COMP4434 Project Mid-report

Xiaochuan Xu
17082834d@connect.polyu.hk

The HK Polytechnic University — May 5, 2021

Introduction

Presentation Link: https://connectpolyu-my.sharepoint.com/:v:/g/personal/17082834d_connect_polyu_hk/EQ_3VxanLoBNq86mMo-cMKYBzWAnWCVm6y4uwfJwyLgAaQ?e=bbfknq

Rating prediction is an important feature that PolyTube should have. Since the rating is a number out of 10, this problem can be viewed as a regression problem. In order to treat this regression problem, several methods can be done such as simple multi-variate linear regression, decision tree method, support vector machine, and neural networks. Each of the method may require different treatment of feature engineering. To start, this project would adopt a linear regression method, with other more advanced methods presented later. This report will start with the feature engineering work in preparation for linear regression. Then we construct the dataset into a train set and a test set. Then we will implement the linear regression model by using gradient descend method. An analysis of the result will be added afterwords. In the end, future work about using other methods will be discussed.

1 Data Preprocessing

This part introduces some pre-processing techinics we use to pre-process the data. I will explain the pre-processing procedures for each of data columns.

1.1 Features

(1) teleplayid and name
They are used to identify a particular data row, which take no influence in our model. I simply drop the two columns.

(2) type
Though in the documentation of the project, a record has type either "short", "medium", or "long". My investigation of the original data shows that there are in total six kinds, which are "short", "medium", "long", "Music", "Special", and "ONA". Moreover, there are data which do not possess a type.

Obviously, they are categorical data. To handle this, I introduce the **dummy encoding method.** That is, I add six more columns named as the corresponding labels. If this record originally has a type of any kind of the above, the corresponding dummy variable will be marked as 1, and the rest of them will be 0. In this way, we transform the type column into six parallel columns. For a record which is missing any type, all of the dummy variables will be 0.

(3) episodes
This is a numerical-type data, which possess direct realistic meaning. There are two options of the handling of this column. Either we treat it as a normal numerical data, or we transform it into a categorical data based on an interval method. The justification of transforming it into a categorical data includes two points.

(a) Episodes are not strictly continuous. Say, there are some numbers which rarely appear because it does not make much sense of having such a number of episodes.

(b) Different number of episodes can be mapped into a realistic setting in daily life. For example, a less-than-3 episodes of a series may be regarded as a short series. A more-than 24 episodes series may be seen as long series. The transformation can entitle this column with more realistic meaning, masking the unimportant details of the number.

Other than that, handling of the missing data is still needed. There are some records which show an unknown label in episodes. In this case, I simply transform the label into 0.

(4) rating
This corresponds to y in the final model. However, there are some rows having this attribute missing. I drop them because this column is critical for testing.

(5) members
This is a continuous numerical data. I drop the row if this value is missing.

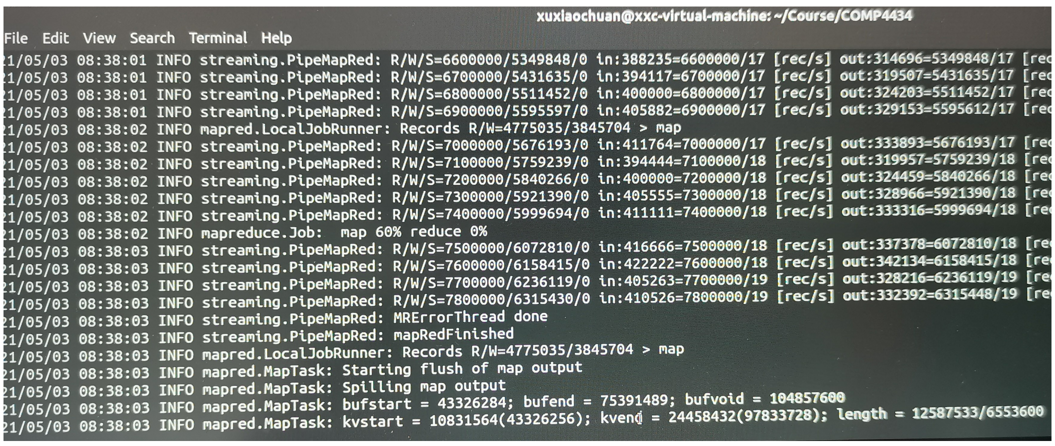
(6) genre
This an important categorical data. A record may possess one or more genres. To handle this, I also adopt the dummy encoding approach. A list of columns are added, which are corresponding to the genres. If some row has genre to be empty, then all of the corresponding dummy variables are set to be 0.

1.2 Incorporate a new feature

Based on the description of the materials, we found the latest ratings for some teleplays in Rating.csv. This can be seen as a valuable feature, which can be attached to the original teleplays.csv as a new column. There are two challenges while implementing this idea.

1. Rating.csv is huge in terms of its size. Traditional methods using Python to calculate the average ratings for teleplays would be too slow. Here, we integrated the **Map and Reduce** approach learnt in the previous lectures to compute the necessary values. I used the docker container, together with the map.py and reduce.py I wrote, to run Hadoop to calculate the average ratings for each teleplay. And finally I got a new txt file, which has the records of the average ratings of each teleplay. The following figure is the screenshot while running the map reduce using Hadoop.

2. However, there are still missing ratings for some teleplayid. To handle this, I use the total average ratings across all teleplays as the replacement of the missing values. This is because latest ratings are precious valuable data, and there is some common trend inside the whole rating data. Therefore, it would be more natural to replace the missing value with the average value instead of using 0.


xuxiaochuan@xxc-virtual-machine: ~/Course/COMP4434
File Edit View Search Terminal Help

```
21/05/03 08:38:01 INFO streaming.PipeMapRed: R/W/S=6600000/5349848/0 in:388235=6600000/17 [rec/s] out:314696=5349848/17 [rec
21/05/03 08:38:01 INFO streaming.PipeMapRed: R/W/S=6700000/5431635/0 in:394117=6700000/17 [rec/s] out:319507=5431635/17 [rec
21/05/03 08:38:01 INFO streaming.PipeMapRed: R/W/S=6800000/5511452/0 in:400000=6800000/17 [rec/s] out:324203=5511452/17 [rec
21/05/03 08:38:01 INFO streaming.PipeMapRed: R/W/S=6900000/5595597/0 in:405882=6900000/17 [rec/s] out:329153=5595612/17 [rec
21/05/03 08:38:02 INFO mapred.LocalJobRunner: Records R/W=4775035/3845704 > map
21/05/03 08:38:02 INFO streaming.PipeMapRed: R/W/S=7000000/5676193/0 in:411764=7000000/17 [rec/s] out:333893=5676193/17 [rec
21/05/03 08:38:02 INFO streaming.PipeMapRed: R/W/S=7100000/5759239/0 in:394444=7100000/18 [rec/s] out:319957=5759239/18 [rec
21/05/03 08:38:02 INFO streaming.PipeMapRed: R/W/S=7200000/5840266/0 in:400000=7200000/18 [rec/s] out:324459=5840266/18 [rec
21/05/03 08:38:02 INFO streaming.PipeMapRed: R/W/S=7300000/5921390/0 in:405555=7300000/18 [rec/s] out:328966=5921390/18 [rec
21/05/03 08:38:02 INFO streaming.PipeMapRed: R/W/S=7400000/5999694/0 in:411111=7400000/18 [rec/s] out:333316=5999694/18 [rec
21/05/03 08:38:02 INFO mapreduce.Job:  map 60% reduce 0%
21/05/03 08:38:03 INFO streaming.PipeMapRed: R/W/S=7500000/6072810/0 in:416666=7500000/18 [rec/s] out:337378=6072810/18 [rec
21/05/03 08:38:03 INFO streaming.PipeMapRed: R/W/S=7600000/6158415/0 in:422222=7600000/18 [rec/s] out:342134=6158415/18 [rec
21/05/03 08:38:03 INFO streaming.PipeMapRed: R/W/S=7700000/6236119/0 in:405263=7700000/19 [rec/s] out:328216=6236119/19 [rec
21/05/03 08:38:03 INFO streaming.PipeMapRed: R/W/S=7800000/6315430/0 in:410526=7800000/19 [rec/s] out:332392=6315448/19 [rec
21/05/03 08:38:03 INFO streaming.PipeMapRed: MRErrorThread done
21/05/03 08:38:03 INFO streaming.PipeMapRed: mapRedFinished
21/05/03 08:38:03 INFO mapred.LocalJobRunner: Records R/W=4775035/3845704 > map
21/05/03 08:38:03 INFO mapred.MapTask: Starting flush of map output
21/05/03 08:38:03 INFO mapred.MapTask: Spilling map output
21/05/03 08:38:03 INFO mapred.MapTask: bufstart = 43326284; bufend = 75391489; bufvoid = 104857600
21/05/03 08:38:03 INFO mapred.MapTask: kvstart = 10831564(43326256); kvend = 24458432(97833728); length = 12587533/6553600
```

After identifying and countering these two challenges, a new feature called "latest rating" is successfully incorporated into the original teleplay dataset. The map and reduce programs are attached in the appendix.

1.3 Normalization

It can be observed that other than episodes and members, the other columns possess a value of 0 or 1.Even though for episodes and members, they differ hugely in the value. Episode ranges from 1 to 100, whereas members have a maximum value over 1 million. To facilitate the easy-training as well as to ensure the performance of the trained model, normalization is necessary in this case. Therefore, we adopt sklearn's *MinMaxScaler* method to normalize the training set.

1.4 Processed Dataset

To implement the above pre-processing, here I mainly used pandas and its functions to drop, add, replace, and merge columns and rows. The dataset processed possess 10011 records and 53 columns.

2 Model Design and implementation

2.1 Train set and Test set

Splitting the dataset into two sets which are used for training and testing is a common approach. Here I used sklearn *train_test_split* method to split the dataset into 4-1 parts.

2.2 Multi-variate Linear Regression

To apply the knowledge learnt in the lecture, I manually re-implemented the linear regression model using gradient descend based on the lab materials. I also intentionally take note of the validation error and training error during the phase of fit. So the change of RMSE of training and testing set can be monitored. As for the training parameters like learning rate and iteration number, I set them to 0.01 and 20000 iterations respectively after comparing the performance and the changing curves of RMSE.

2.3 Neural Networks

For neural networks, I utilized one of the most common neural networks that is used for doing regression, namely, Multi-Layer Perception (MLP) networks. There are a few things to think through before finalizing a concrete and useful model.

1. *Number of hidden layers*

   The number of hidden layers are directly related to the accuracy of training and testing results. Considering the number of input features is about 50, and the total number of training data is about 10000, the number of hidden layers should not be too many so that vanishing gradients can happen, or overfit will be more easily to happen. Therefore, we choose 6 layers as the number of hidden layers.

2. *Activation functions*

   Popular activation functions include ReLu, logsitic, tanh etc. Due to lacking relative experiences in using these different activation functions, I adopted a trial-and-error approach. I used different methods in the training stage, and compared their performance. At last, I chose ReLu as the activation function since it attains great performance in both training and testing stages.

3. *Solver*

   In previous lectures, SGD is our common approach to train the networks. However, adam is another powerful solver that is believed to inherit the power of SGD while overcome some other problems. Therefore, adam is selected as the solver in my model.

Of course, the actual consideration for parameters are not limited to these points only. For instance, learning rate and number of iterations are also two important decisions to be made in the actual training process of the model. However, it is noted that these parameters are comparatively not designed in the beginning. Instead, they are decided with the actual training results. Whether the model is overfit or underfit, how long it takes to converge, these are both the practical consideration to decide these parameters.

As a result, this is the final parameters used in the model.

```
number_hidden_layer = 6
activation_fun = relu
solver = adam
learning_rate = 0.001
number_iteration = 300
l2_penalty_coefficient = 0.0001
```

2.4 Content-based Recommendation System

The main idea of content-based recommendation is simple, which is to find the similar teleplays across all different teleplays. Then we find the favourite teleplay of someone, and recommend him with the most similar teleplays of that teleplay.

Hence, the first step is to construct a similar matrix of the teleplays. I adopted the approach of the lab 9, which used a cosine similarity function. However, the core of this problem is to define similarity. In the lab, movie tags are used to attached the original movie dataset to represent its features. But in our project, there are no tags. All we have is the genres, type, episodes, members and ratings. Therefore, I am thinking how to express the features of the movies.

Apparently, genres and types of the movies are an obvious characteristic which can well expressed the feature of movies. As for episodes, my experience told me that we can also categorize different movies based on its number of episodes. For instance, for number of episodes exceeding 21, it is often regarded as a "long series"; for number of episodes less than 5, it is considered as a "short series". In this way, we can further break down the continuous attribute of the movies like members and ratings into categorical data. The summary is shown as follows.

1. *Episodes*

   Long series: > 21; Middle series: > 5 and $<= 21$; Short series: $<= 5$

2. *Members*

   Big crowd: > 100000; Middle crowd: > 1000 and $<= 100000$; Little crowd: $<= 1000$;

3. *Ratings*

   Good movie: > 8 Normal movie: > 6 and $<= 8$ Bad movie: $<= 6$

In this way, we get a series of features of the movies. Then I aggregate all of these features into on string, just as the lab did, and used this string to represent its similarity features. After getting a similarity table, we can easily find the most similar movies with the favourite movies of that particular user.

2.5 Collaborative Recommendation System

Different from content-based recommendation system, collaborative recommendation system took the advantage of using others' data, which in turn can automatically recommend teleplay to users that they haven't watched yet.

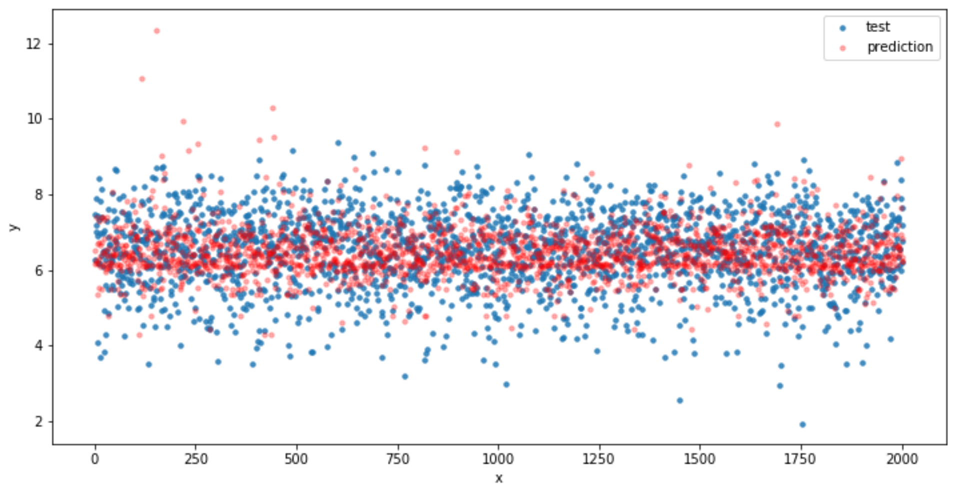
Hence, in this question, different users' ratings on different teleplays become paramount, which means Rating.csv is the main source of this problem. Based on the materials of lab 9, we can easily make a data pivot table, for which we use to train the model in SGD.

However, one practical problem is, rating.csv is too large, which possesses 7873120 rows. My computer cannot execute pivot function on it because of the limited memory. Therefore, I have to scale down the size of data. I firstly eliminate those teleplays which are watched by user 53698. This scales the number of rows down to aroun 3 million. However, my computer is still not capable of manipulating such large data. I have to scale-down the user number, from around 90000 users to 10000 users, which includes user 53698. This scales down the number of users to around half million, and my computer is able to process and train these data. This can be seen as a potential weakness of my model, since it doesn't make the most of all the data.

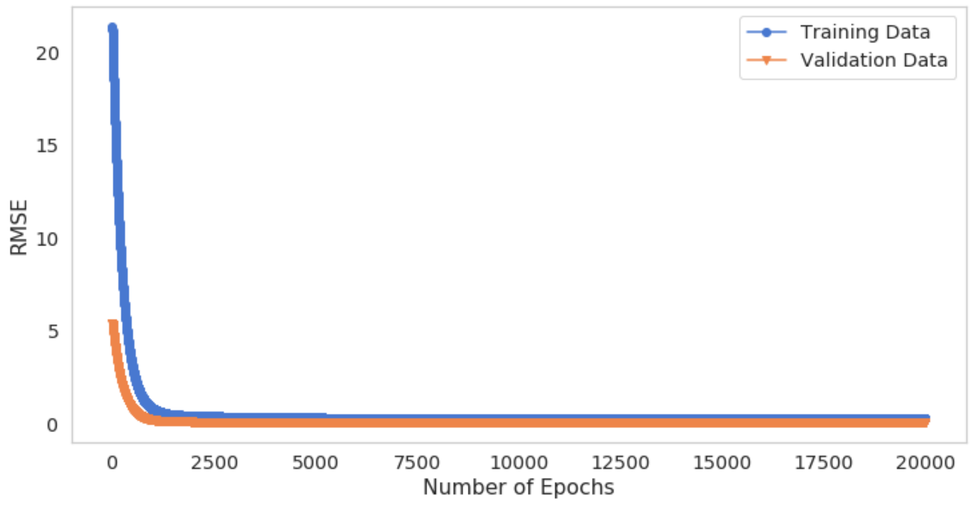
3 Model evaluation and discussion

3.1 Multi-variate Linear Regression

The results calculated by the model are fairly good. I used sklearn mean square error function to calculate the RMSE as well as the R2 score between the prediction y and the test set. Root mean squared error of train set is 0.5133. Root mean squared error of test set is 0.4940. It is discovered that the R2 score is low, which means the linear relation of the variables may be weak. This hints us to adopt other methodology into this problem. The results are shown in the following figure.



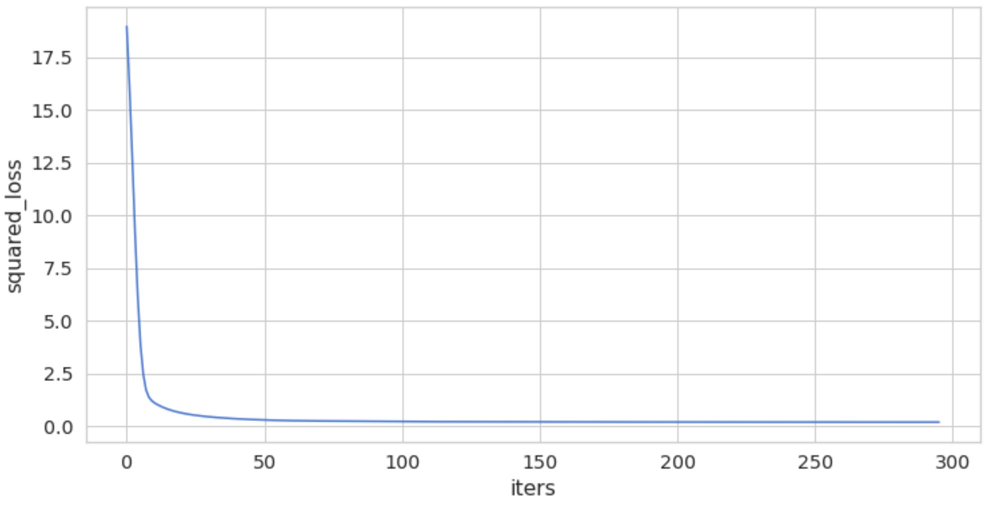
The fitting curves for both traning and testing data are shown below.



3.2 Neural Networks

Neural networks attains slightly better results after tuning the parameters. Specifically, RMSE for training and testing data are 0.3871 and 0.4023 respectively. R2 scores for training and testing dataset are 0.636 and 0.604 respectively. This improves R2 scores as well, which implies the relevancy between models and actual results.

The curves of rmse changing during training is shown below.

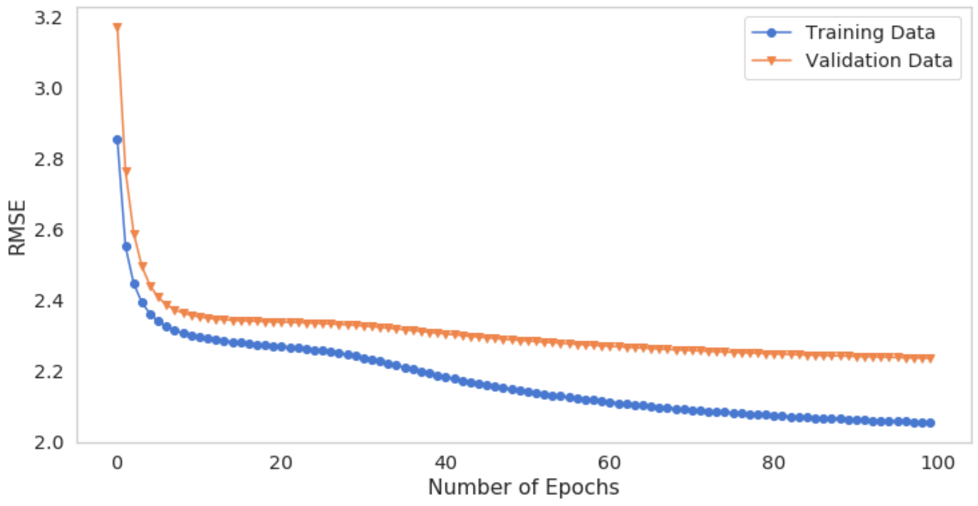


3.3 Content-based Recommendation System

In the implementation process, I firstly identified the favourite movies of user 53698 (rated by him/her as 10). Then I used the similar matrix to find the most similar movies with these favourite movies. The final results are attached in the submitted files.

3.4 Collaborative Recommendation System

After 100 iterations, the loss tends to be stable, which is shown below. Then, I used the trained model to predict the ratings of those teleplays which have not been watched by this user. The results are attached in the submitted files.



4 Summary and Future work

Though I have to conclude here for this report, ways of machine learning are beyond imagination, especially currently we got tons of models to use and even more parameters to tune. However, the completion of this project did render me with valuable and hands-on experiences in designing, implementing and tuning the models. Moreover, the experiences of using Map and Reduce is another highlight of this project.

To conclude, this project did the following things successfully.

1. Use Map Reduce with Hadoop to do data-preprocessing successfully.
2. Use both linear and neural networks method to successfully predict the ratings for the new teleplays.
3. Use both content-based and collborative recommendation method to successfully recommend movies to a certain user.

In the future, more efforts can be spent in designing more appropriate neural networks. Tensorflow and PyTorch can be used to implement those neural networks, for this framework can not only customize the each hidden layers as well as to utilize GPU to accelerate the training.