0006 & 0.9685 & 0.9900 \\
30 & 1.0602 & 1.0019 & 0.9666 & 0.9972 \\
40 & 1.0592 & 1.0043 & 0.9660 & 1.0031 \\
50 & 1.0589 & 1.0064 & 0.9658 & 1.0078 \\
\hline
\end{array}
\]

Table 1: RMSE on Neighbor Size

4.2.2 Experiment on Similarity Algorithms

Adjusted Cosine always achieves better results over Cosine, Pearson and SVD item-feature similarity.

The best RMSE score of Cosine, Pearson, Adjusted Cosine and SVD item-feature similarity algorithms are shown in Figure 2. The Adjusted Cosine similarity algorithm discards the user variance and hence achieves better prediction accuracy.

4.2.3 Experiment on Similarity Correction

We apply the similarity corrections described in Section 3.2 to Cosine, Pearson, Adjusted Cosine and SVD item-feature similarity algorithms. It is observed that Cosine, Pearson and SVD item-feature similarity algorithms get better RMSE scores when similarity corrections are included. However Adjusted Cosine similarity does not. Detailed RMSE scores are shown in Table 2. Pearson similarity receives nearly 2.8% enhancements.

4.2.4 Experiment on Item Effects

We try the experiments on RMSE scores of Cosine, Pearson, Adjusted Cosine and 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 3 shows the detailed RMSE improvements on Cosine, Pearson, Adjusted Cosine and SVD item-feature similarity functions with item effects. There is about 10% enhancement for Cosine similarity, 5.5% for Pearson and 5% for SVD item-feature similarity.

Table 4 shows the best RMSE scores of Cosine, Pearson, Adjusted Cosine and SVD item-feature similarity algorithms with similarity correction and item effects. Cosine, Pearson and SVD item-feature similarity CF algorithms achieve the best results by including similarity corrections and item effects. We only include the item effects for Adjusted Cosine similarity function since it does not benefit from support based similarity correction.
## 5 CONCLUSION

In this paper we show the extension study of item-based similarity Collaborative Filtering algorithm on Netflix Prize data. The experiments implement Cosine, Pearson, Adjusted Cosine and SVD item-feature algorithms. Each algorithm is implemented in P-Tree with different neighborhood sizes. The experiment suggests optimal neighborhood size ranges from 20 to 30. The results also show support based similarity corrections and item effects significantly improve the prediction accuracy of Cosine, Pearson and SVD item-feature algorithms. When the similarity corrections and item effects are included in the similarity functions, Pearson and SVD item-feature Collaborative Filtering algorithms achieve more accurate predictions.

## 6 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


