NYU TANDON SCHOOL OF ENGINEERING

Final Project

Movie recommender

Team Member:
Xiantian Sun. xs884
Yufei Ren yr873
Ziyu Su. zs1254

Problem Statement

•Using collaborative filtering and PCA to build a recommender based on user's past rating of movies

•Predict or filter preferences of movie according to the user's choice.

•Using dataset from http://files.grouplens.org/datasets/movielens/ml-latest-small.zip

•Dataset contains movie dataset(movies information) and rating dataset(userID, moiveId, rating).

Movie Dataset(9743 movies)

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
5	6	Heat (1995)	Action|Crime|Thriller

*

Rating Dataset(100836 rating data)

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
5	1	70	3.0	964982400

*

Loss function

$$L = \sum (X - Xhat)\ \text{^}2$$

Formulation

•Ratings for movies dependent on small number of latent factors.

•Mathematically model as:

$$R_{ij} \approx \hat{R}_{ij} = b_i^u + b_j^m + \sum_{k=1}^{K} A_{ik} B_{jk}$$

- R_{ij} =Rating of movie j by user i
- b_i^u =Bias of user i
- b_j^m =Bias of movie j
- K = number of latent factors. Typically small $K \ll N_{user}, N_{movies}$
- A_{ik} = Preference of user i to factor k
- B_{jk} = Component of factor k in movie j

Cited: Formulation from Pro. Sundeep Lec11 https://github.com/sdrangan/introml/blob/master/lectures/Lect11_PCA.pdf

Training Procedures

•Find all the ratings for each movie and construct as ratings matrix

•Remove the mean for all the non-zero values.

•Use PCA on the data and use K-fold to determine how many PCs can get the best accuracy.

•Find the the prediction of X based on the PCA.

K-fold

As we use more and more PCs to compute the prediction, we can see our model perform better and better. Accordingly the runtime will increase. Since long runtime for this project, we used colab GPU to run the project.

```
nfold = 5

# Create a K-fold object
kf = KFold(n_splits=nfold)
kf.get_n_splits(X)

# Number of PCs to try
ncomp_test = np.arange(100,400,10)
print(ncomp_test)
num_nc = len(ncomp_test)

# Accuracy:  acc[icomp,ifold]  is test accuracy when using `ncomp = ncomp_test[icomp]` in fold `ifold`.
RMSE = np.zeros((num_nc,nfold))

# Loop over number of components to test
for icomp, ncomp in enumerate(ncomp_test):
    print(ncomp)
    # Look over the folds
    for ifold, I in enumerate(kf.split(X)):
        Itr, Its = I
        #print(ifold)
        # TODO:  Split data into training
        Xtr, Xts= Xs[Itr],Xs[Its]

        # TODO:  Create a scaling object and fit the scaling on the training data
        pca = PCA(n_components = ncomp)

        # TODO:  Fit the PCA on the scaled training data
        pca.fit(Xtr)
        Ztr = pca.transform(Xtr)

        # TODO:  Transform the test data through data scaler and PCA
        Zts = pca.transform(Xts)
        Xts_hat = np.dot(Zts,pca.components_)

        # TODO:  Measure the accuracy
        RMSE[icomp, ifold] =np.sqrt(np.mean((Xts - Xts_hat)**2))
```

*

Evaluation Result

Recommend user several movies that user never watched based on recommender system

```
import re
def recommend(user, k=5):
    recomm = []
    np_movie_Id = np.array(movies.movieId)
#     np_movie_title = np.array(movies.title)
    for i in range(Y.shape[1]):
      n = np.argwhere(np_movie_Id==i)
      if len(n)>0 and X[user,i] == 0:
#           print(n)
          s = str(movies.title[n[0]])
          s = re.sub('\nName: title, dtype: object', '', s)
          recomm.append((s, Y[user,i]))
    recomm.sort(key=lambda val:val[1], reverse=True)
    return recomm[:k]
```

```
#Recommend movies for user 1
RECOM = recommend(1)
print (RECOM)
```

```
[('3248    Rape Me (Baise-moi) (2000)', 3.3283647665525002), ('2218    Goldfinger (1964)', 3.
3278125327181014), ('2845    Sinbad and the Eye of the Tiger (1977)', 3.3278125327181014), ('
2881    Tao of Steve, The (2000)', 3.3278125327181014), ('2913    Urban Legends: Final Cut (2
000)', 3.318856691084882)]
```

*

Conclusion

•The key method of this project is PCA and SVD. PCA can help us get rid of some noise of the dataset and find out several key factors(principle component), then we can use the component to do the reverse transform to get the predicted rating of each movie. We also do k-Fold, calculate the accuracy by comparing the predicted data and original data, compute the MSE on the units that is non-zero in the raw dataset.

•To further improve the recommending accuracy, we will try to use a much larger dataset next time and try several other machine learning algorithms.