Recommendation System

Saaket Agashe

May 2022

1 Data Selection and Preprocessing

For this project, I use the 5-core amazon product reviews data set of Video Game reviews. This contains 231,780 different data points. From the data set I use the following fields for predicting user ratings.

1. overall: Product rating from 1-5
2. reviewerID: Unique ID of a reviewer
3. asin: Unique ID of a product
4. reviewText: Full user review as text
5. summary: Title of user review

The data from the source is in a .json.gz format. I iterate through the dataset file and convert it into a list of dictionaries in python. I then convert this list to a Pandas Dataframe where the keys of the dictionaries are the column headers.

Train Test Split: I use a stratified train-test-split method to create hold out test set. the split is stratified on reviewerID. This results in there being nearly 80% of products reviewed by any given user in the training set and nearly 20% of the products reviewed by the same user in the test set.

2 Rating Prediction Algorithm

I have chosen to use the Matrix Factorization method as the rating prediction algorithm. I use the Pytorch library and create a custom Pytorch Dataset for Matrix Factorization. The dataset extracts the necessary fields from the dataframe, i.e reviewerID, asin, and the overall rating. These reviewerID and asin are converted to integers and the mapping from the integers to the original IDs are stored in python dictionaries. The Dataset returns tensorized versions of the reviewerID, asin (ProductID) and the rating.

	Matrix Factorization	Neural Network (Custom)
RMSE	1.3	1.06
MAE	1.01	0.71
Precision	0.14	0.32
Recall	0.13	0.22
F1 Score	0.11	0.21

Table 1: Evaluations of ratings on test set

The matrix factorization algorithm is implemented as a Pytorch Module. The reviewer and product factor matrices are implemented as Pytorch Embeddings. Embeddings are $(numProducts, embeddingSize)$ and $(numReviewers, embeddingSize)$ dimension matrices. Furthermore, I include reviewer and product biases as $(numProducts, 1)$ and $(numReviewers, 1)$ dimension vectors. The embedding weights are initialized uniformly.

During training and testing, the model takes in a batch of reviewer_ids and product_ids, multiplies the corresponding vectors from the embedding matrices and returns the sum of this product as the overall rating from the reviewer for the product.

2.1 Finding the User and Product Factor Matrices

The training loop for finding user and product matrices uses a Stochastic Gradient Descent optimizer on the Root Mean Squared Loss of the predicted and target ratings. I use a batch size of 512 and a learning rate of 0.01. The Training and Testing losses over time are shown in figure 1. I use the SparseAdam pytorch optimizer, which accelerates the rate of convergence of stochastic gradient descent and can handle sparse embeddings. The Mean Squared Error, and Mean Absolute Error for the test set is show in Table 1.

3 Advanced Topics: Item Recommendation

3.1 Creating the recommendation list

For creating the recommendation list I consider only the test dataset. Here, for each unique user, and each product that is not found in the training set I obtain a rating score prediction. These predictions are sorted in a descending order. The top-10 product recommendations for each user are considered as the recommendation list.

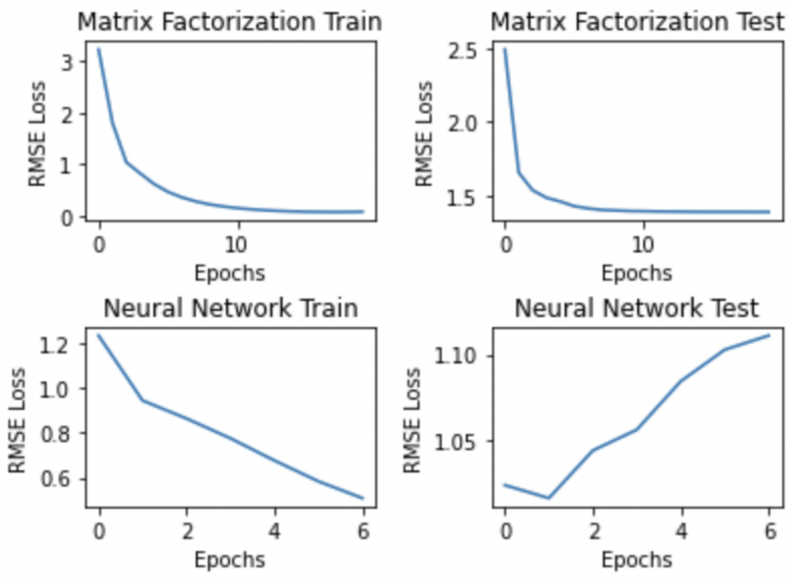

Figure 1: Loss Plots for Matrix Factorization and Neural Network

3.2 New Algorithm (Extra Credit Section)

Neural Networks have shown to be successful at numerous machine learning tasks. Since learning the user and product factors is an ML problem, I use neural networks to learn better factors as well as creating an end to end rating prediction system.

The algorithm includes the two factor matrices used in Matrix Factorization method. For each example reviewer_id and product_id I obtain vectors from the factor matrices. Instead of multiplying these vectors, I concatenate them. These concatenated vectors are then passed through a 2 layers of a Linear Feed Forward Neural Network. I use the ReLU activation function in between the two layers.

Table 1 shows the improved performance on the quality of rating prediction after using this Neural Network based algorithm.

The code for this work can be found at:

https://github.com/saa1605/recommendation_system