[redacted]

数据挖掘实验报告

PREDICT THE CLICK-THROUGH RATE OF ADS GIVEN THE QUERY AND USER INFORMATION

组　　长 [redacted]

组　　员 [redacted]

组　　员 [redacted]

报告日期 Jan.11th 2019

Content

1. Experimental purpose

Search advertising has been one of the major revenue sources of the Internet industry for years. A key technology behind search advertising is to predict the click-through rate (pCTR) of ads, as the economic model behind search advertising requires pCTR values to rank ads and to price clicks.

CTR Prediction:

Ratio of users who click on an ad to the number of total users who view the ad:

$$CTR = \frac{Clicks}{Impressions} \times 100\%$$

Used to measure the success of an online advertising campaign.

In this task, given the training instances derived from session logs of the Tencent proprietary search engine, soso.com, participants are expected to accurately predict the pCTR of ads in the testing instances.

2. Experimental content

2.1 Data Introduction

The training data file is a text file, where each line is a training instance derived from search session log messages. To understand the training data, let us begin with a description of search sessions.

A search session refers to an interaction between a user and the search engine. It contains the following ingredients: the user, the query issued by the user, some ads returned by the search engine and thus impressed (displayed) to the user, and zero or more ads that were clicked by the user. For clarity, we introduce a terminology here. The number of ads impressed in a session is known as the ’depth’. The order of an ad in the impression list is known as the ‘position’ of that ad. An Ad, when impressed, would be displayed as a short text known as ’title’, followed by a slightly longer text known as the ’description’, and a URL (usually shortened to save screen space) known as ’display URL’.

We divide each session into multiple instances, where each instance describes an impressed ad under a certain setting (i.e., with certain depth and position values). We

aggregate instances with the same user id, ad id, query, and setting in order to reduce the dataset size. Therefore, schematically, each instance contains at least the following information:

Table-1 information of training data

Name	Value	Description
Click	0	The numbers of Clicks
Impression	1	The numbers of Impression
DisplayURL	14756578758696272233	The URL of ad
AdID	21546673	ID of the ad
AdvertiserID	3625	ID of the advertiser
Depth	2	The number of ads impressed in a session
Position	1	The order of an ad in the impression list
QueryID	2158764	ID of the query.
KeywordID	400887	ID of the keyword
TitleID	2048060	ID of the title
DescriptionID	1468187	ID of the description
UserID	0	ID of the user

Each of the samples represents a certain session information obtained from a user's query, where each user query word QueryID, advertising keyword KeywordID, the ad title TitleID, and the ad description DescriptionID all correspond to a string of tokens.

There are five additional data files, as mentioned in the above section:

1. queryid_tokensid.txt
2. purchasedkeywordid_tokensid.txt
3. titleid_tokensid.txt
4. descriptionid_tokensid.txt
5. userid_profile.txt

All file description:

Table-2 information of file

Filename	Records	Size
training.txt	149639105	9.9GB
test.txt	20297594	1.3GB
query_tokensid.txt	26243606	704M

descriptionid_tokensid.txt	3171830	268M
purchasedkeywordid_tokensid.txt	1249785	26M
titleid_tokensid.txt	4051441	171M
userid_profile.txt	23669283	283M

2.2 Data Analysis

The distribution of ad with clicking or not:

Table-3 the distribution of training data of click

Click	Quantity
Click = 0	142981068
Click > 0	6658036

As the following figure shows, 1 represents the 'click' > 0

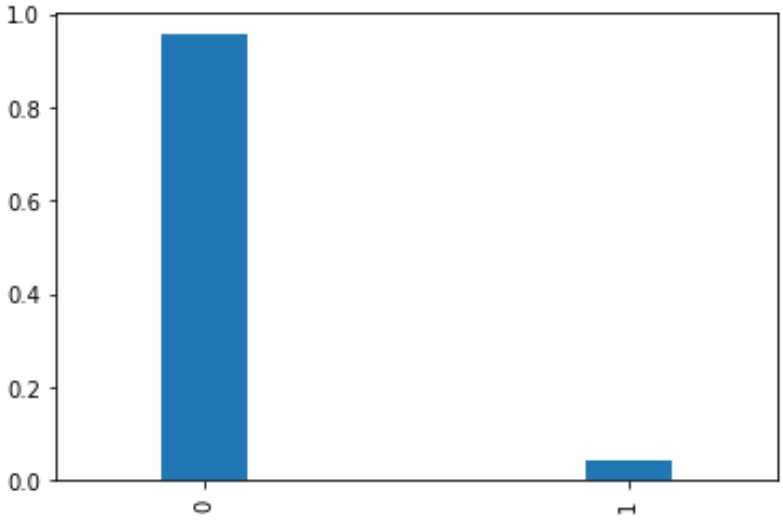

Figure-1 the distribution of training data of click

The ad which is clicked only account for 4.66% of all ads.

Table-4 the distribution of training data of user of click and impression

User	Click	Impression
User = 0	2475564	9931452
User != 0	5742069	136268327

The behavior of the unknown user makes it more difficult to analyze.

The '1' represents the 'User != 0 '

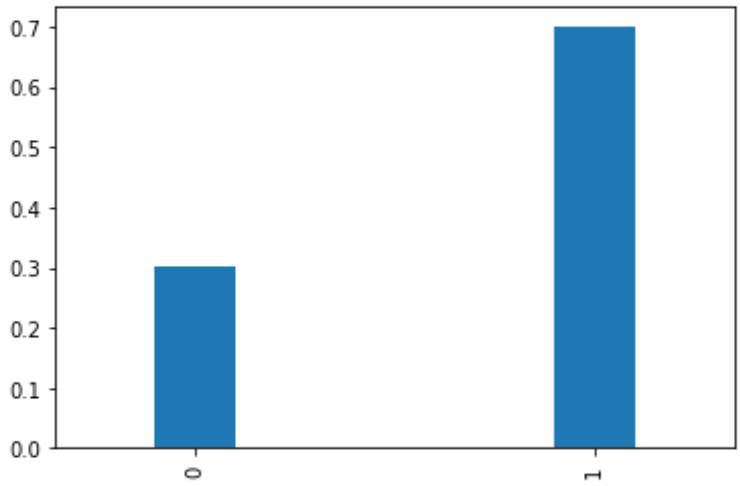

Figure-2 the distribution of user of click

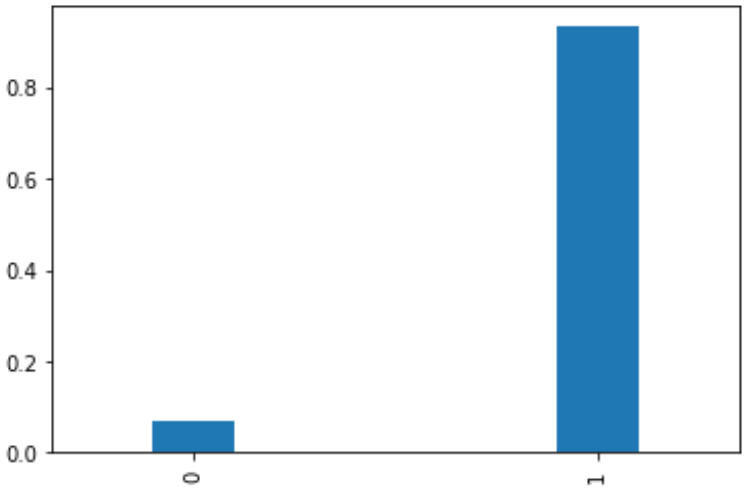

Figure-3 the distribution of user of impression

2.3 Evaluation

The evaluation metric for this task is the Area Under Curve (AUC), which is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.

2.4 Our works

After we discover that this project is the competition, then we look for some paper about this competition. So, our next is work is to do the things that the paper have written.

In this project, we took some detours and tried many methods, but we had the basis of the paper. Our overall direction is toward better results. Our main work is to firstly sample the data, secondly expand the data in accordance with the number of positive and negative samples, and then deduplicate them. After extracting the features, we one-hot these features.

The main flow is shown below.

1. Sample
2. Binary classification
3. Expand data
4. Remove repeated data
5. Feature exaction
6. One -hot
7. Model training
8. Prediction
9. Evaluation

3. Experimental process

3.1 Preprocessing

3.1.1 Sampling

The advertising data given is very large. Due to the limitations of experimental hardware, it will take lots of time to train the model when using all the data, which is not conducive to make a feature selection and have the model optimized. We sample randomly some data in the training data at a ratio as a training data set, what's more, we also use some data as a validation data set.

The training set and the validation set sampled from the original data set play an important role in our whole work, when we do this work, we can know the effect of our models, even though there exists the cold starting problem.

Sample:

1) Random sample from the training set

2) Separately sample from the training set which the click is zero and the click is not zero.

3.1.2 Binary Classification

We try to exploit binary classification model to solve the problem.For each instance that we have sampled,when the click is >0, we just simply to let it equal to 1.Then after doing this action, we have get a training set which has the click only equal to 1 or 0.Then we use the Bayesian to test the result.

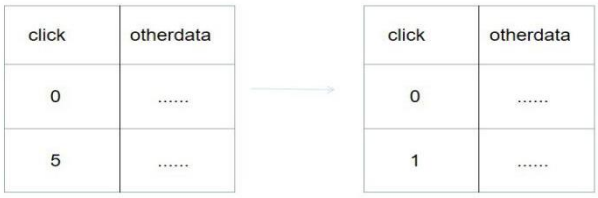

Figure-4 binary-classification

3.1.3 Expand Data

In our work, we assume that there should positive examples and negative samples. Therefore, we supplemented the original data with two positive samples and three negative samples is the data's click is 2 and the impression is 5. In this way, we have enough positive samples and negative samples for training. After doing this action, our result has an improvement.

3.1.4 Remove Repeated Data

Since the data in the above is enlarging, so we think that we can remove the repeated data, and only keep up single data, such that for two positive samples, we only retain one row of data instead. After doing this, our result is improved.

3.1.5 Feature Extraction

In my work, when we do the Byesian and LinearSVC and LR, we use the same features. According to the paper, when using different model, we should choose to use different set of features, we will do this work later.

The dataset contains information on UserID, AdID, QueryID, and etc. I use 'adid', 'position', 'queryid', 'keywordid', 'userid' as categorical features in our models.

In addition, we use depth, (depth-position) / depth, user's age and user's gender as addition features.

Table 5-The features that we use

ID	Feature Description
Value_AD	Value of AdID
Value_Query	Value of QueryID
Value_Keyword	Value of KeywordID
Value_position	Value of PositionID
Value_user	Value of UserID
depth	The de[th
Relative-depth	(depth-position) / depth
User_age	The age of the users
gender	User's gender

During the experiment, we find that (depth-position) / depth just has four values: 0, 0.5, 0.3334, 0.6667. I make an adjustment for this feature

$$deppos = \left\lceil \frac{(depth - position) \times 10}{depth} \right\rceil$$

3.1.6 One-Hot Encoding

We use the library to call the one-hot function to encode the former features we have get before together, and we get a parse feature. Then we train the data over the data after one-hot.

3.1.7 Feature extraction of LinearSVR

However, some categories come with only a few or even no instances. Computing the click-through rate directly for those categories would result in inaccurate estimations because of the insufficient statistics. Thus, we apply smoothing methods during click-through rate estimation.

$$pseudo\_CTR = \frac{\#\text{click} + \alpha\beta}{\#\text{impression} + \beta}$$

In this formula, set α as 0.05 and β as 75. We use this *pseudo_CTR* for *AdID*, *AdvertiseID*, *QueryID*, *TitleID*, *DescriptionID*, *UserID*, *KeywordID*, *Display Url*, *relative position (depth - position) / depth*, user's *gender* and user's *age*.

For example, to extract the *pseudo_CTR* of *AdID*.

```
pCTR_Ad = training.groupby(['adid']).agg({'click': np.sum,
          'impression': np.sum}).reset_index()
pCTR_Ad['pCTR_Ad'] = (pCTR_Ad['click'] + 0.05 * 75) / (pCTR_Ad['impression'] + 75)
pCTR_Ad = pCTR_Ad.drop(['click', 'impression'], axis=1)
training = training.merge(pCTR_Ad, on='adid', how='left')
```

We calculate the *pCTR_Ad* by grouping the training data by '*adid'* and calculate the summation of '*click'* and '*impression'*. Then use the *pseudo_CTR* formula to do smoothing and calculate the *pseudo_CTR*.

Especially, for the *gender* and *age pseudo_CTR* features. We first need to join the *training.txt* and *userid_profile.txt* on *userid.*

```
training = training.merge(userid_profile, on='userid', how='left')
```

Then we fill the *NaN* feature in the joined new table, because some unknown users or some users are known but do not fill the *gender* or *age*.

```
values = {'gender': 3, 'age': 0}
training = training.fillna(value=values)
```

We then stack all the *pseudo_CTR* features by merging all the feature into one long vector.

Table 6-pseudo_CTR features

ID	Feature Description
pCTR_Ad	Pseudo click-through rate of AdID
pCTR_Advertiser	Pseudo click-through rate of AdvertiserID
pCTR_Query	Pseudo click-through rate of QueryID
pCTR_Title	Pseudo click-through rate of TitleID
pCTR_Description	Pseudo click-through rate of DescriptionID
pCTR_User	Pseudo click-through rate of UserID
pCTR_Keyword	Pseudo click-through rate of KeywordID
pCTR_Url	Pseudo click-through rate of Display Url
pCTR_RPosition	Pseudo click-through rate of relative position (depth - position) / depth
pCTR_Gender	Pseudo click-through rate of user's gender
pCTR_Age	Pseudo click-through rate of user's age

3.2 Model and Approach

The main structures of the following two approaches are showed as follows:

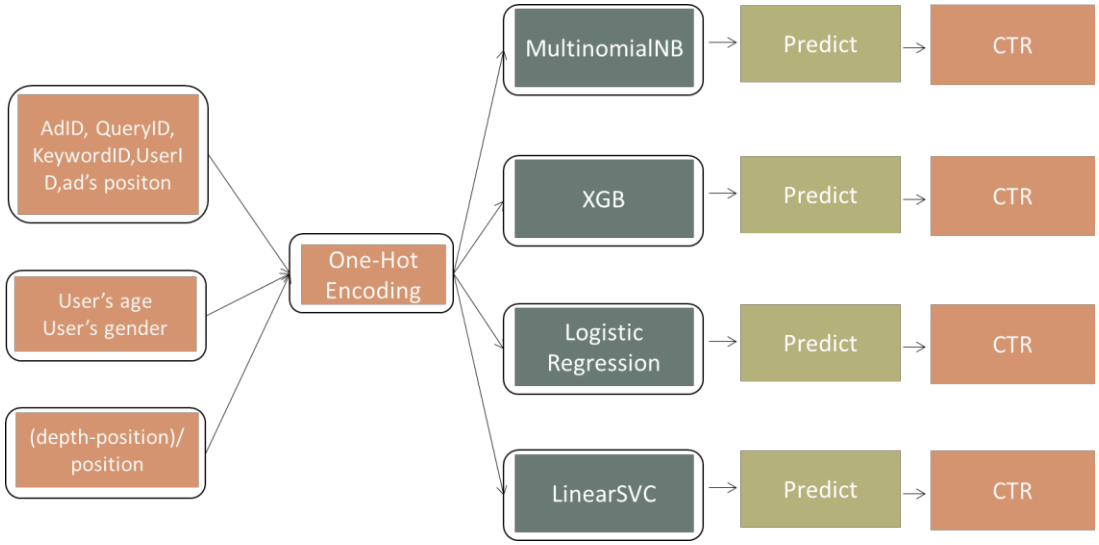

Figure-5 the structure of the classification approach

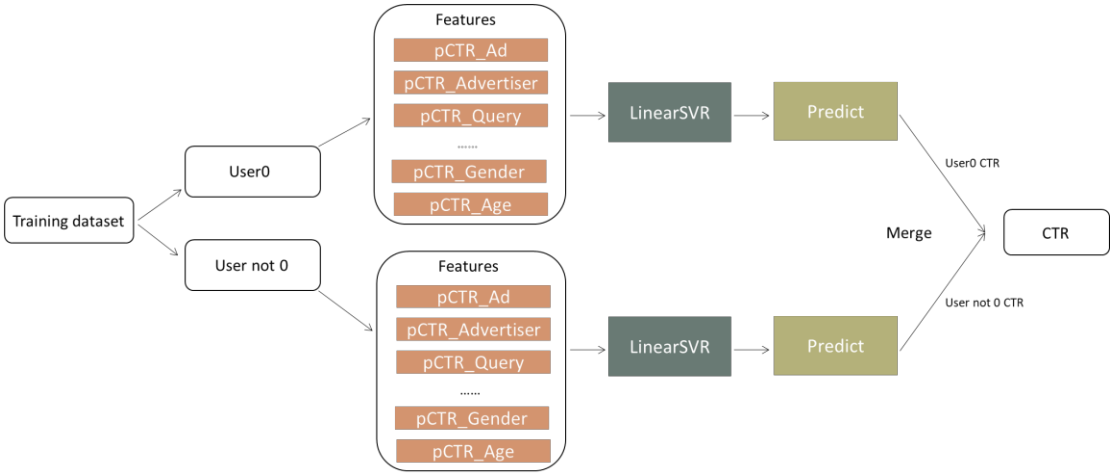

Figure-6 the structure of the regression approach

3.2.1 Bayesian

Naïve Bayes is a simple probabilistic classification model which is based on Bayes's Theorem. It exploits a strong independence assumption to efficiently handle a large dataset. We adopt the multinomial model for each feature in naïve Bayes. After getting a probabilistic classifier, we use the estimated conditional probability as the ranking criteria.

First, we didn't use one-hot encoding to encode the feature. And the score of AUC got 0.67. And then, I used the one-hot encoding, and trained again, there result improved.

Through adjusting the alpha from 0.01 to 0.8, we got different AUC score, and we

find that the alpha between 0.7 and 0.8, the result got higher. The highest score we got is 0.743.

3.2.2 LinearSVC

After doing the pre-processing, we call the library of LinearSVC to fit the training data and the predict the result. And we use the parameter classweight=’balanced’ to do this work, and have a result of 0.723, since it is not as high as the Bayesian that we have done before, so we finally give this method.

3.2.3 LR

Logistic regression is the most widely used CTR prediction model in the industry.

In my experiment, we also randomly sample the training data to train logistic regression model. we have tried different number of samples; the result gets better when the data sample size is larger.

For the logistic model, we also have tried the different feature. The raw categorical features, the categorical features with addition feature, and the feature with or without one-hot encoding.

In the meantime, in my model, we use the penalty of l2 norms to avoid overfitting problem and set the different C (Inverse of regularization strength) to train the model.

Then, through analyzing the parameters of logistic regression, we find that the solver (algorithm to use in the optimization problem) can make sense.

In the first, we use the liblinear which iteratively optimizes the loss function using the coordinate descent method, and then we use the 'sag' (the random average gradient decreases) which is more suitable for large data.

For LR with liblinear, when the scale of data enlarged from 20 million to 60 million, the results of AUC from 0.723 to 0.759. And for the LR with 'sag', the best AUC score we get is 0.7759.

In addition, we have also tried the SVC method, however, it didn’t make sense. After training two days, it still didn’t end up. Finally, I gave it up.

3.2.4 XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework.

I adopt the XGBClassifier model for each feature in XGBoost. And the features are using one-hot encoding. The AUC socre for the first sample method is 0.643, and for the second method, the score is 0.649.

3.2.5 LinearSVR

Because we view the prediction task as a regression task, so we naturally use SVR as our regression model. More concrete, we use LinearSVR as our regression model. We use all the 11 statistical features which are all numerical features, as input features of our LinearSVR.

After that, we also tried the simpler LinearRegression model and found that it has a little lower AUC but more efficient in the training progress.

We separate the user0 and user not zero into two training sets, because the user0 is the user that system cannot distinguish. These users have some strange behaviors comparing to the user not zero.

We then trained two regression models on the dataset user0 and user not zero, finally we got two regression models. One is to predict the CTR of the user0, the other is to predict the CTR of the user not zero. After the prediction, we got two lists of CTR. We merged the two lists into a single one, and use this merged list as our result on the whole to calculate the AUC score.

In prediction progress, we use the statistical features in the training set. For example, we use the *pseudo_CTR* of *AdID* to set the same *AdID*'s *pseudo_CTR*, as one of the test samples features. Same as *AdvertiseID*, *QueryID*, *TitleID* etc.

And, if the id is not in the training set, we use the *Average pseudo_CTR* in the training set to fill the unknown id in the test set.

Do the same operation as we did in the training set. Separate the test dataset into two subsets, one set is user0, the other is user not zero. Use the futures describe above as the futures of test data set. Feed it into the two regression models and we can get two prediction lists for the CTR. And, we merge the two lists of CTR into a single one as CTR on the whole test set.

Use the CTRs in the final list as the sort key in AUC. After tuning, we got the best result about 0.75 AUC score in user not zero, and about 0.73 AUC score on the user0. However, when we merged the two CTR lists, we finally got about 0.73+ AUC score on the whole test dataset.

3.2.6 Model Combination

According to the paper, we can see that they combine the models using a non-linear method, however, we think that can we combine the models using linear method. However, the main job for this job is to find the best weight that can combine these two model's result perfectly. We just choose to normalize the weight to do the job.
That is

$$\text{predict} = predict_A * \frac{result_A}{\sqrt{result_A^2 + result_B^2}} + predict_B * \frac{result_B}{\sqrt{result_A^2 + result_B^2}}$$

However, we have found a problem that they are not in the same order, so we normalize each of the two predict probability to prove that they are in the same order. After doing this action, our result has improved a lot, from 0.7759 to 0.7792.

4. Experimental results and analysis

4.1 Result and Analysis

We have tried different size sample of the data, and adjust the parameters of the model, we get different result.The best results of each model are showed as follows:

Table-7 the best result of each model

Model	AUC
XGB	0.649
MultinomialNB	0.743
LinearSVC	0.752
LinearSVR	0.73
LR	0.7587
LR with sag	0.7759
Combined	0.78337

Bayesian is the first method that we have tried, from using the binary classification to adding some positive samples and negative samples to expand the data, after doing this action, we find that this result improves a little. Then we find that this is have too many data, so we remove something repeated, after doing this action, we find that this makes the result a little improved. Then, we try to one-hot the features, after doing this, we find that the result still improved. Also, we have found that when we sample more data from training set, the result will get better.

For LinearSVR, the AUC score on the user not zero is well but the AUC score on the user0 is low. And, the final result to combine the two we get a result not so good. Because of the poor performance on the prediction of the user0.

4.2 Conclusion

We did this project through a process from scratch, during which we explored a lot of things, and later found that this is the competition project of KDD CUP2012.By searching the relevant content on the Internet, we found the paper related to this competition. By reading the content in the paper and referring to the practice in the paper, we gradually understand the process of this competition and know that we should adopt different methods to improve our performance.

4.3 Further Work

1.using other files

We can use the similarities between the known and unknown items to reduce the impact of cold-start problem. That means to use the features of the most similar known items as the features of unknown items. Or we can use the cosine similarity between query tokens and title tokens and a new feature. Because according to common sense, if the user query content is more similar to the Ad title that showed to the user, the user can be more likely to click the Ad.

We can also use the length of the tokens of query as a new feature, because the longer query may contain more detail information of the user.

So, it may be more precise to recommend the proper Ads to users.

2. DeepFM

As we learned from the class, we know that it is a good way to predict CTR. However, due to time limitation, we haven't tried it. In the future, we can have a try.

3. Learn more about cold starting problem

We have found that some users in training set don't have relevant data in the testing set, so there exists a problem of cold starting. This is a common problem in the recommendation system. To solve this problem, we look for the recommendation book to try to solve this problem.

References

[1] 邓丽芳. 搜索广告点击率预测中的冷启动问题研究[D]. 2016.

[2] Kuan-Wei Wu, Chun-Sung Ferng, Chia-Hua Ho, An-Chun Liang, Chun-Heng Huang, Wei-Yuan Shen, Jyun-Yu Jiang, Ming-Hao Yang, Ting-Wei Lin, Ching-Pei Lee, Perng-Hwa Kung, C Wang, Ting-Wei Ku, Chun-Yen Ho, Yi-Shu Tai, I-Kuei Chen, Wei-Lun Huang, C. K. Chou, Tse-Ju Lin, Han-Jay Yang, Yen-Kai Wang, Cheng-Te Li, Shou-de Lin, Hsuan-Tien Lin. A Two-Stage Ensemble of Diverse Models for Advertisement Ranking in KDD Cup 2012. Department of Computer Science and Information Engineering, National Taiwan University, 2012.