CSE 272 Recomendation System

Jacob Low 2023

May 26, 2023

1 Summary

This repository is divided into two separate Python notebooks that each implement a different recommendation system built with PyTorch. Both implementations are tested on the Amazon Movies and TV reviews data set. Each notebook follows the same structure: data preprocessing, dataset construction, model definition, model training, and model testing. The Matrix Factorization notebook includes an additional section that generates a recommendation list consisting of 10 items for 175 randomly sampled users in the testing set. The file "recs.txt" contains a recommendation list. I sampled 175 users (from the 300k users in the testing set) due to the lack of efficiency in the brute force ranking algorithm.

2 Data Preprocessing

The file "Movies_and_TV.json" is read into a pandas data frame. All columns are dropped except the user ID column denoted as "reviewerID", product ID column denoted as "asin", and rating column denoted as "overall". All users with less than 5 reviews are removed from the data frame to ensure that each user has a sufficient number of ratings. The sklearn preprocessing LabelEncoder is used to replace each unique user and product string with an integer value to be compatible with the torch dataset class. Train and test data frames are constructed by sampling 80% of each users reviews into the training data and the remaining 20% into the testing data. The train and test data frames are saved because all three notebooks follow the same preprocessing steps.

3 Dataset Construction

The MoviesTVDataset class inherits from the abstract torch Dataset class and is modified to match the data frame schema. This new class converts the data frames into torch Datasets which are compatible with the torch DataLoader. The DataLoader is used to train each model in smaller batch sizes.

4 Model Definition

I implemented two different recommendation systems: Matrix Factorization and Neural Collaborative Filtering. Both models inherit from the abstract torch nn.Module class.

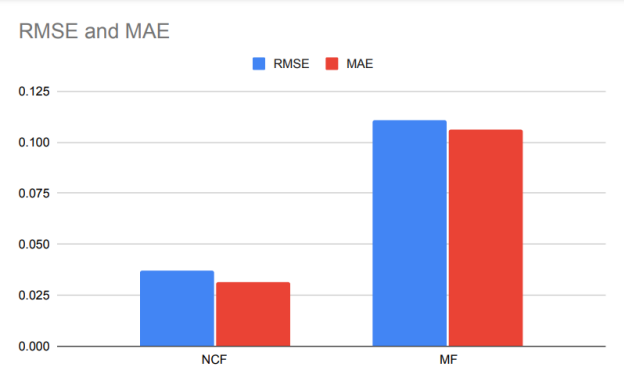

Figure 1: Root Mean Square Error and Mean Absolute Error on test dataset

4.1 Matrix Factorization

The goal of the Matrix Factorization algorithm is to build a matrix of users an products filled with known and predicted rating values. We need to learn two factor matrices (one for users and one for products) that represent the rating between user and product when multiplied together. The following model learns 32 dimensional embedding vectors for each user and product that capture this relationship.

The forward pass of the Matrix Factorization model generates 32 dimensional embeddings for each user (first factor matrix), and 32 dimensional embeddings for each product (second factor matrix). The diagonal of the dot product of the two factor matrices represents the rating of each (user, product) pair in the training batch.

The Mean-Squared Error loss is used with Adam optimizer to train the model. Since the training data set is very large (2.9M ratings), the model is trained for 1 epoch and achieves a Root Mean Square Error of 0.11068 and Mean Absolute Error of 0.10611 on the test data set.

4.2 Neural Collaborative Filtering

The goal of the Neural Collaborative Filtering (NCF) algorithm is to learn a function that can predict the ratings of users for items that they have not yet interacted with. NCF uses neural networks to capture more complex patterns and relationships between the users and products. The following model learns the weights to a single linear layer that captures the relationship between each user and product.

The forward pass of the NCF model generates 32 dimensional embeddings for each user and 32 dimensional embeddings for each product. The embeddings of each (user, product) pair in the training batch are concatenated and passed as input to the linear layer. The linear layer produces a predicted rating for each (user, product) pair in the training batch.

The Mean-Squared Error loss is used with Adam optimizer to train the model. The model is trained for 1 epoch and achieves a Root Mean Square Error of 0.03715 and Mean Absolute Error of 0.03154 on the test data set. This model achieved better accuracy than the Matrix Factorization approach (as shown in Figure 1), but took significantly more time to train and evaluate.

5 Recommendation List Generation

The remaining sections focus on ranking and further evaluation with the Matrix Factorization based recommendation system.

I chose to randomly sample 175 of the 300k users in the test data set for ranking evaluation due to the lack of efficiency in my recommendation list algorithm. Since the data set contains over 140k products, I create a temporary MoviesTVDataset instance for each of the 175 sampled users. The temporary dataset pairs the current user with all 140k products (minus the products the user rated in the training set). The DataLoader is then used to predict the current users rating of each item in batches of 2048 items. The top 10 ratings are saved from each batch. After all of the batches have been predicted, the top 10 overall ratings are added to the "recs.txt" file. Each line of the "recs.txt" file contains the user's reviewerID followed by the top 10 products asin.

6 Model Testing

In addition to MAE and RMSE metrics on the test dataset, I also use Precision and Recall to evaluate the Matrix Factorization implementation. I use a dictionary to store a list of predicted rating and actual rating pairs for each user in the test dataset. The Precision and Recall metrics require two parameters to be defined: k and threshold. The parameter k represents the number of recommended items for each user and threshold represents the minimum rating to be recommended. Since the smallest number of reviews a user can have in the test set is 1 (all users with less than 5 reviews are removed from the original data frame and 80% are sampled for training), I set the parameter k to 1 and the rating threshold to 3.5. The average precision @ 1 across all users is 0.65 and the average recall @ 1 across all users is 0.47.

I also calculated the precision and recall with k = 10 because there are a significant number of users in the testing set with at least 10 prediction/ground truth pairs. The average precision @ 10 across all users is 0.65 and the average recall @ 10 across all users is 0.67.

The test data set contains 311,221 users and 700k ratings. 168,848 of the 311.221 users (over 50%) only have 1 ground truth rating in the test dataset. Since there is a significant portion of the testing set with only 1 ground truth rating, the precision and recall metrics are less meaningful for high values of k.

7 References

Neural Collaborative Filtering
Matrix Factorization
Neural Collaborative Filtering YouTube