MyAnimeList Recommendation

Using ML to Intelligently Recommend Anime

Oscar Li

9/12/19

Problem Statement

Nowadays, we have access to nearly every show ever created at our fingertips, whether it be from a legal source such as Netflix or Crunchyroll, or one of the myriad of illegal sources. With this many options, the paradox of choice makes it hard to commit to watching a single show. While anime recommendation services do exist, most of them either rely on user recommendations or only feature the most popular shows. To resolve this issue, I aim to create a model that is able to automatically recommend anime based on all of the shows that the user previously watched and rated.

Dataset

The dataset was collected using the code from https://github.com/VirtualLights/MyAnimeList-Scraper. This code scrapes the MyAnimeList website in order to collect usernames, then records the anime lists for each user. Since seasonal anime airs weekly, the code ran for a week in order to prevent bias. In addition, the script collects the information every 5 minutes to prevent bias due to users living in different timezones. Users were tracked using their usernames, and the shows that they watched were identified using the anime id number generated by MyAnimeList. For each show, I tracked the status (whether they were planning to watch, watching, dropped, or completed the show) as well as the score that they assigned to it (out of 10).

Data Cleaning

For the best results, I filtered the dataset to only include the records where the user had completed the show and rated it above a 0 (since MyAnimeList treats unrated shows as having a score of 0). However, in the Jupyter Notebook I have left commented out code with alternative data cleaning methods employed. These will give a larger overall dataset, but will have more noise.

Evaluation Metric

To evaluate the effectiveness of the model, I chose to use the AUC (area under the curve) score. While alternative scoring methods exist (RMSE, root mean square error), they are not as effective in this case.[1] The AUC score measures the area under the ROC curve (receiver operating characteristic curve), which plots the true positive rate against the false positive rate.[2] In order to achieve a more accurate score, I used 5-fold cross validation and calculated the average of the AUC scores.

[1] https://github.com/lyst/lightfm/issues/180#issuecomment-302484982
[2] https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc, https://sites.northwestern.edu/msia/2019/04/24/personalized-restaurant-recommender-system-using-hybrid-approach/

Methodology

The dataset is stored as a csv file. I used pandas to parse the csv file into a DataFrame, then filtered the DataFrame according to the guidelines specified in Data Cleaning. Using the ‘user’ and ‘anime-id’ columns, I built a LightFM dataset to map user-item interactions. To perform cross-validation, I used scikit-learn’s KFold function to split the DataFrame into train and test sets. I then fitted the model with the train set, and then calculated the AUC score for both the train and the test sets. For the model, I used the BPR (Bayesian Personalized Ranking pairwise loss) loss function. After all 5 splits were finished, I averaged the AUC score.

Result

```
Average collaborative filtering train AUC: 0.9214513778686524

Average collaborative filtering test AUC: 0.8995880842208862

Runtime of kFold is: 901.299551486969 seconds.
```

Conclusion and Next Steps

For me, this project was a great introduction to recommender systems, while still having a tangible purpose. The high AUC scores were a surprise to me, but there are many areas of improvement and bias reduction that may be more realistic.

In the future, I would change how I gathered data. The only website that I gathered data from is MyAnimeList. While MAL is a popular site, it is predominantly used by a western audience, and therefore is not truly representative of anime watchers as a whole. In addition, LightFM is capable of performing hybrid recommendation, using both user and item features. Possible user features include the friends of each user, the clubs that they are a member of, and the favorites they have selected. Possible item features include the studio/producers, the genres, the popularity, and the (age) rating. After collecting the data, there are additional filtering steps that I could take. A common trend is that a user will group ratings towards extremes (such as rating all anime in the 8-10 range). One solution to this is to subtract the user's mean rating from each of their ratings, and converting it to a binary (-1 for negative, 1 for positive) score. When evaluating the data, LightFM has 4 different loss functions; for this project, I only utilized the BPR loss function. In the future, I would like to try the other loss functions in order to see if they produce better results.

As a next step, it may be possible to upload this project to a web server in order for other people to get recommendations. I have already implemented a function to get recommendations, but since it relies upon MyAnimeList, it is likely not fully stable under high load. A possible workaround for this is to track anime names instead of the internal MAL id's, and for users to import their anime lists through XML.