Taobao Recommendation System

XIE Yina
yxieat@connect.ust.hk

WANG Ruolan
rwangbh@connect.ust.hk

WANG Zhicong
zwangee@connect.ust.hk

WANG Zhe
zwangec@connect.ust.hk

ZHENG Xiaowen
xzhengao@connect.ust.hk

1 Introduction

The rise of e-commerce sites and the formation of users online shopping habits have brought a huge amount of online consumers behavior data. By mining users behavior data, a lot of interesting patterns can be retrieved and the conversion rate of e-commerce can be improved with the intelligent recommendation system that have better knowledge of users. To improve the online shopping experience, effectively predicting users final consumption choice from the behavior logs is of great importance. In this project, we aim to predict users final consumption choices based on the past behavior of the users.

2 Data Description

Tianchi provided the behavior data of 20,000 users and the information of millions of items.The data contains two parts.

- **User Behavior Data** The first part is the dataset with the mobile behavior data of users in the set of all items.
- **Items Data** The second part is the dataset with the category and geographic information of items.

The training data contains the user behavior data of certain quantity of sampled users in one month (2017.11.18 2017.12.18). The evaluation data is the purchase data of these same users of the items in the next day (2017.12.19). We need to develop a recommendation model to predict the purchase behavior of the users of the items in the next day.

3 Feature Engineering

3.1 Data Preprocessing

Remove outliers (double 12) As Figure 1 shows, the sales volume of double 12 is very different from that of normal days. Adding them to the training dataset will affect our forecast, so we delete the outliers.

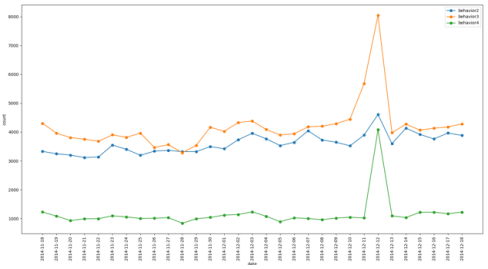

Figure 1: Amount of data under different dates

Down sampling Because the data comes from real sales scenarios, the problem of data imbalance is very serious, which is up to 99:1. We used the method of down-sampling to solve this problem.

Hot Items After analyzing, we found that more than eighty percent people have interactions with top three thousand hot items every day, Which means most buying behavior occurred in the top three thousand hot items. If we recommend them accurately, our final result would have a high accuracy.

3.2 Feature Construction

We designed feature engineering for two models respectively since time series model (TSM) and non-time series model (Non-TSM) have different characteristics. TSM requires data maintaining consistency in time dimension, and then it will discover periodicity of data in time dimension based on time consistency. Non-TSM has no information about time dimension, so we need to use time window artificially to compensate for the lack of time dimension.

We combined feature types, feature objects and Collaborative filtering to generate 103 new attributes. Then, by drawing a heat map, we removed the attributes with low correlation and generated the final new attributes.

Feature type When building features, we find that there are two kinds of features. One is global features, which do not change with time, such as user preferences, long-term conversion ratio, etc. The other is the current features, which will change with time, such as the hot 3,000 items of today, short-term conversion ratio and so on.

Feature object In this project, there are seven types of objects to construct feature engineering: users, products, categories, regions, users-products, users-categories, users-regions.

- **Users** Each user will have some attributes, such as whether he is an active user, what kind of user he belongs to, and which users he is similar to?(from collaborative filtering) And also each user will have his own purchase preferences, such as in which area he prefers to buy goods, and which category of goods.
- **Items** For each item, we judge whether it belongs to the popular 3,000 commodities, and make statistics on its total click ratio and purchase ratio (global current).
- **Categories** For the category of products, we also calculated the click ratio and purchase ratio (global current). At the same time, we judged whether the category is a sustainable development category by the increase of the purchase ratio.
- **Regions** Click ratio and purchase ratio (global current).
- **users-products** Click ratio and purchase ratio (global current). Judging whether users and products are in the same region, which is a useful for users who have regional preferences.
- **users-categories / users-regions** Click ratio and purchase ratio (global current).

Collaborative filtering In this project, only items have their own category attributes, so we try to improve the accuracy of our recommendation by using collaborative filtering to classify users. We have 20,000 users and millions of products. If we use all the information to construct a matrix of user-products, the matrix will be very large and very sparse. So we only chose products that have been purchased more than 1000 times and users who have purchased products more than five times as elements of the matrix.

- **Get matrix** In this m*n matrix (m is the number of users and n is the number of products), The behaviors of viewing, collecting, adding shopping carts and being purchased are marked as 1, 2, 3, 4, and the rest are marked as 0.
- **Clustering** We used distance.pdist() to calculate the distance between data one by one, and then we used linkage function (in hierarchical clustering) to get the matrix,we chose the ward as our method to generate this matrix based on the cluster result (ward balance the sample), finally used fcluster() to generate the subcluster. We divided users into 100 categories.

Based on the 103 new features generated from the combination of feature types, feature objects and Collaborative filtering, we used the heat map below, which can reflect the correlation between features and classification results, to filter features and got the final list of 83 features.

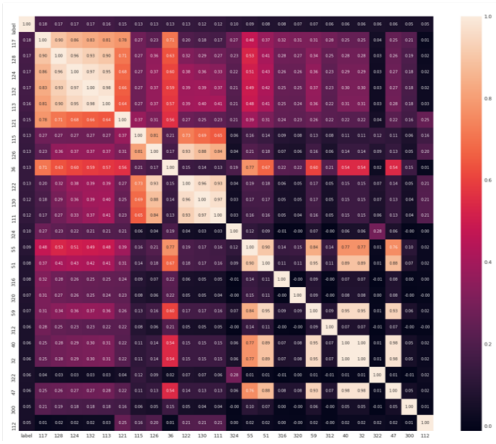

Figure 2: Features correlation heatmap

For Non-TSM model, we use the technique of time sliding window to capture the time series inforamtion.

Time Sliding Window For Non-TSM, besides using the features in TSM, we also designed a time window to solve the problem of time dimension. Sliding window is mainly used for capturing time series information. Actually, user behavior type in different days is very correlated. For example, if the user has a lot of interactions with one item in the past days, from clicking to collecting. It will be likely that the user will buy it in the next following days. Thus, sliding window is very important for building recommendation system. The sliding window can be represented by the following picture.The sliding window can be represented as the picture below.

In our project, we set the sliding window of seven, five, three, two days. It means when we constructing feature of the t-th day, we will use the information of past seven, five, three and two days of this user. When the day is closer to the predicting day, the information will be much more useful. So, the sliding window interval becomes smaller and smaller. On the country, the information which is very far from the predicting day will have little influence for the prediction, so we set seven as upper bound to collect information.

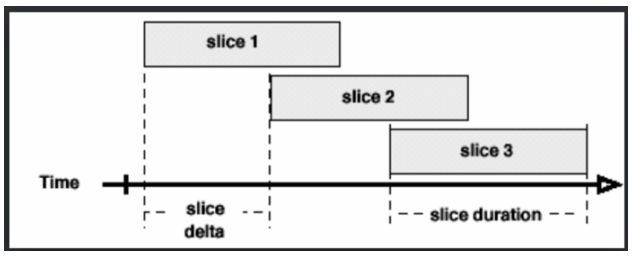

Figure 3: Illustration of sliding window

3.3 Label Construction

Label construction for Non-TSM We use the first days user-item pair as index and the next days behavior type information to construct label. If the user bought the item in the next day, we will label the user-item pair as 1. If not, we will label it as 0.

Label construction for TSM We take out the users and items which have appeared in the first day and then construct num(uers)*num(items) records. If the user bought the item in the next day, we will label the record as 1. If not, we will label it as 0.

4 Models

There are some models we learned this semester, which will be tested on the dataset later. We will make use of the python packages sklearn and tensorflow to implement those models.

4.1 Evaluation Method

The performance of our model is evaluated by F1 score, which is an classical method to consider both Precision and Recall.

- **Precision** The proportion of users who are genuinely interested in recommending information from users who receive the push information.
- **Recall** The proportion of users who receive recommendation information from all potential users.

F1 score is the harmonic mean of precision and recall. In the same time we increase the prediction precision, we don't want lose customers. We can push the recommendation to 100 users just to find 10 customers. So in this situation, the recall score is also important. So our purpose in this project is to increase the prediction results' F1 score as much as possible.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

4.2 Time series model

The data contains the time series information, which lead us to have a try on RNN models, like LSTM.

4.2.1 Introduction of RNNs

The RNNs are popular neural networks these years, mainly because normal neural networks could not handle the time series problems, which means they cannot make use of the previous step information.

4.2.2 Advantages of LSTM

For LSTM[1], it has certain improvement on the traditional RNNs, which take long-term memory into consideration. This kind of network can have better performance on Natural Language Processing which need long term memory to conquer the nearby redundant words. This kind of effect may also apply on the recommendation system. The next behavior of the user may not only be influenced by the current behavior but also by the actions long time ago.

4.2.3 Data set

The data set contains 1000 different user-item pairs and their action of 10 days. The attributes here only contain the data for each time step.

4.2.4 Experiments on different settings

- **influence of time length** The time length determines how many information in all the network can receive and process. The longer time length will bring more information but the GPU and time consumption will increases.
- **influence of number of stacked LSTM cells** The stacked LSTM is like the multilayer perceptron or multilayer CNNs. This will determine how complex the model is. The more complex model will perform well on sophisticated problems but will over-fit on simple problems.
- **influence of sequence or non-sequence return** LSTM provides two kinds of return. One returns the states of all the times and the other only returns the last state. For this data set, the mode with multiple returns have better performance. One possible reason for this is that it has more chances to train the last FC layer and the small size of the data set is also another possible reason.

Experiment results According to Table 1, larger time step will bring more information but may also increase the noise, so an appropriate length of five performs the best. Also more or less stacked LSTM cells will lead to underfit or overfit.

Parameters	F1 score	time
time step =3	0.54	44s
time step = 5	0.58	38s
time step = 8	0.33	42s
stacked num = 1	0.02	46s
stacked num = 3	0.54	44s
stacked num = 5	0.25	110s

Table 1: The influence of different settings in LSTM.

Model	F1 score	time
LSTM	0.58	38s
Xgboost	0.75	0.6s

Table 2: The comparison between LSTM ans XGboost

Final model The final model is showed in Fig 4

4.2.5 Conclusion

The Table2 shows that the performance of LSTM is lower than XGboost on the data set. Due to this low performance, we will not spend much time on this kind of model any more.

4.2.6 Limitations

Although in this experiment, the LSTM is worse than the XGboost, it does not mean LSTM is 100% not suitable for this task. In this experiment, we only use part of the data to test the performance, however, we could not apply this result to the full data set due to the time and hardware limitation. Another question is that we may not find the best method to dealing with the sparse data set. Here we view the user-item pair as the unique id to group all the information. In this scenario, the data will still be sparse because one user may not have high frequent actions on one particular item. As an potential improvement, we can use user itself as the key to group the information, which will greatly reduce the data size and increase the density of the data, however, another problem is that the item information is limited, which will bring other problems to this task.

4.3 Machine learning models

4.3.1 Introduction of Machine learning models

- **Decision Trees**[2]:Decision tree model plays a vital cornerstone in classification. so we will use several powerful ensemble decision tree learning models which will decrease variance and bias both, such as, XGBoost, LightGBM. We used XGBoost in our experiment.

- **Artificial Neural Network**:ANN is based on a collection of connected units or nodes called artificial neurons with advantage of that NNs can achieve better performance on non-linear complex problems, however, it is time consuming to tune the parameters of NNs.

- **Ensemble learning**:Ensemble methods can achieve better predictive performance by gathering advantages and avoiding shortcomings of the above models, we used Gradient Boosting, XGBoost in our experiment.

- **Logistic Regression** Single model has the fastest speed, in case our dataset is really large, we used single model logistic regression to test our feature engineering result.

- **Naive Bayes** There is an strong unbalance between the positive labels and the negative labels, so we used generative model to do our experiment, this kind of model is regardless of the distribution of the label, we don't have do other things with the model to create enough recommendations.

4.3.2 Handle the imbalanced dataset

The positive labels only account for about 1% in the training data, which leads to a very unbalanced result in the test set. Only less than 10 sets of items are recommended at the end. Thus, instead of using the default threshold to assign labels, we decided to rank the predictions by their probability first and then select certain percentage of test data that are most likely belong to the positive class as the final result. To choose a proper percentage threshold, we plot the ROC curve to see how our model behaves under different percentage thresholds.
Figure 6 is the ROC curve under different percentage thresholds in the training data. Though the model has the highest precision when we assign 30% of testing data to positive class, the recall is extremely low. And in recommendation systems, high recall is more desirable. The ROC curve provides us a standard to choose percentage. We test the results using thresholds that provide acceptable recall and the scores we got in the competition can be found in Table 3.

Percent	1%	**2%**	3%	6%	NAN
F1 score	8.44%	**10.23%**	9.82%	7.28%	0
Accuracy	0.12	0.10	0.08	0.04	0

Table 3: The result of different percentage selected.

Model	RF	GB	LR	MLP	NB	XGB
F1 score	8.95%	9.50%	8.04%	4.57%	0.18%	**10.23%**
Accuracy	0.08	0.09	0.08	0.04	0.002	**0.10**

Table 4: The result of different model(best parameter after Grid Search).

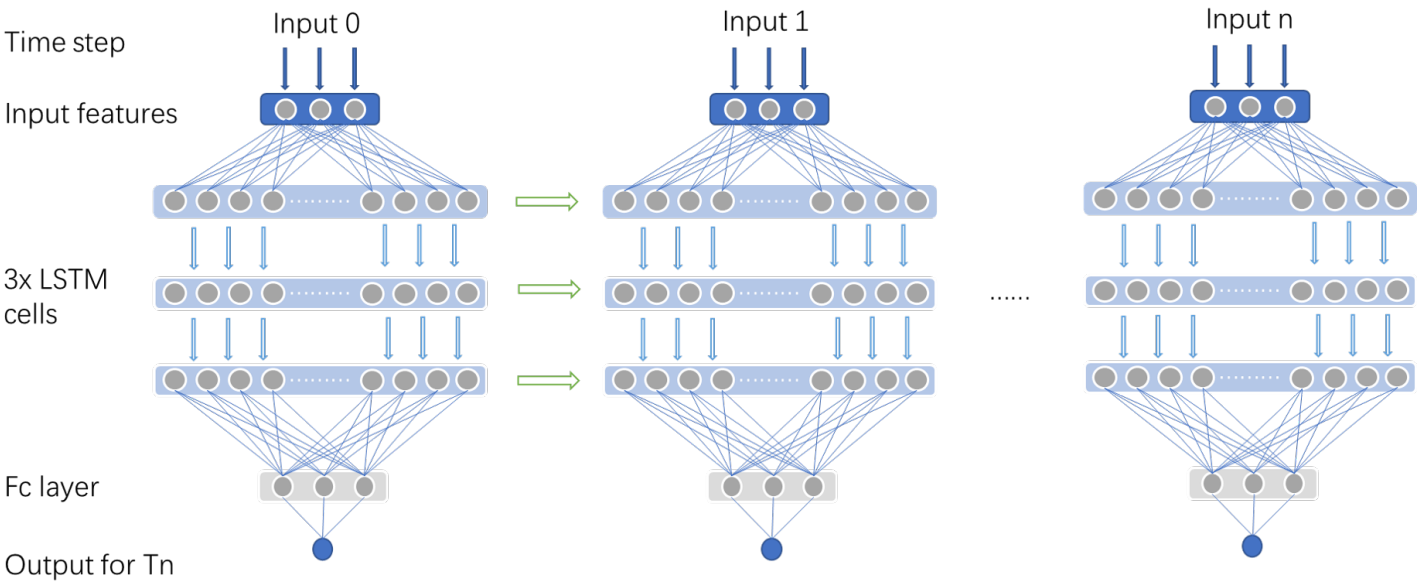

Figure 4: the structure of LSTM model

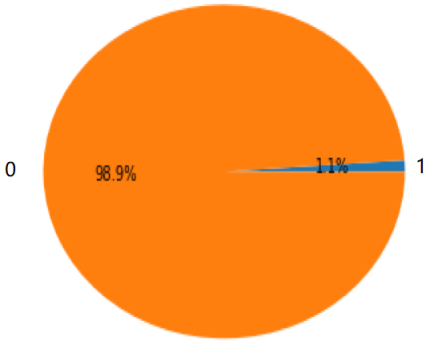

Figure 5: Class Distribution in Training Data

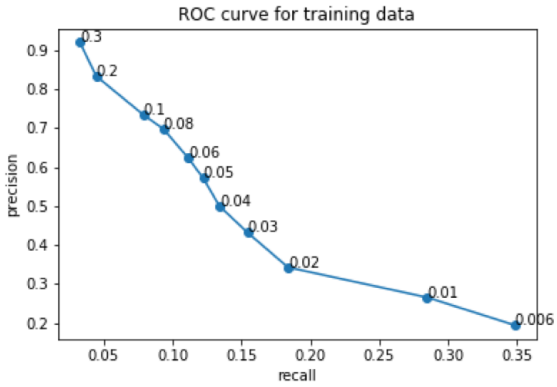

Figure 6: ROC curve for training Data

4.3.3 Conclusion

From the above tables we can figure out that the 2% is the best percentage for this question. And the XGBoost model performed best under these circumstances. That's why we finally chose the XGBoost model.

5 Results

The F1 score we got on Tianchi was 10.23% while the accuracy was 0.1. Our group rank the 43rd among more than 10,000 teams who participated in the competition.

6 Acknowledgements

XIE Yina(yxieat): Data Pre-processing and Feature Engineering for Time Series Model
WANG Ruolan(rwangbh):Data Pre-processing and Feature Engineering for Non-Time Series Model
Wang Zhicong(zwangee): Experiments and analysis on time series model LSTM. Help testing the non-TSM models.
Zheng Xiaowen(xzhengao): Evaluate the performance of model under different percentage thresholds.
Wang Zhe(zwangec): Experiments on machine learning models, do the grid search and find the best model with best parameters.

References

[1] Felix A. Gers and Jürgen Schmidhuber. Long short-term memory learns context free and context sensitive languages. In Věra Kůrková, Roman Neruda, Miroslav Kárný, and Nigel C. Steele, editors, *Artificial Neural Nets and Genetic Algorithms*, pages 134–137, Vienna, 2001. Springer Vienna.

[2] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.