Rating Prediction Task on Movielens Dataset Using K-Latent Factor Model

Linbin Yang	Yufeng Gao	Bosi Cheng
A53277054	A53274343	A53271697
l2yang@eng.ucsd.edu	y2gao@eng.ucsd.edu	bocheng@eng.ucsd.edu

ABSTRACT

Recommender systems have become ubiquitous in our lives. However, they are far from optimal. For this project, we focus on rating prediction task in recommendation system. The dataset we use is Movielens , one of the most common datasets for building a Recommender System. We put forward K-Latent Factor Model (KLF) in this paper and compare it with current rating prediction methods including Deep learning method and Collaborating Method. We evaluate the performance of the three Models on the Movielens dataset through their running time and Root Mean Square Error (RMSE) and summarize the pros and cons of the three models.

Key Words: Recommendation System, Rating Prediction, Latent Factor Model

1. INTRODUCTION

A recommender system is a type of information filtering system which attempts to predict the preferences of a user and make suggestions based on these preferences. There are a wide variety of applications for recommendation systems and they have become increasingly popular over the last few years and are now utilized in most online platforms that we use. The content of such platforms varies from movies, music, books and videos, to friends and stories on social media platforms, to products on e-commerce websites, to people on professional and dating websites, to search results returned on Google.

Traditionally, there are two main types of methods to construct a recommender system: Collaborative filtering approaches and content-based filtering. Collaborative filtering approaches build a model from a user's past behaviour (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in. Content-based filtering approaches utilize a series of discrete characteristics of an item in order to recommend additional items with similar properties.
Every user has different preferences and likes. In addition, even the taste of a single user can vary depending on a large number of factors, such as mood, season, or type of activity

the user is doing. For example, the type of music one would like to hear while exercising differs greatly from the type of music he'd listen to when cooking dinner. Another issue that recommendation systems have to solve is the exploration vs exploitation problem. They mustexplore new domains to discover more about the user, while still making the most of what is already known about of the user.

Due to the quick development in recommender systems, users constantly expect good recommendations. Now in most recommender systems, rating serves as an important indicator that reflect users' preference. For examples, when we buy things online through Amazon or Alibaba, customers are often asked to rate the product they have bought, score ranges from 0-5. In this paper, we seek to build efficient model to predict rating scores for user on things he or she might never encounter before.

The method we use is K-Latent-Factor Model, our experiment shows that it outperforms traditional collaborate filtering method and deep learning model on MovieLens dataset. The final rating score we got from our prediction model can be widely applied in recommender system. For examples, for this final project, we use our K-Latent-Factor Model to generate a ranking list for one specific user on movies he never watches before and we select top 100 movies title and its corresponding type to mine key words that represents this user's taste. Our work is shown on the experiment and analysis part of this paper.

Part 3 of this paper would introduce the dataset we use, the running mechanisms of K-Latent-Factor Model, Deep Learning Model and Collaborating Filtering Model. Part 4 is about how we set up our experiment and performance of the three models. In part 5, we give our analysis based on the performance shown on part 4.

2. METHODOLOGY

2.1. Dataset Analysis

The dataset we applied is one-million Movielens dataset. All together there are 1,000,209 rating record. Some basic characteristics of this dataset in concluded in the table shown below:

Table 1 Basic Characteristic of Movielens Dataset

Rating Record	Total Number of Users	Total Number of Movies
1,000,209	6040	3883

To train our model on this dataset, we split the data into training set (90%) and validation set (10%). We do a simple analysis on the rating score which is the goal of our prediction model.

We find that most users tend to give high rating score (>=4) on the movies they watch. This means that, we definitely cannot compare scores between different users because

different users have different rating tastes. This exactly explains why collaborating filtering model does not perform well as it puts in our experiment part. Further explanations about this is in the analysis part of this paper.

In our assignment, we applied K-Latent Factor Model to predict the possible rating a user may give to a movie. Besides, we also applied Collaborative Filtering Model and Deep Learning Model in order to make comparison. Details are as follows.

2.2. K-Latent Factor Model

Latent Factor Model is a statistical model that relates a set of observable variables (so-called manifest variables) to a set of latent variables. The Simplest method for this prediction task is:

$$f(u,i) = \alpha + \beta_u + \beta_i$$

However, this equation has limitation, that is, it ignores the correlations between users and items, we just treat the user and item independently. In our case, we put forward K-latent-factor model that takes the relation between users and items into considerations. The equation is:

$$f(u,i) = \alpha + \beta_u + \beta_i + \lambda_u * \lambda_i$$

Where the dimensions of λ_u and λ_i are $m * k$ and $k * n$ matrix. k is the hyper parameters, m is the number of users and n is the number of items. Below is our objective function:

$$Obj = \sum_{u,i} (\alpha + \beta_u + \beta_i + \lambda_u * \lambda_i - R_{u,i})^2 + \lambda(\sum_u {\beta_u}^2 + \sum_i {\beta_i}^2 + \sum_u ||\lambda_u||_2^2 + \sum_i ||\lambda_i||_2^2)$$

Generally speaking, there are two methods to optimize objective function, one is Alternating Least Square Method (ALS) and another is Stochastic Gradient Descent (SGD). Here, we use (SGD) to optimize this objective function. Because, the objective function is non-convex. The derivative function for λ_{uk} is as follows:

$$\frac{dObj}{d\lambda_{uk}} = \sum_{i \in I_u} 2 * (\alpha + \beta_u + \beta_i + \lambda_u * \lambda_i - R_{u,i}) * \lambda_{ik} + 2 * \lambda * \lambda_{uk}$$

In our task, wo set appropriate parameters K, λ and use MSE to measure the performance of the model during each iteration in the training process.

2.3. Collaborative Filtering Model

Much of research in recommender systems is primarily influenced by research in information retrieval. Collaborative filtering is the basic technique implemented in research of recommendation system. Collaborative filtering method give recommendations of items for users based on patterns of ratings or usage (e.g., purchases) without the need of external information about user or items. More often information about users and/or items is difficult to capture so application of Collaborative filtering as method seems reasonable.

For each user, recommender systems recommend items based on how similar users liked the item. For example, A and B have similar interests in movies. A recently watched the movie "Chinese Dream in America" and gave it rating score of 4.9. B has not watched this movie, but the system has learned that A and B have similar tastes, it recommends this movie to B. In addition to user similarity, recommender systems can also perform collaborative filtering using item similarity. In our final project, we use Collaborating Filtering Method to do rating prediction.

We measured the Pearson similarity between users and objects. For rating prediction, we do the following calculation:

$$P_{rating(u,j)} = \frac{1}{len(U)} \sum_{i=1}^{n} Sim(u, u_i) * r_{uj}$$

For each movie j that user i has not seen yet, we find the set of users U who are similar to user i and have seen movie j. For each similar user u, we take u's rating of movie j and multiply it by the Pearson similarity of user i and user u. Sum up these weighted ratings, divided by the number of users in U, and we get a weighted average rating for the movie j.

Pearson's correlation coefficient is the test statistics that measures the statistical relationship, or association, between two continuous variables. It is known as the best method of measuring the association between variables of interest because it is based on the method of covariance. It gives information about the magnitude of the association, or correlation, as well as the direction of the relationship.

2.4. Deep Learning Model

Deep Learning (DL) is one of the next big things in Recommender Systems. During the past few years neural networks have shown tremendous success in computer vision, speech recognition, and natural language processing (NLP). Deep learning methods are also becoming a powerful tool to tackle Recommender Systems tasks such as music, news, fashion articles, and mobile apps recommendation.

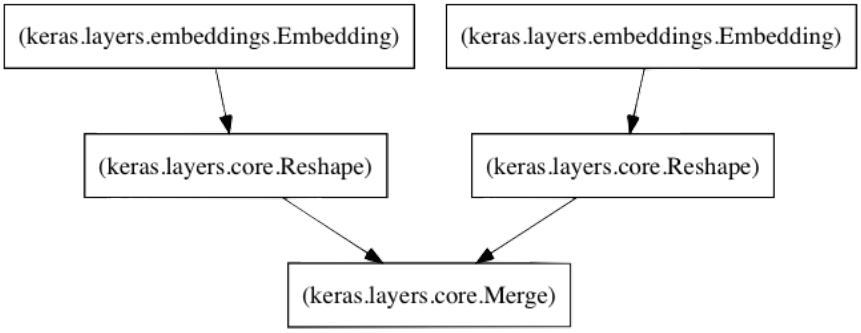


Figure 2 Deep Learning Model on Movie Rating Prediction

Different from the previous collaborative filtering, in deep learning implementation, the model learns the values of embedding matrix itself (See Figure 2). The user latent features and movie latent features are looked up from the embedding matrices for specific movie-user combination. These are the input values for further linear and non-linear layers. We can pass this input to multiple linear or sigmoid layers and learn the corresponding weights by any optimization algorithm.

During the training process above, we saved the model weights each time the validation loss has improved. Thus, we used that value to calculate the best validation RMSE.

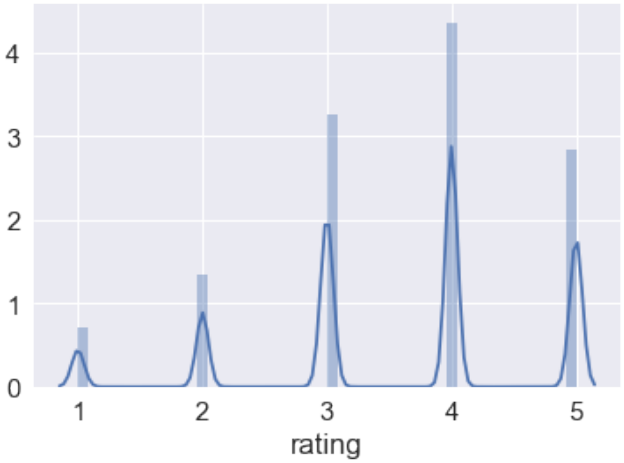
3. EXPERIMENT

3.1. Dataset

The dataset we use for our experiments is MovieLens dataset, one of the most popular dataset available for training and testing a recommender system. The author of this dataset provides datasets of different sizes, for example, 100K, 1M, 10M, 20M. Because most models we read about in other research teams' papers were trained on the 1M size MovieLens dataset, we decided to carry out all our experiments on it so as to make reasonable comparisons.

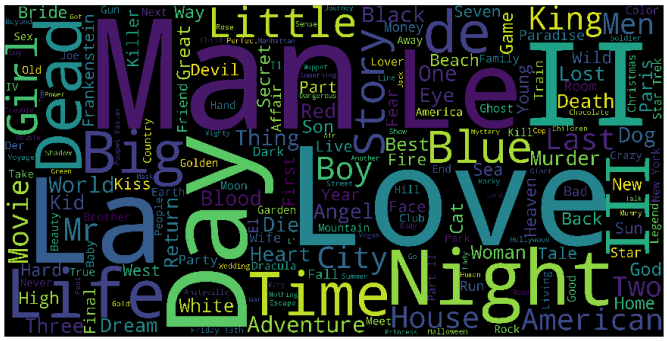
This version of the dataset contains 1000209 anonymous movie rating records of 3883 movies made by 6040 MovieLens users who joined the community in 2000. After performing a simple explanatory analysis on the data, here are some basic statistics and properties of it:

(1) Distribution of user ratings:

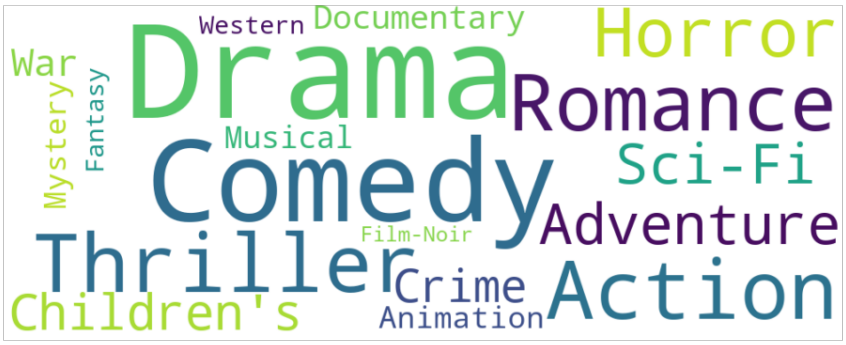


It appears that users are quite generous in their ratings. The mean rating is 3.58 on a scale of 5. Half the movies have a rating of 4 and 5. I personally think that a 5-level rating skill wasn't a good indicator as people could have different rating styles (i.e. person A could always use 4 for an average movie, whereas person B only gives 4 out for their favorites). Each user rated at least 20 movies, so I doubt the distribution could be caused just by chance variance in the quality of movies.

(2) A word-cloud visualization of the movie genres:



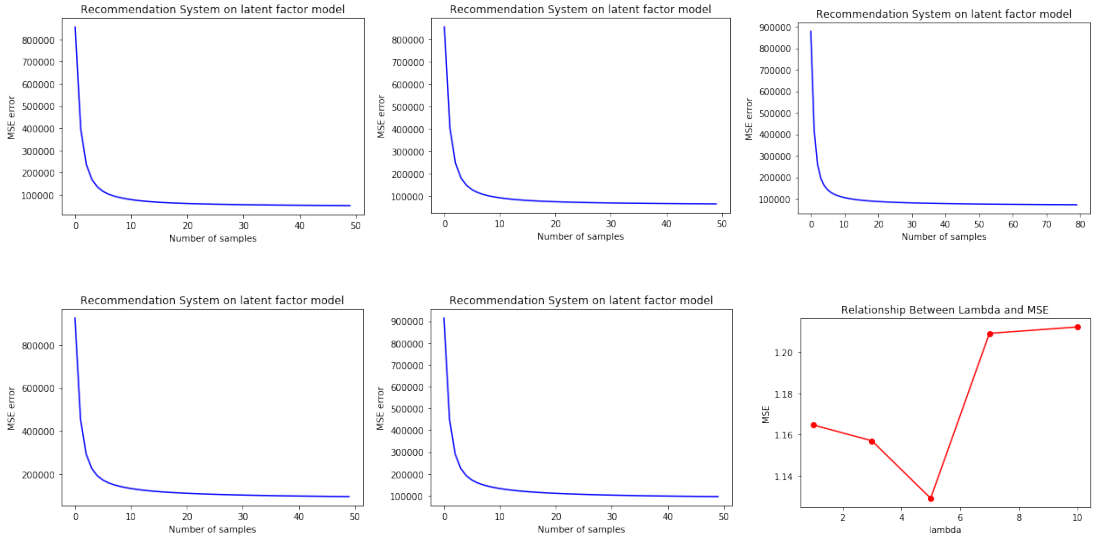
(3) word-cloud of the **movie genres**



3.2. K-Latent-Factor-Model Experiment

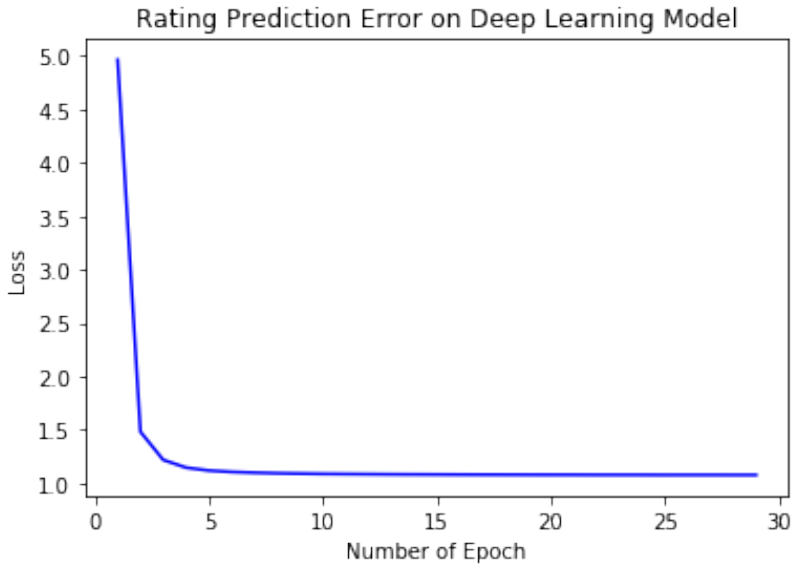
For this model, we used Tensorflow 0.12.1 and keras 1.2.0, and experimented with K = 3, Lambda = [1, 3, 5, 10], learning rate = 0.0001 and used Stochastic Gradient Descent as the optimizing method.

Because we experimented with five different Lambdas, we obtained five training process graphs which demonstrate the relationship between the model's MSE on the training data and the number of samples. (The MSE is scaled up by a factor of 50000) and we also drew a graph to explore the relationship between MSE and the choice of lambda value.



3.3. Deep Learning Model

Below is the training process graph of the deep learning model which indicates the relationship between the training loss and the number of epochs. From this graph we can see, after about 8 epochs, our model converges to 1.1.



When we finally conducted the prediction job with our model on the test data, the RMSE is approximately 1.0437.

4. ANALYSIS

From above experiment results we can see that deep learning model and K latent factor model can perform a better prediction job than collaborative filtering model in that the latter one gives a final MSE on the test which is over 1000 dataset while for both the other two models, MSE on the dataset is around 1. This may be caused by the differences between underlying principles of these algorithms. …

For collaborative filtering model, despite the fact that the author only used 20% of the whole 1M dataset and got a terrible result, we tried to implement this model, only to find that average computers cannot successfully complete the training process in a reasonable amount of time. On the other hand, it took 2 hours to training our K latent factor model and less than 1 hour to training the deep learning model. Even though the training time can be influenced by multiple parameters like K of the latent factor model and the learning rate of the deep learning model, for now, we can say, in terms of the time needed for training, the winner is absolutely the deep learning model.

To demonstrate the practical value of these rating prediction models, here we give two example of how to use these predictions.

(1) Recommender system:

We can use predicted ratings to make recommendations. As we know, these models predict the rating of a user on a movie he/she hasn't rated yet, thus, we can confidently recommend those movies predicted to receive high ratings by a specific

user to him/her. For example, for the first user in MovieLens dataset, based on the predicted ratingour latent factor model

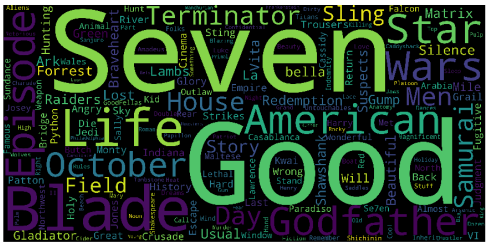
For example, for the first user in MovieLens dataset, the top 25 movies are:

1	Shawshank Redemption	4.93779912379
2	Raiders of the Lost Ark (1981)	4.79908361601
3	Usual Suspects	4.73251760317
4	Braveheart (1995)	4.71826224882
5	Matrix	4.71378360453
6	Silence of the Lambs	4.65342748325
7	Forrest Gump (1994)	4.63624341759
8	Life Is Beautiful (La Vita è bella) (1997)	4.62483146774
9	Gladiator (2000)	4.62423883137
10	Green Mile	4.61708765855

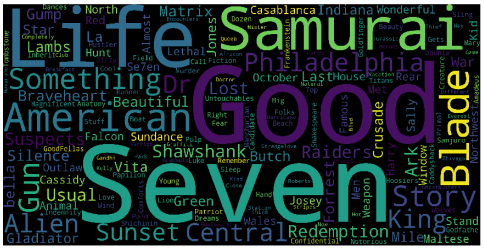
(2) User preference analysis

We can also use predicted ratings to generate a collection of words that have the biggest frequencies in the titles of top 100 movies, which can help us know the preference of a specific user, that is, a movie with what words in the title may be liked by that user.

For example, for the first user in MovieLens dataset, we can obtain two word cloud graphs from our K latent factor model and the deep learning model separately:



Word cloud graph from K latent factor model



Word cloud graph from deep learning model

5. CONCLUSION

We build K-Latent Factor Model (KLF) to solve rating prediction problem on Movielens dataset and run Collaborative Filtering and Deep Learning Model to compare their result with our model. The result shows that KLF has the lowest RMSE and we also prove that Collaborative Filtering model does not fit in with the rating prediction task on the Movielens dataset for different users have different rating taste, take the similarity weighted rating score as predicted result is meaningless and this explains why RMSE on collaborating filtering model is so big.

6. Reference

[1] Sappadla, Prateek, Yash Sadhwani, and Pranit Arora. "Movie Recommender System." Search Engine Architecture, Spring (2017).

[2] Fernández, Guillermo, et al. "Let's go to the cinema! A movie recommender system for ephemeral groups of users." Computing Conference (CLEI), 2014 XL Latin American. IEEE, 2014.

[3] Dataset: https://grouplens.org/datasets/movielens/

[4] Code: https://github.com/LinbinYoung/CSE258-UCSD/tree/master/code/CSE258-FinalProject