Capstone Project
Ibrahim Sheikh

Note

For this project, I chose to work on the Netflix Movie Recommendation project.

Data Preparation & Exploration

The data was provided to us in two files: a text file with the movie number, user id, and rating and another text file with just movies. To begin this project, I decided to build a pandas data frame. I stored the user ids and ratings in a dictionary and matched it to the movie then created a data frame from the dictionary. The initial data frame had 5,000 columns and 472,542 columns. This indicates a very large data frame with more than 2.3 billion entries. However, the majority of the data was missing as most users had not seen most movies.

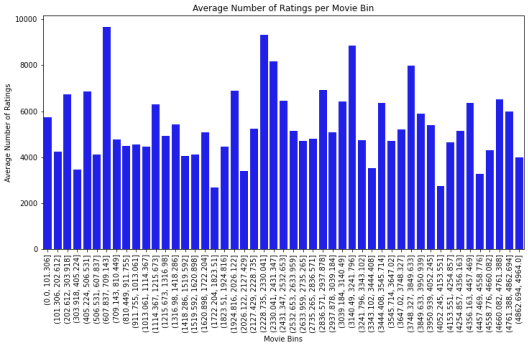

Figure 1

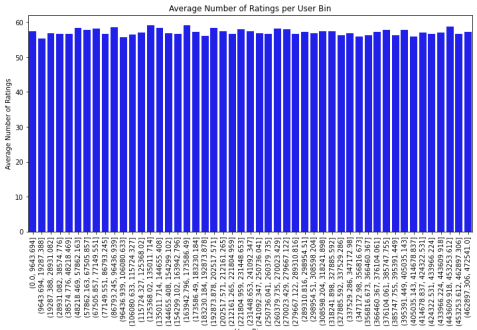

Figure 2

In an initial investigation, I found that there were over 2.3 billion missing entries. To be precise, 98.86% of the data was missing. This led me to explore how many ratings does every movie have. Thus, I binned movies by 100 to see how many non-null ratings were there. Figure 1 shows how the average varies between 300 to 1,000 movies. This tells us how the data is missing and it is not systematically. Figure 2 shows how many movies did users rate on average. As we can see, it is a pretty uniform distribution which means on average, users rated about 55 movies. This figure, like figure 1 is also binned by users so it can fit in a plot.

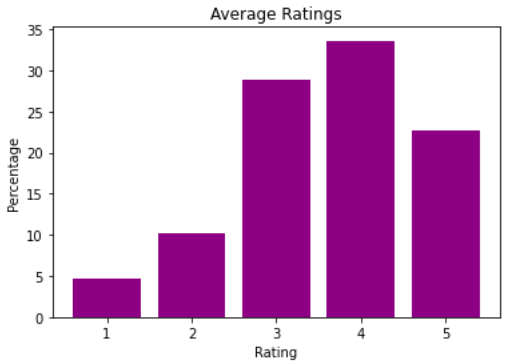

Figure 3

Next, I was interested in how the non-null values distribute over ratings. The ratings ranged from 1-5 and Figure 3 shows how the ratings were distributed. We see that 4 is the most common rating accounting for almost 35% of all ratings, followed by 3, followed by 5. This suggests that on average, users are pretty generous with ratings. This is probably because of self-selection bias. Because users chose the movies they watched, we would expect them to rate it higher. Users could simply not finish/stop watching movies they did not like.

One of the most surprising things about this dataset was that there were no movies that were completely unrated. The movie with the lowest number of ratings was Gone in 60 Seconds where it only had 13 ratings. The most frequently rated movie was The Cookout with 193,941 reviews.

Filtered Data

Next, I decided to use judgement calls and filter the dataset as there was far too much missing data. There were two questions around which I filtered my dataset. The first was "If a user has not rated enough movies, are they a reliable rater?" and the second was "If a movie does not have enough ratings, can it be rated reliably?" Thus, I began to filter along these axes. In my first attempt I used the 25% threshold for movies and users. I dropped movies that were below the 25% threshold (fewer than 191 ratings) and users who had less than 10 ratings. In this attempt, my data frame's shape shrunk to 354,000 rows and 3,750 columns. In this attempt, I still had 98.02% data as missing and the distribution of ratings was very similar to the original dataset. Because there was no marginal gain, I also decided to use the mean to remove movies and users. In this attempt, I removed users who had less than 30 ratings and movies which had received less than 600 ratings. The dataset shrunk to 224,000 rows and 2,400 movies. Because I had now lost half the dataset, I decided to move along with the first attempt. *This was also discussed with Pascal*

The Model

For this project, given the amount of data we had, how much of it was missing, its lack of linearity and direction, I decided to build a fully connected neural network. There was an input layer which took the data as input and outputted a size of 128. Then there was a hidden layer that reduced dimensionality from 128 to 64. Lastly, there was an output layer that took the 64-size layer and returned the predicted rating. There was a dropout layer to prevent overfitting and used a probability of 0.5. The dropout layer does this by setting random values to 0 and is commonly used in training neural networks.

Next, I defined a forward function that determines how the data flows through the model. Here, I used a ReLU activation function. Activation functions help introduce nonlinearity in neural networks and are necessary for training as they help overcome the vanishing gradient problem. In the forward pass, a user's embeddings (row and movie indices) are retrieved and passed through in the fully connected neural network. Next, I started to prepare the data for this model. For this model, I did not use the missing data in training to avoid changing the distribution of the real data, thus, missing data was not imputed.

First, I ensured to use float32 as the data type because PyTorch works well with this type. Next, I prepared a test set where I randomly chose one non-null rating from every movie and changed that rating to NaN in the training set to avoid leakage. Next, I set up the model. I set the embeddings size to 50, changed the device to GPU as I was using Google Colab and the optimizer as the Adam Optimizer with a learning rate of 0.001. I used the mean squared error loss as the criterion for training. With a batch size of 32, I began training this model with 5 epochs.

Analysis & Results

After approximately 12 hours and 33 minutes, the model finished training on the Google Colab GPU.

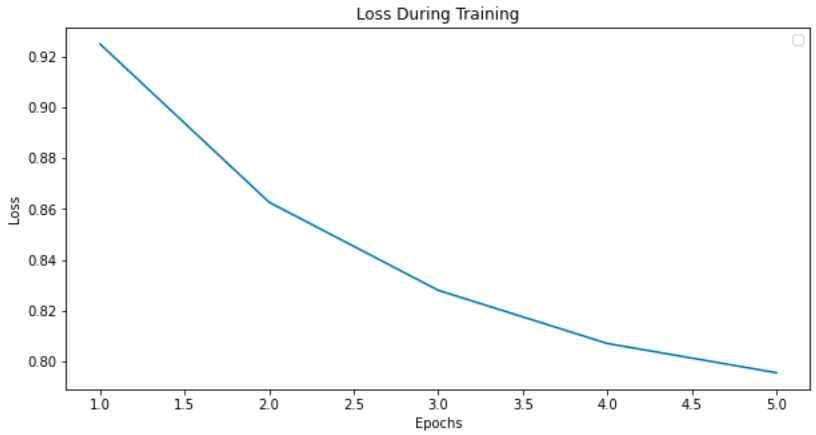

Figure 4

Figure 4 displays the model's training loss over five epochs. As we can see, there is an ideal downward curve which shows the model is learning after every epoch thus there is less loss. The first epoch led to a loss of 0.92 while the last one had a loss of 0.80. This shows the model is learning well over the training data but the learning is slowing as well. The gradient of the curve is decreasing at every epoch thus further epochs may not be as beneficial for the model.

The model achieved a Root Mean Squared Error of 1.155. The model performed above my expectations. I think a neural network is a great model for this task given the complex dataset and expected non linearity between the columns.

RMSE: 1.1555