Rotten Tomatoes Rating Regression Model

Terry Feng

Abstract

A deep neural network regression model was created to predict the Tomatometer and audience ratings of movies. A dataset of over 17,000 movies as preprocessed and used to train this model. Information such as runtime, content rating, actors, authors, directors, release month, and production company was either One-Hot encoded (categorical variables) or normalized (continuous variables). The model was four layers deep with an input layer of 240 nodes, two hidden layers of 64 nodes, and an output layer of 1 node. ReLu was the activation function of choice, except for the output node which used a linear activation function. The resulting model had a 15.22% mean absolute error when predicting audience rating and a 21.46% mean absolute error when predicting Tomatometer rating. The model was able to make low accuracy predictions for movie ratings.

1. Introduction

Rotten Tomatoes is a review aggregator for movies and TV series. It collected reviews from both professional critics and amateur viewers to produce two ratings per film: Tomatometer score and audience score. These scores denote how well a movie is received by watchers, as such, they serve as an indication to decide if the movie is worth watching. Being able to predict this score before a movie is released would help people decide their interest in the movie. In this project, I plan to create a deep learning neural network to perform regression on known attributes of unreleased movies to predict their eventual review score. The results of this model demonstrated that a movie's rating could be predicted at a rate that is better than random with low to medium accuracy.

2. Problem Definition and Algorithm

2.1 Task Definition

The input will be a dataset of over 17,000 movies pulled from the Rotten Tomatoes website. The information that will be fed into the model will be as follows: runtime, content rating, genres, directors, authors, actors, release month, and production company. The output will be either the predicted Tomatometer rating or the predicted audience rating. This is an interesting and important problem because it could save people a lot of time when determining if a movie should be watched or not.

2.2 Algorithm

The work to be done is split into two sections: preprocessing the data and training the model.

For preprocessing the data, we first convert the categorical data we have into One-Hot encoded variables. Most of our data is categorical, so everything besides the runtime and the ratings will be converted. One problem presents itself however, as there are large amounts of unique values for some categories, 200,000 actors for example. As such, we will only encode the most frequent 50 unique values of each category. After processing the categorical variables, we then normalize the continue variables of runtime and the two rating with a min-max scaler formula. After dropping all the rows of data without a y-value to feed into the model, we have finished preprocessing the data.

For the model itself, it is a four-layer neural network. The first layer, the input layer, has 240 nodes and a ReLu activation function. There are then two 64 node hidden layers also with ReLu activation. Finally, the output layer is 1 node with a linear activation function.

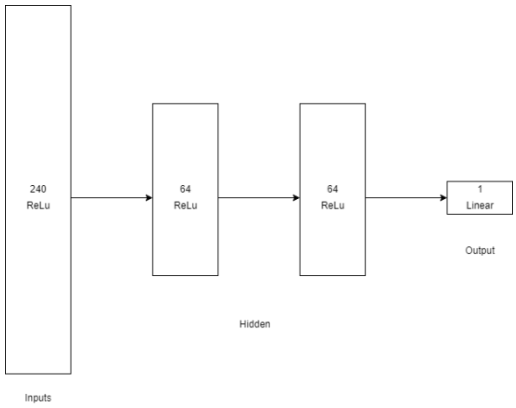


Figure 1. Model architecture

After the model is trained, we will be able to feed in a vector containing a movie's information and get a predicted rating back. As an example, consider a movie titled "X". We will construct the input data by normalizing its runtime and putting 1s in the One-Hot columns if the movie contains the specified category. Afterwards, we ask our model to predict its rating, and the model will output a number from 0%-100%.

3. Experimental Evaluation

3.1 Method

The evaluation of the method was done so after training the model for 10 epochs. I used mean absolute error as the loss function and evaluated the model on the testing dataset which was split from the main dataset. A lower loss function value meant a better model. The training/testing data was generated randomly from the dataset using a 70/30 split after all the processing described in section 2.2. This is realistic data as it is actual movies that had reviews and ratings. For optimizing the model, some independent variables investigated were the amount of training epochs, the number of unique values One-Hot encode, and the number of hidden layers. The dependent variable was the loss function after evaluating on the test set.

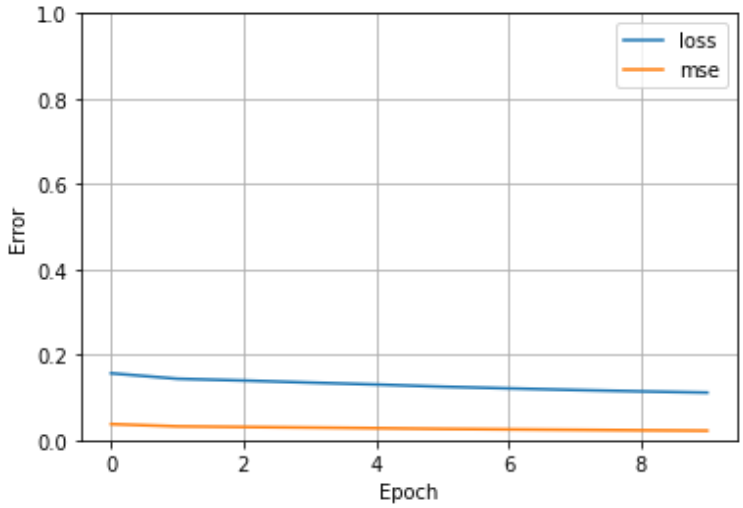
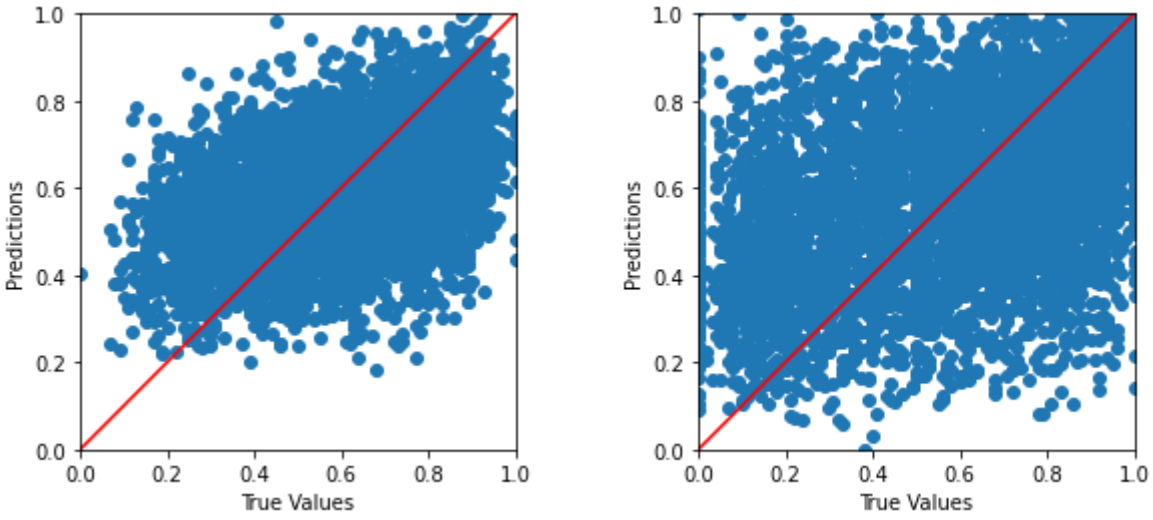


Figure 2. Mean absolute error and mean squared error after training.

3.2 Results

The results were varied depending on whether we were predicting on the audience score or the Tomatometer score. The audience score had a mean absolute error of 15.22% and showed some non-random predictions when compared to the true score. However, the Tomatometer rating had a mean absolute error of 21.46% and showed no significant predictions. This was likely due to a large amount of 0% true ratings that were fed into the model.



Audience Rating

Tomatometer Rating

Figure 3. Predicted values vs. true values for both audience and Tomatometer rating.

3.3 Discussion

As we can see in the audience rating plot in **Fig 3,** there is some predictive accuracy for the model. For high true values, we are more likely to see high predicted values. Similarly, for low true values, we are more likely to see low predicted values. The predictions follow an increasing trend and cluster, albeit the cluster is quite wide, around the 100% accuracy line in red.

For the Tomatometer rating, there is less of a trend and less clustering; however, we can still see that at the extreme ends of low and high true values, the predicted values are less likely to be completely wrong. The lower accuracy is likely due to the large amount of data with 0% rating as their true value, visible on the lefthand side of the plot. It likely confused the model and resulted in lower accuracy. If these 0% ratings were omitted, it is likely that the model would improve.

4. Related Work by Others

I found a related work that built a classifier model to predict which bucket of ratings a movie would fall into based on the reviews left on the site. My work differs in that instead of using reviews to predict the rating, I assume that the quality of the movie is related to the reviews it will receive and thusly the rating it gets. As such, I input movie attributes into the model instead of individual reviews. I also perform a regression on the rating instead of classifying it into four buckets the quartiles.

5. Improvements

I believe that one of the biggest limitations of this project was the amount of categorical data present. Some of these categories also had high cardinality, which created problems with One-Hot encoding due to the number of unique values. As such, I had to only encode the top 50 most frequent unique values, leading to a lot of information that was lost.

One improvement that I could have done is to change my data processing method for high cardinality categories such as authors, directors, and actors. I could have used entity embeddings to find relationships between the lists of people and convert those similarities into number which I could work with. I would treat the entire list of actors of a movie as one string, and use it as an input into an entity embedding model with the movie rating as the predictor variable. After this intermediary model has completed, I would have relationships between the actor lists of each movie. I could also use this technique on directors and authors. This way, I could preserve the information that One-Hot encoding could not.

6. Conclusion

- The model could predict with low accuracy the ratings of movies. Audience: 15.22% mean absolute error. Tomatometer: 21.46% mean absolute error.
- Audience rating was predicted with higher accuracy than Tomatometer rating.
- The model's accuracy improved with training epochs until it plateaued.
- Other processing method of categorical data could have been used to improve model accuracy.

7. References

Brownlee, J. (2020, August 27). *Regression tutorial with the Keras Deep Learning Library in python*. Machine Learning Mastery. Retrieved May 8, 2022, from https://machinelearningmastery.com/regression-tutorial-keras-deep-learning-library-python/

Leone, S. (2020, November 4). *Rotten tomatoes movies and critic Reviews Dataset*. Kaggle. Retrieved May 8, 2022, from https://www.kaggle.com/datasets/stefanoleone992/rotten-tomatoes-movies-and-critic-reviews-dataset?resource=download&select=rotten_tomatoes_movies.csv

Rosebrock, A. (2021, April 17). *Regression with keras*. PyImageSearch. Retrieved May 8, 2022, from https://pyimagesearch.com/2019/01/21/regression-with-keras/

Technical Documentation:

https://keras.io/api/

https://pandas.pydata.org/docs/index.html

https://www.tensorflow.org/tutorials