Movie Rating Prediction based on Netflix Prize Data Midterm Report

P01-Zhangqi Zha zzha@ncsu.edu
Gang Zhang gzhang22@ncsu.edu
Junhua Ma Jma20@ncsu.edu

Abstract

In this course project report, we present a few prediction models on the famous Netflix Prize Dataset. Based on our implementations, we compare three techniques of machine learning: k-nearest neighbor classification (KNN), Singular Value Decomposition(SVD) and Artificial Neural Networks(ANN). We also describe a few data sampling techniques that we applied in our work. Our results show SVD method outperforming the other two methods in this small study.

Keyword: Netflix Prize; KNN; SVD; ANN.

1 Background

In 2006 Netflix lunched the contest to find a more accurate movie recommendation system to replace their old system, Cinematch. They proposed a one million dollars prize to anyone who could improve over the Cinematch system by at least 10% Root Mean Squared error (RSME). After three years of the contest, in 2009 the grand prize was awarded to team "Bellkor's Pragmatic Chaos". The movie recommendation system is a system that can recommend some movies to the user based on their previous movie preferences. The participants were provided with a Dataset consisting of 100,480,507 ratings that 480,189 users gave to 17,770 movies.

2 Introduction

2.1 Objective

Predict the rating a user will give a movie based on the movies that user has rated, as well as the ratings similar users have given to similar movies.

2.2 Description of the Dataset

The dataset consisting of 100,480,507 ratings that 480,189 users gave to 17,770 movies. The training dataset was splited to 4 files. In side each file, movie id line is stop by ":", each subsequent line after the movie id line corresponds to a rating from a customer, as shown below:
movie id:
CustomerID,Rating,Date
- MovieIDs range from 1 to 17770 sequentially.
- CustomerIDs range from 1 to 2649429, with gaps. There are 480189 users.
- Ratings are on a five star (integral) scale from 1 to 5.
- Dates have the format YYYY-MM-DD.

2.3 Related Works

During the contest and the final grand prize, there have been a lot of data mining and machine learning techniques went into the winning solutions. We reviewed these related works in the following aspects.

2.3.1 Normalization of Global Effects

Normalization of Global Effects captures the baseline rating (usually the mean over all the ratings) with two biases: user specific effect and movie specific effect.

2.3.2 Neighborhood Models

The standard approach is to take some similarity metric (e.g., correlation or a Jaccard index) to define similarities between pairs of movies, take the K most similar movies under this metric (where K is perhaps chosen via cross-validation), and then use the same similarity metric when computing the weighted mean[1].

2.3.3 Matrix Factorization

Compare to the neighborhood approach, matrix factorization approach is a more global view that decomposes users and movies into a set of latent factors, whereas the neighborhood approach takes a very local approach to ratings. The typical way to perform matrix factorizations is to perform a singular value decomposition on the ratings matrix (using stochastic gradient descent and regularizing the weights of the factors)[1].

2.3.4 Restricted Boltzmann Machines

Restricted Boltzmann Machines provide another kind of latent factor approach that can be used.

2.3.5 Regularization

Regularization was used to prevent the models overfitting on the dataset. Ridge regression was heavily used in the factorization models to penalize large weights, and lasso regression (though less effective) was useful as well. Many other parameters (e.g., the baseline predictors, similarity weights and interpolation weights in the neighborhood models) were also estimated using fairly standard shrinkage techniques[1].

2.3.6 Ensemble Methods

Ensemble methods make the different algorithms mentioned above can be combined to provide a single rating that exploits the strengths of each model. In the paper detailing their final solution, the winners describe using gradient boosted decision trees to combine over 500 models; previous solutions used instead a linear regression to combine the predictors[1].

2.4 Our Solutions

In this course project, we tried three types of methods: Neighborhood based method, matrix factorization based method and neural network based method. We will evaluate the performance of each method on rate predicting, and compare their accuracy and efficiency.

3 Method

3.1 Sampling

Sampling is concerned with the selection of a subset of individuals from within a statistical population to estimate characteristics of the whole population. Two advantages of sampling are that the cost is lower and data collection is faster than measuring the entire population. In our case, because of the large amount of data and related small memory laptop, sampling from the dataset will be a practical implementation. Simple random sampling and stratified sampling are two major sampling methods. A stratified sampling approach is most effective when three conditions are met:

- Variability within strata are minimized.
- Variability between strata are maximized.
- The variables upon which the population is stratified are strongly correlated with the desired dependent variable[5].

3.2 KNN

The k-nearest neighbors algorithm (KNN) is a non-parametric method used for classification and regression. With KNN, given a point (u, m) to predict rating, we compute the k most similar points and average the ratings of those points somehow to obtain our predicted rating r. Naturally, different spaces, similarity metrics and different averaging techniques would affect the performance of KNN. The training dataset provided by Netflix was terribly huge which consist of one hundred million ratings. If we were to run the algorithms and waiting for the method to train, we would definitely waste a large amount of time. Therefore we extracted smaller dataset from the original one. Pearson's correlation coefficient is the covariance of the two variables divided by the product of their standard deviations. The form of the definition involves a "product moment", that is, the mean (the first moment about the origin) of the product of the mean-adjusted random variables; hence the modifier product-moment in the name.
consider a given user u_i rates a movie with a distribution $R_i \approx (u_i, \sigma_i)$, a similarity metric between users u_i and u_j is the correlation coefficient between the two distributions R_i and R_j:

$$\rho_{ij} = \frac{E[(R_i - u_i)(R_j - u_j)]}{\sigma_i * \sigma_j} \tag{1}$$

We estimate the covariance and variances by considering the M movies user i and j have in common and

$$E[(R_i - u_i)(R_j - u_j)] \approx \frac{1}{M}\Sigma(r_{ik} - u_i)(r_{jk} - u_j) \tag{2}$$

$$\sigma_i \approx \sqrt{\frac{1}{M}\Sigma(r_{ik} - u_i)^2} \tag{3}$$

$$\sigma_j \approx \sqrt{\frac{1}{M}\Sigma(r_{jk} - u_j)^2} \tag{4}$$

The value of the Pearson Correlation Coefficient lie in the interval [-1,1]. At knn the values of this similarity function lie in the [0,1] interval. And it's easy to convert from the first interval to the second is to $(\rho_{i,j}$+1)/2

3.3 SVD

Matrix factorizations or matrix decompositions are powerful tools in the recommender systems. The two broad uses of Matrix factorizations are Alternating Least Squares (ALS) and Singular Value Decomposition (SVD). The first one is more or less a generic approach which can be combined with many factorizations and the second is one specific factorization by itself.
Singular Value Decomposition or SVD approximates a single matrix A by the product of three matrices

$$A = U\mathbf{\Sigma}V^* \tag{5}$$

where $\mathbf{U}$ is an $m \times m$ real or complex unitary matrix, $\mathbf{\Sigma}$ is a $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal, and $\mathbf{V}$ is an $n \times n$ real or complex unitary matrix. The diagonal entries σ_i of $\mathbf{\Sigma}$ are known as the singular values of $\mathbf{A}$. The columns of $\mathbf{U}$ and the columns of $\mathbf{V}$ are called the left-singular vectors and right-singular vectors of M, respectively.

Here in the recommender system, the so called user-item (user-movie in the netflix dataset) interactions are modeled as inner products in the latent factor space with dimensionality. Each item is associated with a vector $q_i \in R^f$. Each user is associated with a vector $p_\mu \in R^f$ and the elements of the vector p_μ involve characterizations of the level of interest that the user has in items that highly correspond, whether this interest is positive or negative[1]. Thus the inner product of the two vectors approximates the rating of the users rating of any specific item i, denoted as $r_u i$, in the estimate[2]

$$r_{ui} \approx q_i^* p_u \tag{6}$$

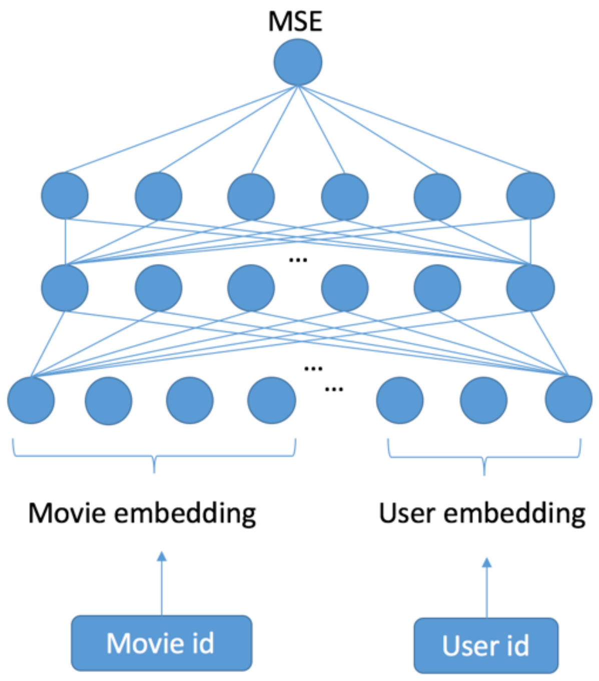
Once the system maps the vectors that describe the user-item interaction, it is a relatively simple task for it to then use the above equation to estimate the rating a user would give to a given item.
It is the case that two items of the same rating are not equal in quality. One might be a film that exhibits higher quality either in its general regard by critics, the cost and effort that goes into creating the film, etc. Thus ratings alone cannot determine what movies should be recommended and that is when biases are added to the equation. So the first-order approximation of the expected rating for an item i by user u would be[2]

$$r_{ui} = \mu + b_i + b_\mu + q_i^* p_u \tag{7}$$

3.4 ANN

Although our task is a regression task and Neural Networks are mainly dealing with classification problem, we still want to test the performance of Neural Networks on Netflix data. Our Neural Network approach first simply applied a full-connected Neural Network. However, unlike Softmax output layer in classification problem, we sum over all weighted values in last layer and use it as our regression output. Then use minimal square error as our lose function. In other words, we built a Neural Networks with a linear regression as its output, and the properties linear regression used are generated from neural network. With gradient decent, we can not only upgrade the weight of linear regression, but also upgrade the weights in NN in back propagation which help to generate better properties. the structure of Neural Network is like pic1:

In the training process, we trained movies embedding and users embedding to get better rating prediction. And After training, we can also use these embedding for clustering. This idea is pretty like word embedding generated in language model training process. As a result, similar movies and users will have similar embedding, which make them close to each other in embedding space. Thus we can simply use k-means to clustering these embedding. Also, for better visualization, we also plan to use t-SNE algorithm to generate scatter plot in 2D space for visulize the result.



4 Experiment & Results

4.1 Sampling

We have tried simple sampling and stratified sampling methods on the dataset. Here are some results of different sampling methods and spaces based on SVD algorithm.

SVD Algorithm Results	RMSE
Simple Sampling 0.1% of all data	1.0370
Simple Sampling 1% of all data	1.0085
Simple Sampling 10% of all data	0.9327
Stratified Sampling user id less than 80000 and movie id less than 5000	0.9574

4.2 KNN

Baseline Measure:
if the user distributions are independent, then the best we can do is to consider every user individually and estimate the rating as the mean of all prior ratings of a particular user.

	RMSE	MAE
Fold 1	0.9848	0.7612
Fold 2	0.9941	0.7726
Fold 3	0.9998	0.7750
Fold 4	0.9960	0.7750
Fold 5	0.9930	0.7697

4.3 SVD

To be completed... As the data points given increasing, the RMSE was decreasing, but the computation time was raising drastically.

4.4 ANN

For the parameter setting for NN approach, we select leaning rate among {0.1, 0.01, 0.001}, the batch size B among {100, 300, 1000}, the activation function among{'sigmoid', 'relu'}. We define the best configuration according to MSE in validation data. In our experiment , the best configurations are... (need to be completed) We select a sampled subset of training data for our NN because without enough computing resource, NN will take months to train to whole data or even 10% of it. However, with small sampled dataset, we found NN is so easily get overfit and performed bad on testing data. After some research we applied Dropout to every layer of our NN, and solved overfitting problem.

5 Discussion

References

[1] E.Chen, http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/

[2] S. Gower, Netflix Prize and SVD. pages 1–10, 2014.

[3] Y. Koren, The BellKor Solution to the Netflix Grand Prize, 2009.

[4] Zheng, Yin, et al. A neural autoregressive approach to collaborative filtering.

[5] https://en.wikipedia.org/wiki/Sampling$_{(}statistics)$.

Appendix

Teammate and Work Division
As earlier planned, we explored the dataset together and come up with the sampling method. Zhangqi worked on data format transform and sampling process, SVD algorithm and report preparation. Gang and Junhua foucused on KNN and ANN algorithms respectively.