MA 544 Final Project
Recommendation Systems
By:
Gokul
Nicholas
Samruth

Abstract:

There are many ways to develop a recommendation system. But which way is the best for getting users to try new things? This experiment aims to test this by looking at two types of recommendation systems (K means clustering and SVD clustering) and, using an identical movie reviews dataset, test to see which system will provide a better recommendation for a single user.

Introduction:

What are Recommendation Systems:

A recommendation system is a subclass of information filtering that seeks to predict the preference a user would give to an item. In simpler terms, it tries to guess how much a particular user would like (or dislike) a particular item given external, but relevant, information. Recommendation systems are usually used in a variety of areas, but they are mostly used by companies to help “push” customers to buy new things based on their existing interests.

The bread and butter of a recommendation system is their filtering system. This is how the recommendation system predicts if a user likes (or dislikes) any given item. There are two main filtering systems, Collaborative filtering and Content based filtering

Collaborative Filtering:

Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past. The system generates recommendations using only information about rating profiles for different users or items. By locating peer users/items with a rating history similar to the current user or item, they generate recommendations using this neighborhood. For example, if Person A likes Item X, and Person B likes Item X and Y and Person C likes Item X, Y and Z, Collaborative filtering will predict that Person A will most likely like Item Y since users who like Item X have a higher chance of liking Item Y. A key advantage with Collaborative filtering is that the model isn’t required to have a deeper knowledge of the content it is predicting. For example, movies tend to be complex as there are multiple combinations for movies (Dark Comedy vs Family Comedy are vastly different experiences despite falling under the Comedy umbrella). However, under Collaborative filtering the model does not need to take into account these complexities as it is only focusing on what similar users enjoy. That being said Collaborative filtering runs into a few problems. For one, if a new item is introduced, it is harder to make accurate predictions as there is no prior data to pull from. Moreover, the more items/users in a given system the more computational power is required to make predictions. Finally, if a system is big enough the model runs into the problem of sparsity, where even popular items will not have that many ratings which leads to a lot of dead space in the data set.

Content Based Filtering:

Content-based filtering methods are based on a description of the item and a profile of the user's preferences. These methods are best suited to situations where there is known data on an item (name, location, description, etc.), but not on the user. Content-based recommenders treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on an item's features. For example, if Person A likes Item X which has tags 1 and 2, the model will recommend Item Y which also has tags 1 and 2.The main benefit of Content Based filtering is that recommending new items is significantly easier as each new item comes with their tags which the model can use to recommend to users. However, the biggest downside is that it tends to pigeonhole recommendations. For example, if a user watches a Comedy then CB filtering will only recommend comedies as there is that similar tag. But it will never recommend a Horror movie as most horror movies tend to not have comedy in it, which thereby pigeonholes the user to only watch comedies.

The Mathematics behind a Recommendation System:

One of The main mathematical concepts used in this particular experiment will be K-means Clustering. In essence K means Clustering is a method of vector quantization that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster.

In more mathematically terms, Given a set of observations (x1, x2, ..., xn), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k (≤ n) sets S = {S1, S2, ..., Sk} so as to minimize the within-cluster sum of squares or:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

Where the mean is the mean of points in Si

In terms of a recommendation system K means clustering would work with a hybrid of content based and collaborative filtering. The only issue would be determining the number of clusters required to conduct the clustering. To solve this the elbow method will be used, which works by plotting the ascending values of K versus the total error obtained when using that K.The goal here being to find the ka that for each cluster will not rise significantly the variance.

The other concepted used in this experiment is SVD. SVD is a matrix factorisation technique, which reduces the number of features of a dataset by reducing the space dimension from N-dimension to K-dimension (where K<N) using:

$$A = USV^T$$

Where A is a m x n utility matrix, U is a m x r orthogonal left singular matrix, S is a r x r diagonal matrix and V is a r x n diagonal right singular matrix. In the context of the recommender system, the SVD is used as a filtering technique. It uses a matrix structure where each row represents a user, and each column represents an item. The elements of this matrix are the ratings that are given to items by users.

Data Set:

The data set used in this experiment comes from MovieLens data set (https://grouplens.org/datasets/movielens/) . For runtime sake the 100k size data set will be used. Per the data set's ReadMe, there are three csv files; Ratings, Tags, and Movies. Ratings contain userId, movieId, rating and timestamp. Tags contain userId, movieId, tags and timestamp. Finally Movies contain movieId, title and genre. Movies are imported from https://www.themoviedb.org/, and include the year of release in parentheses. Ratings are made on a 5 -star scale, with half star increments included

Results:

Code:

For the SVD Recommendation System, the team used the in built python SVD() and other functions to build the recommendation system model

In [ ]:
```
df_title = pd.read_csv('../input/movie_titles.csv', encoding = "ISO-8859-1", header = None, names = ['Movie_Id', 'Year', 'Na
df_title.set_index('Movie_Id', inplace = True)
print (df_title.head(10))
```

In [39]:
```
reader = Reader()

# get just top 100K rows for faster run time
data = Dataset.load_from_df(ratings_data[['userId', 'new_movie_id', 'rating']][:], reader)
#data.split(n_folds=3)

svd = SVD()
cross_validate(svd, data, measures=['RMSE', 'MAE'])
```

Out[39]:
```
{'test_rmse': array([0.87007861, 0.87520837, 0.87542801, 0.87252011, 0.869551  ]),
 'test_mae': array([0.6684968 , 0.67203905, 0.6709185 , 0.66903059, 0.66902215]),
 'fit_time': (3.4100866317749023,
  3.498782157897949,
  3.44977068901062,
  3.565463066101074,
  3.4956393241882324),
 'test_time': (0.12617945671081543,
  0.12566661834716797,
  0.09174680709838867,
  0.08976411819458008,
  0.13264799118041992)}
```

In [66]:
```
f = ['count','mean']

df_movie_summary = ratings_data.groupby('movieId')['rating'].agg(f)
df_movie_summary.index = df_movie_summary.index.map(int)
movie_benchmark = round(df_movie_summary['count'].quantile(0.7),0)
drop_movie_list = df_movie_summary[df_movie_summary['count'] < movie_benchmark].index
```

Finally a method was created that will take in that model and given an user id, print out a list of movies that would best fit said user's interests.

In [67]:
```
def movie_recs(user):
    user_liked_movies = ratings_data[(ratings_data['userId'] == user) & (ratings_data['rating'] == 5)]
    user_liked_movies = user_liked_movies.set_index('movieId')
    user_liked_movies = pd.merge(movies_data, user_liked_movies, on="movieId")
    Movies_liked = user_liked_movies['title']

    user_liked_movie = movies_data.copy()
    user_liked_movie = user_liked_movie.reset_index()
    user_liked_movie = user_liked_movie[~user_liked_movie['movieId'].isin(drop_movie_list)]

    # getting full dataset
    data = Dataset.load_from_df(ratings_data[['userId', 'movieId', 'rating']], reader)

    trainset = data.build_full_trainset()
    svd.fit(trainset)

    user_liked_movie['Estimate_Rating'] = user_liked_movie['movieId'].apply(lambda x: svd.predict(user, x).est)

    user_liked_movie = user_liked_movie.drop('movieId', axis = 1)

    user_liked_movie = user_liked_movie.sort_values('Estimate_Rating', ascending=False)
    recommended_movies = user_liked_movie[['title','Estimate_Rating']]

    print('Movies The User Likes')
    print('')
    print(Movies_liked.head(20))
    print('')
    print('')
    print('Recommended Movies with estimated Ratings:')
    print('')
    print(recommended_movies.head(10))
```

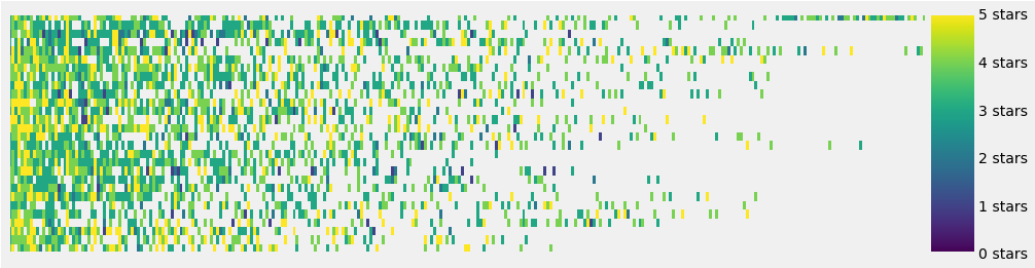
For the K Means Clustering, a sparse data frame was created with the columns being movies and the rows being users. If the user saw the movie the rating would be in the matrix at that position. An arbitrary number of clusters (15) was taken. The algorithm was run on the compressed sparse row (CSR) matrix. The data was classified into 15 distinct user groups. A heatmap can display a single group's user ratings on movies for a cluster.

Forrest Gump (1994)	Men in Black (a.k.a.	Jurassic Park (1993)	Matrix, The (1999)	Back to the Future	Fight Club (1999)	Indiana Jones and the Last	Star Wars: Episode IV - A New	Pulp Fiction (1994)	Armageddon (1998)

Finally, the average rating of each movie in the cluster can be taken and ranked from largest to smallest. That list of movies is compared to the list of the movies the user has already. This way the recommendations made are not movies the user has already seen. The top 20 movies the user has not seen were kept. Finally, the recommendation system would recommend a random 4 movies from that list.

Results:

The SVD Recommendation System results are as follows:




```
In [69]:  1  movie_recs(user = 143)
```

```
Movies The User Likes

0             Forrest Gump (1994)
1          Sixteen Candles (1984)
2                   Shrek (2001)
3           Legally Blonde (2001)
4      Walk to Remember, A (2002)
5             Uptown Girls (2003)
6          Chasing Liberty (2004)
7           50 First Dates (2004)
8           13 Going on 30 (2004)
9      Bill Cosby, Himself (1983)
10                    Up (2009)
11                Fired Up (2009)
12           Hangover, The (2009)
13    (500) Days of Summer (2009)
14                Tangled (2010)
Name: title, dtype: object


Recommended Movies with estimated Ratings:

                                              title  Estimate_Rating
937     Seventh Seal, The (Sjunde inseglet, Det) (1957)         4.245808
512                         Beauty and the Beast (1991)         4.223449
6571                            Eastern Promises (2007)         4.192814
7467                                      Tangled (2010)         4.174376
3194                                        Shrek (2001)         4.164004
975                                Cool Hand Luke (1967)         4.158392
711                                     Notorious (1946)         4.132487
966                    Manchurian Candidate, The (1962)         4.123778
4076     Harry Potter and the Chamber of Secrets (2002)         4.100718
228   Like Water for Chocolate (Como agua para choco...         4.092270
```

The K Means Clustering Results are as follows:

```
Users ratings for movies in cluster:
Kill Bill: Vol. 2 (2004)      5.0
Gone with the Wind (1939)     5.0
Finding Neverland (2004)      5.0
Forrest Gump (1994)           5.0
Clockwork Orange, A (1971)    5.0
Name: 142, dtype: float64
User 143 has 77 movie reviews


Movies Average Rating in Cluster not seen by user:

Ratatouille (2007)                                  5.000000
African Queen, The (1951)                           4.833333
Cinema Paradiso (Nuovo cinema Paradiso) (1989)      4.812500
Psycho (1960)                                       4.750000
Stand by Me (1986)                                  4.636364
```

Observations:

Looking at the team's results for user 143 it can be said that the SVD recommendation system, while fairly accurate, does have some slight issues. For example, user 143 is stated to have liked Shrek and Tangled, yet the recommendation system predicts that the user will like Shrek and Tangled. This result, while true, doesn't really provide anything of value for the user. On the other hand, if we look at user 143's liked movies we can see that generally speaking the user likes more light hearted family friendly movies and dramas. Going through the recommended movies we can see a lot of movies that fit the same description. Movies like Harry Potter and the Chamber of Secrets, Beauty and the Beast and Cool Hand Luke fit either the family friendly movie niche or the drama niche.

Looking at the results for K means clustering, The system would need more research. It is difficult to know what the appropriate number of clusters would be. Too many clusters and there are not enough users in each group in order to make recommendations. Too few clusters and the recommendations are not specific enough. The highest rated movies would constantly be recommended to every user. Further analysis would have to be done to optimize that decision. Along with the number of clusters, the process could be optimized, by dynamically determining the cutoff of movies in each group. Right now, an arbitrary 20 movies were chosen to recommend. The cutoff can be done by rating and not the top n number of movies.

Conclusion:

In conclusion, looking through the results of the experiment the team can gather that while the SVD system is overall faster (on average it tends to take 4.5 seconds for this dataset), K-means will be more customizable because you can change the cluster number which allows for more precision recommendations.