Music Recommendation Using Collaborative Filtering

Vuong Khut
Department of Computer Science, Earlham College

Introduction

In the last 25 years, the use of recommender systems has expanded rapidly. These systems, designed to produce recommendations that match users' interests, are incorporated into areas such as news, entertainment, and research. One common technique used by recommender systems is collaborative filtering, which produces recommendations for a user based on choices of users with similar preferences. This work describes a recommender system for music tracks using collaborative filtering.

Collaborative filtering is widely used; however, the method does not scale well with large datasets. It is important to build a recommender system using collaborative filtering that not only is accurate but also runs efficiently. In this project, I attempt to build a recommendation systems for music tracks using collaborative filtering with two optimization methods, dataset rescaling and automatic halting. These methods will be built and tested with different parameters to address the performance issue while retaining an acceptable accuracy.

Background

1. Collaborative Filtering
Collaborative filtering provides recommendations for a target user by estimating the utility of different items based on the habits or ratings of other users. The advantage of this technique is that the behavior of a user can generally be predicted from users who are similar. For example, suppose we have a user-song table as shown below, where 1 denotes a user has listened to a song, and 0 if they have not.

<table>
<thead>
<tr>
<th>User</th>
<th>Song 1</th>
<th>Song 2</th>
<th>Song 3</th>
<th>Song 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

The algorithm in this case may determine that u1, u2, and u4 are similar (since they all listen to s1, s2, s4), and recommend song s3, which has been listened to by both u1 and u4.

2. Types of Ratings
Determining the similarity of different users requires recommender systems to assign a numerical value for each user-item pair which represents the user’s rating of the item. There are two types of user ratings that can be used by recommender systems, explicit rating and implicit rating.

- **Explicit Rating**
  - Users explicitly provides feedback for songs.
  - Ratings are not provided by users, but are produced based on some activity.
  - Scale: 1-5 where 1 means “strongly dislike” and 5 means “strongly like”.
  - Rating = value.
- **Implicit Rating**
  - Users are not asked to provide ratings for every song, so we make use of implicit feedback.
  - Scale: binary, where 1 is “listened to” and 0 is “not listened to”.
  - Rating = 1 if listened to, and 0 otherwise.

Testing and Evaluation

Given a user u, we retain k components of the vector u, where k is a positive integer less than m, and set the remaining m - k components to 0. We can assume that u has the form

\[ \mathbf{u} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_k \\ \mathbf{0}_1 \\ \mathbf{0}_2 \\ \vdots \\ \mathbf{0}_{m-k} \end{pmatrix} \]

Then, we apply the collaborative filtering algorithm to generate recommendations for u. To prevent the model from producing too many recommendations, we set the maximum number of recommendations to be 10, and the songs chosen are the most frequent songs among the list of similar users. The components that were set to be 0 are now compared with the corresponding real data.

The model is assessed based on the runtime as well as the following measures:

- **Precision**: The number of songs which were hidden that are in the list of recommendations out of all recommended songs.
- **Recall**: The number of songs which were hidden that are in the list of recommendations out of all hidden songs.

Let us represent the proportion of similar users that must have listened to a song for that song to be recommended, and let 0 be the largest value the angle between u_i and u can have for them to be considered similar. The following graph shows precision and recall rates, respectively, for different combinations of k and 0.

Methods to Improve Runtime

In this work we implement two methods that are used to reduce the runtime.

A. Randomly select a subset of the dataset. This method selects a fixed-size subset of the dataset, which is generated at random. The subset size parameter is determined from experiments to see whether our model can run within a reasonable amount of time without decreasing the accuracy rate by a significant amount.

B. Halt after finding enough similar users. The execution is stopped once the number of similar users found is sufficient to generate recommendations.

Experiment Results and Conclusion

Denote s as the proportion of the generated subset compared to the full dataset, and m as the number of similar users threshold to halt. We use \( s = 0.3 \) and \( 0.4 \) as these values give the best recall rate in the previous step. The execution time of different values for s and m are shown in Table 3.

<table>
<thead>
<tr>
<th>s</th>
<th>m</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>500</td>
<td>1.84270</td>
</tr>
<tr>
<td>0.4</td>
<td>500</td>
<td>2.09147</td>
</tr>
<tr>
<td>0.3</td>
<td>1800</td>
<td>1.93840</td>
</tr>
<tr>
<td>0.4</td>
<td>1800</td>
<td>2.06406</td>
</tr>
<tr>
<td>0.3</td>
<td>2500</td>
<td>2.52140</td>
</tr>
<tr>
<td>0.4</td>
<td>2500</td>
<td>2.62814</td>
</tr>
<tr>
<td>0.5</td>
<td>2500</td>
<td>3.07830</td>
</tr>
<tr>
<td>0.3</td>
<td>5000</td>
<td>2.30840</td>
</tr>
<tr>
<td>0.4</td>
<td>5000</td>
<td>2.92790</td>
</tr>
</tbody>
</table>

The average runtime before deploying these methods is 5.81 seconds, whereas the runtimes documented in Table 3 range from 183 to 2011 seconds. Lower values of s and m further reduces the runtime. Furthermore, the precision and recall rates attained using these methods are both slightly higher than the corresponding values in tables 1 and 2. The results of these experiments show that the runtime-improvement methods both reduce the overall runtime and retain the accuracy of the model.

From the results of this project, for related future work, we would like to explore the hypothesis that the best accuracy of this model is attained after finding a certain number of similar users. We would also want to explore other methods which manipulate the dataset to reduce the runtime. Such methods would need to consider properties of the dataset so that the collaborative filtering function can make less comparison without losing accuracy.

References


Acknowledgements

This project is a part of the Senior Capstone Experience of the Department of Computer Science at Earlham College. I would like to thank Dr. David Barbella, Dr. Ajit Chavan, Dr. Xunfei Jiang, and Dr. Charles Peck of Earlham College for their support in this project.