Name: Nooreldean Koteb

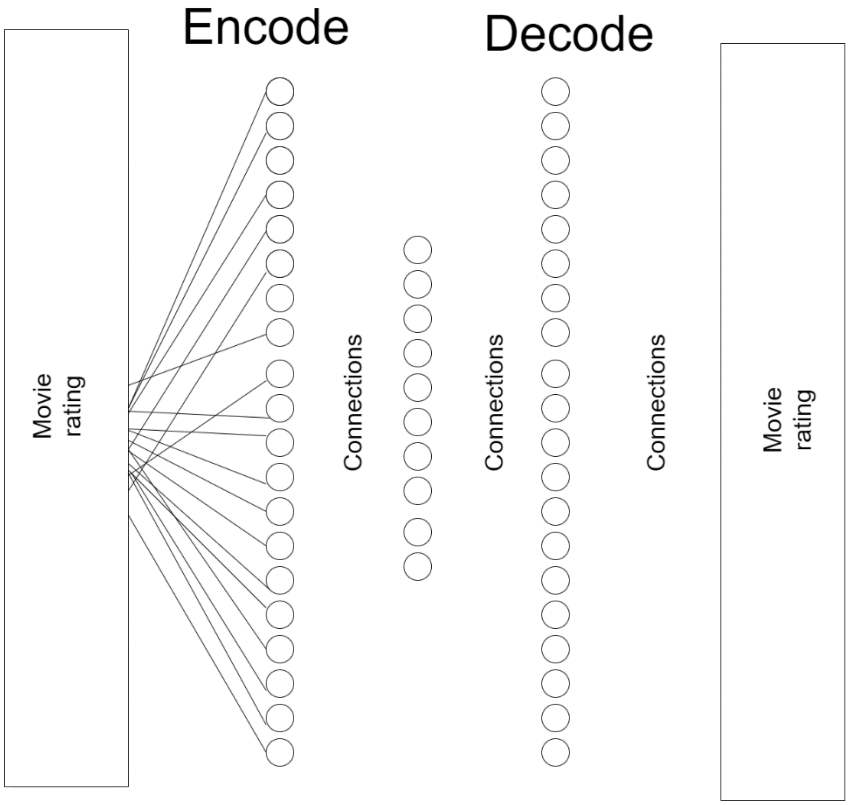
Miner Username: IcyEagle

RMSE: 0.89

Approach

For my Approach I decided to implement a Stacked Auto Encoder since this type of neural network is a recommender system. I began by making a lookup dictionary for all the users and movies assigning them numbers from 0 to n for the length of users and movies as a value and their real user_id/movie_id as the key. This made it a lot easier to work with the data since there were a ton of random id numbers, and this would have made them harder to deal with. I then created user profiles. A list was created for each unique user with a length of all possible movies I was training and testing for. I then set the ratings to 0 except for the ratings the user had already rated. This is the data that I feed to the neural network.

I ended up using a 5-layer auto encoder with 20 nodes and 10 nodes as it encodes and decodes as shown in the figure below. I used RMS prop as my optimizer function with a learning rate of (0.00001).



(Sorry I gave up on drawing all the line connections in this diagram)

When I was done training my model, I ran the neural network one last time on the training data however I did not calculate loss or optimize it this time. I saved the outputs for all the users then I went through my test data and extracted the user and the movie from the results of the neural network. Each

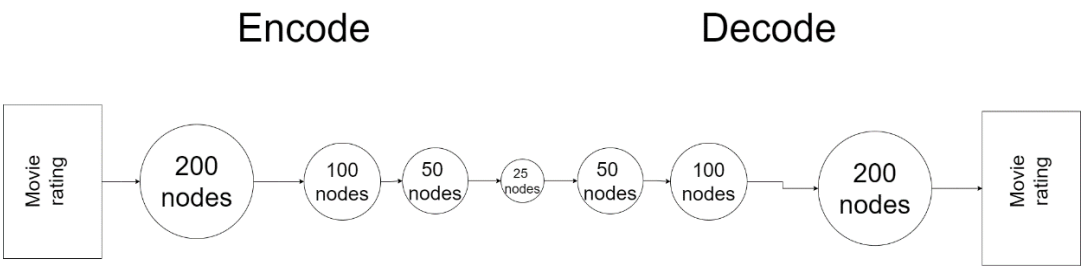
user was extracted then from each user the rating of the movie in their profile movie list was extracted. I then rounded this to the nearest tenth and wrote that to a file.

Methodology

This project was very difficult not because it was hard to build, but because knowing what data is relevant and how to use it was very difficult. At first, I complied a dictionary with all the data files nested inside each other and was thinking about doing a clustering method. After thinking about it I quickly realized how difficult it would be to not only cluster this data accurately but also manage to find accurate recommendations for new test data. So, I decided neural networks may be a better way to go as it learns from trial and error rather than a simple distance formula. Classification using a regular ANN however did not seem like a fun idea since I would have to process a ton of data using the bag of words format and using a ton of files from the given "additional_files" directory ("actors, directors, tags etc). This probably would have been very slow and resource intensive on my computer.

It seemed to me that the best solution would be one that incorporates both clustering of movies and users based on interests while also using a neural network to classify new test cases into specific clusters. Fortunately, I remembered two types of neural networks I learned about last summer that are actually perfect for this problem since they are recommender systems. Restricted Boltzmann machines and Auto encoders. Although I was sure I could have come up with a much more data intensive and messy method to solve this problem. I decided to go with a stacked auto encoder to solve this problem because it seemed like the simplest way to get accurate results using only the train.dat file.

I began testing with a very basic 5-layer neural network with 20 nodes in the second and fourth layers and 10 in the third layer of the encoding/decoding filter. RMSprop was used for my optimizer function and 50 epochs with a learning rate of 0.01. This managed to give me a of about 0.91. I then increased the epochs to 150 thinking it would improve more, however my results decreased to 0.95. I then tried adding more layers and nodes to the neural networking going as far up as 9 layers with 200 nodes in the second layer of the encoding/decoding filter (200 -> 100 -> 50 -> 25 <-50<-100<-200) This did not improve my results however it did not really hurt it either. I then tried increasing and lower the epochs, but this still didn't do much.



I tried changing my optimizer to SGD, but that also didn't improve my performance. Finally, I changed the learning rate on RMSProp and my score i slightly to 0.89. I tried lowering it more, however it started having an inverse effect on my results. I noticed that no matter what I did, the loss would start flattening out around 0.883 on this training set, and if it ever did drop below 0.88 the actual results on miner were bad. So, I thought maybe incorporating dropout to these nodes might help, however that usually had a

negative effect on my results. I think to improve my score further I would have required more rating data, or possibly a better optimizer/optimizer parameter that I am not aware of at the moment.