CS182: Introduction to Machine Learning
Movie Recommendation System
Final Project Report

Pengyu Ji
2019533156
jipy@shanghaitech.edu.cn

Jiaxuan Ma
2019533143
majx@shanghaitech.edu.cn

Abstract

Our project is an application project focused on movie recommendation system. We collect and process datasets from Movielens. After that, We implement five different methods:(NMF,PMF,CF_KNN,AutoRec,AutoGluon), tune hyperparameters and compare their performance with homogenized and random recommendations as baseline. Finally, we will give some conclusion.

1 Introduction

Recommendation system is applied in many aspects by many mature Internet enterprises. Several traditional collaborative filtering methods such as matrix decomposition and content based methods have been developed in this field. However, those methods are some kind of primitive. We are trying to combine recommendation system with deep learning and ensemble learning to get better performance. The specific member contribution will be in `readme.md` in source code zip.

2 Dataset

Data Collection and Exploration We choose $Movielens\_1M\_Version$ as our dataset. The dataset contains `ratings.csv`, `movies.csv`, `links.csv` and `tags.csv`. We mainly focus on `ratings.csv`. It is a rating table containing 100836 effective records from 610 users and 193609 movies. The average rating given by each user is 3.5, with standard deviation around 1.04. It implies that this is a market that generally gives high evaluation on movies and the evaluation does not fluctuate much.

Data Cleaning and Preprocess As we can see, the fill rate of effective records in origin `ratings.csv` is around $100836/(610 * 193609) = 0.08\%$, which is too small and will lead to a very sparse matrix when implementing further machine learning models. Considering above, we decide to preprocess the data and generate new ratings table to improve the accuracy of our model in future.

1. ratings_mini.csv: In ratings_mini.csv, we only consider records from users whose ID < 30 and movies which has ID < 100. The fill rate now is around $125/(30 * 100) = 4.17\%$.
2. ratings_mini1.csv: In ratings_mini1.csv, we only consider movies which has ID < 100 and the top 30 users who watch most number of these 100 movies. The fill rate now is around $750/(30 * 100) = 25.00\%$.
3. ratings_mini2.csv: In ratings_mini2.csv, we only consider the top 100 movies which are watched by most users and the top 30 users who watch most number of these 100 movies. The fill rate now is around $2500/(30 * 100) = 83.33\%$.

We will focus our work mainly on `ratings_mini1.csv` and `ratings_mini2.csv`.

Preprint. Under review.

3 Methodology

In this project, We totally tried 5 methods, including traditional collaborative filtering methods based on matrix decomposition and more cutting-edged ones, such as AutoRec and AutoGluon, which is recommendation system combined with deep learning and ensemble learning correspondingly. The environment and external library required by the whole project are relatively simple: only numpy, pandas, sklearn, tensorflow, autogluon and nni for hyperparameters tuning are needed.

3.1 Non-negative Matrix Factorization

In NMF, we decompose rating matrix V to WH and construct corresponding optimization problem as below:

$$\begin{aligned} \min_{W,H} \quad & \|V - WH\|_F^2 \\ s.t. \quad & W, H \geq 0 \end{aligned}$$

then we simply use stochastic gradient descent(SGD) to calculate derivatives for W, H and iteratively find the best parameters. While for nonnegative, in every iteration we should set a threshold at 0 to avoid parameters being negative.

3.2 Probabilistic Matrix Factorization

PMF is highly similar to NMF. In PMF, we construct optimization problem as below:

$$\min_{U,V} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - U_i^T V_j)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N}\|U_i\|_F^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M}\|V_j\|_F^2$$

where $I_{ij} = 1$ only if user i has rated movie j.
We also use SGD to fit the model with trainning data. While we will treat λ_U and λ_V as hyperparameters, instead of directly estimate them as in lecture. In fact, when $\lambda_U = \lambda_V$, PMF is the same as NMF.

3.3 Collaborative Filtering_K Nearest Neighbours

This model is a kind of collaborative filtering method based on similarity between users or between items. This contains simple idea that in a relatively large and mature movie market(each user has rated many movies), similar movies will be given close score by similar users. In our model, we can switch between user_based and movie_based, and choose pearson_mode or cosine_mode to measure the similarity. So there are totally 4 types of combinations to generate our model. After we calculate all the similarities, we will find top K most similar item. Then the rating will be predicted by a weighted-average of these K ratings. I will list two measure functions as below:

$$Pearson\_Measure: \quad s_{a,b} = \frac{\sum_{i\in I_{a,b}}(r_{a,i} - \bar{r_a})(r_{b,i} - \bar{r_b})}{\sqrt{\sum_{i\in I_{a,b}}(r_{a,i} - \bar{r_a})^2}\sqrt{\sum_{i\in I_{a,b}}(r_{b,i} - \bar{r_b})^2}}$$

$$Cosine\_Measure: \quad s_{a,b} = \frac{\sum_{u\in U_{a,b}}(r_{u,a}r_{u,b})}{\sqrt{\sum_{u\in U_{a,b}} r_{u,a}^2}\sqrt{\sum_{u\in U_{a,b}} r_{u,b}^2}}$$

3.4 Deep Learning_AutoRec

This model is a combination of recommendation system with deep learning. In this model, I uses $Mr.XiuzeZhou$'s work for reference. It uses an autoencoder to reduce the dimension of raw data. Just like autoencoder which is similar to PCA, AutoRec is also an extraction of information from origin ratings table. It can also be divided into user_based and movie_based, the corresponding objective function is listed as below:

$$user\_based: \quad L = \sum_{i=1}^{m}\|r_{\cdot,i} - h(r_{\cdot,i}, \theta)\|^2 + \frac{\lambda}{2}(\|W\|^2 + \|V\|^2)$$

$$movie\_based: \quad L = \sum_{u=1}^{n} \|r_{u,\cdot} - h(r_{u,\cdot}, \theta)\|^2 + \frac{\lambda}{2}(\|W\|^2 + \|V\|^2)$$

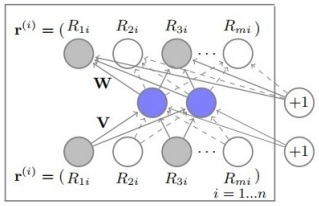
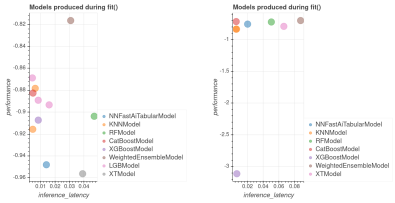


Figure 1: AutoRec Structure

3.5 Ensemble Learning_AutoGluon

AutoGluon enables easy-to-use and easy-to-extend AutoML with a focus on automated stack ensembling, deep learning, and real-world applications such as tabular data. We use tabular prediction here, which predict values in column of data table based on other columns' values. Load training data first, and here we only need to set the label "rating" to the predictor and let it fit on the training data. During fitting, AutoGluon automatically infers performance metric and the type of each feature, trains multiple models which has various hyperparameters and ensembles them together to make prediction.

The figures describe the actual models that were trained during fit and how well each model performed on the held-out validation data.



(a) mini1 (b) mini2

Figure 2: AutoGluon

4 Experiment

4.1 Hyperparameters Tuning

We use NNI(Neural Network Intelligence) to do hyperparameters tuning in an efficient and automatic way. The reference operations are simple as well: update origin code and get hyperparameters set, define search space by json script, launch and config experiment while doing experiments management by opening the Web UI url in browser. It is ease-of-use, scalable, flexible, efficient and also provides powerful visualization.

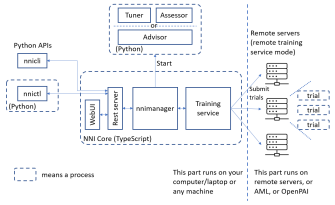
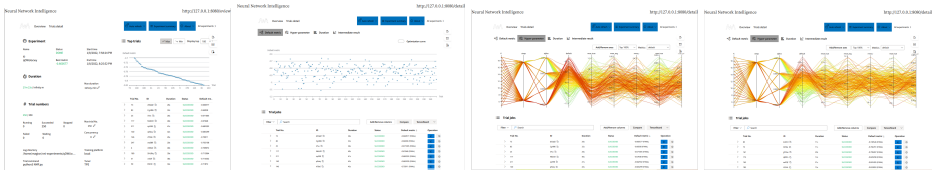


Figure 3: NNI Architecture

Here we will show several graphs for hyperparameters tuning process. We take NMF as an example.

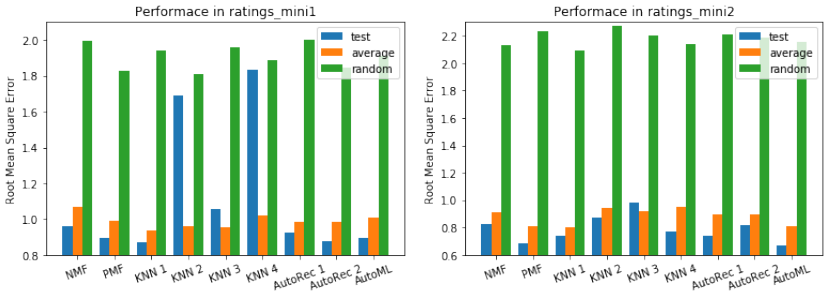


(a) Main Interface (b) Overview on mini2 (c) Overview on mini2 (d) Overview on mini1

Figure 4: Hyperparameters Tuning Process

4.2 Performance Measure

We will use RMSE error to measure the difference between our predictions and true ratings. Moreover, we choose homogenized recommendations(using averaged-trainning-rating as predictor) and random recommendations(using random rating as predictor) as baseline. We will compare our model prediction error with baselines. Here are two preformance graph on `ratings_mini1.csv` and `ratings_mini2.csv` correspondingly.



(a) mini1_performance (b) mini2_performance

Figure 5: Performance

KNN1,KNN2,KNN3,KNN4: user_pearson, movie_pearson, user_cosine, movie_cosine
AutoRec1, AutoRec2: user_based, movie_based

5 Conclusion

1. Among all the performace, PMF, AutoRec, AutoML are always doing a good job, while NMF behaves mediocrely and KNN fluctuates much. This once again shows that Deep Learning and Ensemble Learning have excellent performance under such problems with clear task and framework.
2. In `ratings_mini1.csv`, several KNN methods have poor performance, while in `ratings_mini2.csv`, their performance become better. This is actually reasonable because KNN models make sense on the premise that we are facing a relatively "large and mature market". When there is not enough interactions between users and movies, the similarity between user or between movies can not be extracted correctly.
3. Though we are not possible to find a market like `ratings_mini2.csv` implies, doing some clustering berfore recommendation may be a good idea to generate similar market. This kind of "inactive" recommendation leads to "information cocoons" as well. Even though there is plenty of information, we are still facing something we like but not fresh.
4. In this project, our work mainly focuses on CF methods. In future, we can combine CF methods with Content_Based methods to get better performance when facing brand-new movies. with new requirements on dataset. In addition, trying frameworks such as surprise and cornac which have been developed maturely is an alternative way.

6 Reference

[1] Sedhain et al. (2015) Autorec: Autoencoders meet collaborative filtering.

[2] Salakhutdinov et al. (2008) Probabilistic matrix factorization. NIPS: 1257-1264.

[3] Lee et al. (1999) Learning the parts of objects by non-negative matrix factorization. Nature: 788.

[4] Nick Erickson et al. (2020) AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data.

[5] Neural Network Intelligence. (2020) `https://nni.readthedocs.io/zh/latest/`