Movie recommendation Project checkpoint

Xinyu Jiang
CSCI6502-001B
xiji6874@colorado.edu

Qinglu Sun
CSCI6502-001B
Qinglu.Sun@colorado.edu

Qiuyang Wang
CSCI6502-001B
Qiuyang.Wang@colorado.edu

1. Introduction

In modern days, watching movies play an essential role in the people's daily entertainment. However, with the development of the Internet, the cost for people finding a movie they want to watch is getting higher and higher. As an information demander, searching useful data from a sea of data is usually a time-consuming work. The recommendation system is an important tool to solve this problem. The recommendation system mines data from existing API, then enables information to be displayed in front of users who are interested in.
Due to that reason, we aim to design a movie recommendation software that collect user's movie watching history, extract movie genre and analyze those data, then finally create list of movies that users potentially interested in and visually display it.

2. Related work

Movie Recommendation engine
The scope of the project is very much similar to Movie Recommendation Engine which aims to recommend movies based on user input. The trending Movie Recommendation Engines on the market are using different algorithms to provide suggestions through a filtering process based on user preferences and other factors.
For instance, Netflix utilises a two-tiered two-based ranking system where ranking happens in each tow (strongest recommendations on the left) and across rows(strongest recommendations on top). Each row highlights a particular theme and is typically generated using one algorithm. The recommendation system developed by Netflix uses variety algorithms, Personalised Video Ranking, Top-N Video Ranker, Trending Now Ranker, etc.

There are 4 types of trending recommendation engines

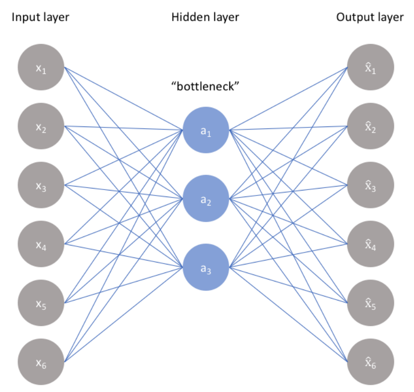
(a) Content-Based
The Content-Based Recommender relies on the similarity of the items being recommended. The basic idea is that if you like an item, then you will also like a "similar" item. It generally works well when it's easy to determine the context/properties of each item.

(b) Collaborative Filtering
The Collaborative Filtering Recommender is entirely based on the past behavior and not on the context. More specifically, it is based on the similarity in preferences, tastes and choices of two users. It analyses how similar the tastes of one user is to another and makes recommendations on the basis of that.

(c) Matrix Factorization
The goal of Matrix Factorization is to learn the latent preferences of users and the latent attributes of items from known ratings (learn features that describe the characteristics of ratings) to then predict the unknown ratings through the dot product of the latent features of users and items.

(d) Deep Learning
The user latent features and movie latent features are looked up from the embedding matrices for specific movie-user combination.

(e) AutoEncoder
Autoencoders are an unsupervised learning technique in which we leverage neural networks for the task of representation learning. Autoencoder takes an unlabeled dataset and frames it as a supervised learning problem tasked with outputting x‘, a reconstruction of the original input x. This network can be trained by minimizing the reconstruction error, L(x,x‘), which measures the differences between the original input and the consequent reconstruction. The bottleneck is a key attribute of our network design; without the presence of an information bottleneck, the network could easily learn to simply memorize the input values by passing these values along through the network.



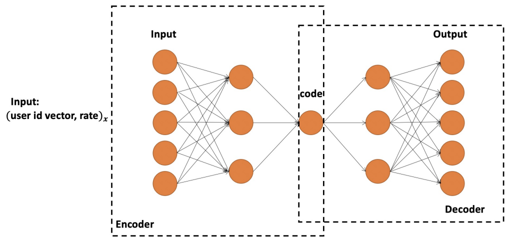
3. Proposed work

(a) Data analysis
Explore popular genres for different ratings(PG,G,R and etc.) and years.
Explore relationship between duration of a movie and its rating,between actors and rating, rating of actor and movie's rating, relationship between director and rating, release date and popularity, language of movie and its rating.

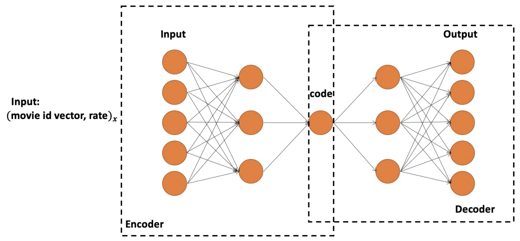
(b) Content based
The content based recommendation system includes item representation which is extract certain feature for each movie, user profile learning which is to learn user's preference profile using movies user like and dislike by using user's rating for each movie in the past and generate recommendation based on a group of movies that is most correlated with user profile. We are going to use TF-IDF and Count Vectorizer for feature selection and normalize feature values so that the representation vectors between different articles are normalized to a magnitude, which is convenient for the operation of the following steps, we are going to use TF-IDF for overview of the movie, it is essential to reduce the weight of terms appearing in a large number of overviews, hence we need IDF. In a large English text corpus, certain words are exceedingly common, hence there is hardly any important information about the actual content of the overview. If we only use the TF of the data to classify the classifier directly, these exceedingly frequent terms will mask the frequency of the rarer but more meaningful terms. Thus these words diminishing the uniqueness of the overview. We will use count vectorizer for combined feature of genres, director and actors. We are going to use KNN, decision tree and linear classifier to build model for determine whether or not user is going to like a new movie using past user's profile preferences about movies based on user's rating for movies. Finally, we will use cosine similarity for generate recommend movies using sorted similarity score.

(c) Collaborative filtering
Collaborative filtering recommendation discovers user preferences by mining user historical preference data, groups users according to different preferences, and recommends products with similar preference. We shall build both user-based and item based Collaborative filtering recommendation. For user-based, calculate the relationship between users according to the degree of preference of different users for the same movie. Recommend movies among users with the same preferences. For item-based, it is quite identical to the user-based collaborative filtering algorithm, which exchange movies and users. The relationship between movies is gathered by calculating the ratings of different users for different movies. Recommend similar movies to users based on the relationship between movies.
Since the encoder only learns its typical features, and the features can be roughly restored to the original data by the decoder, we uses the co-occurrence matrix in collaborative filtering to complete the auto-encoding of the item vector and user vector. In this way, inputting the learned low-dimensional features into the corresponding neural network can greatly reduce the training time.
In our model, we uses information about user such as the user's gender and age, the year of the film, the specific year the user rated the movie such as which day of the year did the user rated the movie and what day of the week that user rated the movie. When we have IDs, they are ordered in the N-dimensional space of the users or the movies. Related movies/users which is based on user ratings are at a short dis-

tance of Euclidean metric. We also uses the vector user ID, vector movie ID, movie data, ratings data and user data for neural network for nthe prediction function. We also uses the region and time zone information, go through linear transormation and output to .NPY model file as the input of autoencoder. The first autoencoder is trained in the assessment of user data using the values of user data of evaluation for this movie, ratings that user rated this movie, rates vector and vector of the user ID, for our first autoencoder, all values are given for a single movie for different rates by users of this movie, for the second autoencoder which is trained on the data of movie rates, we use the values: the movie identifier vector, Information about the movie, which assessed the user, the rates data of the movie by the user, rates vector. Moreover, all values are set for one user and for different movies, which were evaluated by user.



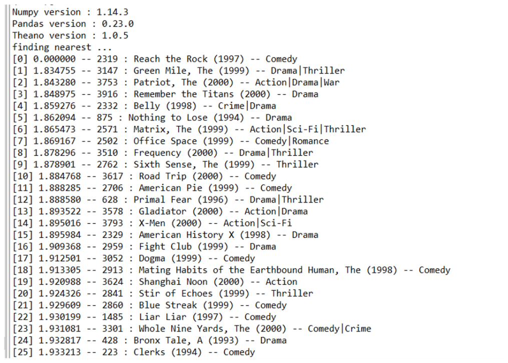
code: movie id vector



code: user id vector

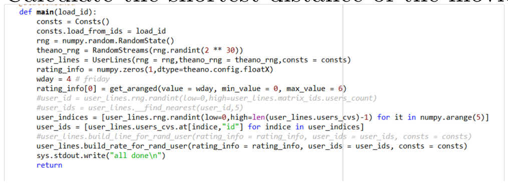
During training process, we are keep records of training loss and verification loss, and we are saving the model every 50 samples trained, we also save movie id, user to movie score, weight and bias. We reiterate our training of neural network and autoencoders to approximate the user evaluation of the movie, we train the neural network for the prediction function on the data on the movie, ratings data, vector movie ID, vector user ID and user data to predict the unseen movie rating which can give the user movies under given conditions, for example if a user have 10 movies watched last week. The most basic collaborative recommendation system might recommend the same movies to him a day, which lacks novelty. By scoring those movies by the day of the week, we can recommend different categories of movies to the user on different day of the week.

.



```
Numpy version : 1.14.3
Pandas version : 0.23.0
Theano version : 1.0.5
finding nearest ...
[0] 0.000000 -- 2319 : Reach the Rock (1997) -- Comedy
[1] 1.834755 -- 3147 : Green Mile, The (1999) -- Drama|Thriller
[2] 1.843280 -- 3753 : Patriot, The (2000) -- Action|Drama|War
[3] 1.848975 -- 3916 : Remember the Titans (2000) -- Drama
[4] 1.859276 -- 2332 : Belly (1998) -- Crime|Drama
[5] 1.862094 -- 875 : Nothing to Lose (1994) -- Drama
[6] 1.865473 -- 2571 : Matrix, The (1999) -- Action|Sci-Fi|Thriller
[7] 1.869167 -- 2502 : Office Space (1999) -- Comedy|Romance
[8] 1.878296 -- 3510 : Frequency (2000) -- Drama|Thriller
[9] 1.878901 -- 2762 : Sixth Sense, The (1999) -- Thriller
[10] 1.884768 -- 3617 : Road Trip (2000) -- Comedy
[11] 1.888285 -- 2706 : American Pie (1999) -- Comedy
[12] 1.888580 -- 628 : Primal Fear (1996) -- Drama|Thriller
[13] 1.893522 -- 3578 : Gladiator (2000) -- Action|Drama
[14] 1.895016 -- 3793 : X-Men (2000) -- Action|Sci-Fi
[15] 1.895984 -- 2329 : American History X (1998) -- Drama
[16] 1.909368 -- 2959 : Fight Club (1999) -- Drama
[17] 1.912501 -- 3052 : Dogma (1999) -- Comedy
[18] 1.913305 -- 2913 : Mating Habits of the Earthbound Human, The (1998) -- Comedy
[19] 1.920988 -- 3624 : Shanghai Noon (2000) -- Action
[20] 1.924326 -- 2841 : Stir of Echoes (1999) -- Thriller
[21] 1.929609 -- 2860 : Blue Streak (1999) -- Comedy
[22] 1.930199 -- 1485 : Liar Liar (1997) -- Comedy
[23] 1.931081 -- 3301 : Whole Nine Yards, The (2000) -- Comedy|Crime
[24] 1.932817 -- 428 : Bronx Tale, A (1993) -- Drama
[25] 1.933213 -- 223 : Clerks (1994) -- Comedy
```

Calculate the shortest distance of the movie



```
def main(load_id):
    consts = Consts()
    consts.load_from_ids = load_id
    rng = numpy.random.RandomState()
    theano_rng = RandomStreams(rng.randint(2 ** 30))
    user_lines = UserLines(rng = rng,theano_rng = theano_rng,consts = consts)
    rating_info = numpy.zeros(1,dtype=theano.config.floatX)
    wday = 4 # friday
    rating_info[0] = get_aranged(value = wday, min_value = 0, max_value = 6)
    #user_id = user_lines.rng.randint(low=0,high=user_lines.matrix_ids.users_count)
    #user_ids = user_lines.__find_nearest(user_id,5)
    user_indices = [user_lines.rng.randint(low=0,high=len(user_lines.users_cvs)-1) for it in numpy.arange(5)]
    user_ids = [user_lines.users_cvs.at[indice,"id"] for indice in user_indices]
    #user_lines.build_line_for_rand_user(rating_info = rating_info, user_ids = user_ids, consts = consts)
    user_lines.build_rate_for_rand_user(rating_info = rating_info, user_ids = user_ids, consts = consts)
    sys.stdout.write("all done\n")
    return
```

Recommend movies for users on Friday for user id 3970

4. **Data**
   We are going to use Open Movie Database(OMDB) for the project, which is RESTful web service. OMDB provides movie information with full content and images with free api calls up to 1000 per day.

5. **Evaluation**

   (a) Score Prediction
   RMSE and MAE, the smaller the error and the better the performance of the recommended system, calculated by difference between user's rating score for the movie and predicted score for that movie given by the recommendation system.

   (b) Top N recommendation
   Calculated by precision and recall based on the user's preference list on the training set, and the user's preference list on the test set.

   (c) Coverage
   Coverage rate indicate that the proportion

of the recommendation movies to the total collection space to discover long tail of items, calculated using information entropy and Gini index.

6. **Milestone**
   We are looking for achieving the following of the milestones we set out at the beginning of the project:

   - **Milestone 1:** Finalizing the topic of the project. In addition to that, we set up group meeting time, venison control software(Github).

   - **Milestone 2:** Instead of building up web-application to visualize recommender, we are considering of using other

   - **Milestone 3:** Connect the front-end and back-end together and start to extract data from the OMDB API.

   - **Milestone 4:** Writing the movie recommendation system base on the OMDB API and user's watching history.

   - **Final Milestone:**Create a visual display of movie list that users potentially interested in, including movie genres and basic introduction of each movies.

References

1. David Chong. 2020. Deep Dive into Netflix's Recommender System. (May 2020). Retrieved February 20, 2021 from https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48

2. James Le. 2018. The 4 Recommendation Engines That Can Predict Your Movie Tastes. (June 2018). Retrieved February 20, 2021 from https://towardsdatascience.com/the-4-recommendation-engines-that-can-predict-your-movie-tastes-109dc4e10c52

3. Jeremy Jordan 2018. Introduction to autoencoders. https://www.jeremyjordan.me/autoencoders/