Benjamin Katz

Final Project

Recomender system

# Link to Colab:

<https://colab.research.google.com/drive/1cEadSla1kyLOwv0m_Jj3ImRX-l9H79nk?usp=sharing>

# Report:

## Methods

A recommender system needs to predict a given score for every user for every item. Every user will have different preferences and every item will have different features. What this means is that there is no easy one to one function of input one item or one user and receive one rating. Instead, a recommender is a many to many function that for every tuple of user and item needs to output one rating.

Some methods to creating these types of systems are as follows:

## Matrix Factorization:

### Description:

Essentially we need to fill in an mxn matrix with users being one axis and movies being the other and at every element (user, movie) we need to put in a rating. One way to fill the matrix is with matrix factorization. Matrix factorization works by taking two matrices, one mxp one pxn with learned elements to multiply together to achieve the target mxn matrix. The elements in these matrices are learned values that are learned through backprop to minimize the loss function of the mean squared error of the target matrix and the training iteration matrices.

### Advantages:

Matrix factorization can utilize collaborative filtering. Because matrix multiplication multiplies row by column, the model can learn to predict a users rating for a given item based on their past ratings of other items.

No content or user information required. The model learns does not need to know anything about the demography of a given user or any information about the makeup of any given item. It learns based on how any given user has rated other items and how other users have rated this specific item

### Disadvantages:

The matrix factorization problem runs into the cold start problem. If a new item is introduced, there is no way to predict how that first user will rate it. If there is no information about a new user, there is no way to predict which items she will rate highly and which she will rate lowly.

Sparsity: when dealing with large data, the matrices will be very sparse which is computationally expensive.

## User centered Content Based model:

### Description:

Content Based models do not just use the user item interaction data but they use information about the content of either the item or the user. Content could include a user’s demography or an item's features. The user centric model only looks at one user with no data about that user herself, along with all the items she interacted with and details about those items. This model is essentially saying that the user that likes items that have *a* and *b* features and dislikes items with *x* and *y* features is likely to like items with *a* and *b* features and dislike items with *x* and *y* features in the future.

### Advantages:

Solves the cold start problem for items. If a new Item comes on the market the model knows who to recommend it to based on the specifications of that item.

It is more explainable, it makes sense why one item is being recommended, not because of convolutional linear algebra, but because the item contains features that the user likes.

Much less sparse than matrix factorization

### Disadvantages:

Does not solve the cold start problem for new users since we have no information about those new users or no which items they like.

Data about each item needs to be available.

## Reasoning:

I chose to build a hybrid out of the matrix factorization model and a user centric content based model. The reason I chose to do this was because there is a plethora of information about each movie available in various databases but there was not any information about the users. If they had information such as the age, gender, city, personality vector, perhaps it would have made sense to build an item centric content based model. I did not have that information so I decided to make a user centric model. Also, I had the intuition that people that have liked one type of movie will like other movies similar to it in the future. I decided to make the two models hybrids by adding the prediction of the matrix factorization as an input feature of the content based DNN. I thought it would make the most sense for a neural network to decide how much weight to give the prediction of the matrix factorization model.

## Progress:

I had spent many hours cleaning these large data frames and merging them into the exact inputs that I needed. I saw there were some movies that had two different movie\_ids so I needed to standardize that. I found a matrix factorization library which was very helpful for making the simple model. I played around with hyper parameters but have been afraid of overfitting. With my training data I was able to get an RMSE of .625 but that lead to overfitting so I ended the training with an RMSE of around .85 which led to an RMSE of the training data of around .95.

Hybrid model: Before I introduced the matrix factorization prediction as a feature to my DNN, I was getting an RMSE of 1.03 with the hybridization I was able to get .98 still not as good as the matrix factorization on its own. With the pure testing data that I only touched at the very end there was an RMSE of .99

## BIbliography:

Thank you to the following resources that have guided my research

<https://blog.insightdatascience.com/explicit-matrix-factorization-als-sgd-and-all-that-jazz-b00e4d9b21ea>

<https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada>

<https://colab.research.google.com/drive/1ksjqKpt-7tTd0cxc7aSm0a6flZGX0rUD?usp=sharing#scrollTo=oB9Y5OTZrTuk>

<https://pypi.org/project/matrix-factorization/#description>