Project Report for COEN296: Tencent Multimedia Ads pCTR Challenge

Shouchun He 1008350
he.shouchun@gmail.com

Christopher Yu 00000478904
cyu813@gmail.com

Abstract

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, which drives the pricing model. The objective of this project is to solve a multimedia advertisements pCTR prediction problem with Machine Leaning methodologies: We do classification on the advertisement titles; retrieve the image attributes (average Hue, Saturation, and value, R, G, B, and file size); group the end users by age and gender; calcuate the pCTR of each advertisement under each age and gender group; and finally build the models between the pCTR and advertisement attributes for each age and gender group. In this project we used classification knowledge and E-M algorithm in the ad. title classification, and we also used the logistic regression in building the models.

1 Project background

This project is solving an ICME challenge which came from:

`http://www.icme2014.org/tencent-multimedia-ads-pctr-challenge`

1.1 Dataset introduction

The challenge provides the advertisement data, which includes the titles (34163 rows, include some blank titles) and image files (24125 files, include one corrupted image file) of the advertisements, the age and gender of all users((23,439,495 lines, 491MB in size), and their behavior data (user id, ad title id, ad image id, impression, and click). The challenge sponsor (Tencent) provided three group of behavior data which are training dataset (23,906,738 rows, 672MB in size), evaluation dataset (3,253,943 rows, 91MB in size), and validation dataset (3,236,631 rows, 91 MB in size).

1.2 Additional requirements

All the data provided by the challenge sponsor are raw data. So we need to process them before using them to build pCTR prediction models. We need to do at least the following processings:

- Process the ad titles and classify them
- Process the ad images and get the attributes of them
- Combine the behavior data with the user data, and calculate the pCTR for the advertisements

1.3 Approaches

So, we will take the following approaches:

- Analyze Ad Image features
- Classification of Titles
- Processing User Data
- Split Training Data by Gender and Age
- Build the model

The details will be introduced in the following chapters.

2 Analyze Ad Image features

2.1 Features selection

As we know, each image contains lots of attributes, such as, colors (RGB), Hue/Saturation/Value, and the image sizes. Sometimes the colors and HSV distributions may cause some visual impact to the users who see the image. When files transferred on Internet, the image size may impact the image loading time and somehow it also impact the user experience. To simplify the feature analysis, here we only selected 7 features: Image size; average RGB color values (0 - 255); average H/S/V values (0 - 255).

2.2 Image analysis tool

We found a web based image analysis API:

`http://mkweb.bcgsc.ca/color_summarizer/?api`

So we wrote a program to call the web API by passing the URL of the ad image file and then analysis the HTTP response, and pick out the 7 features we need and save them into a text file for future use. The web API needs about 6 seconds to process an image file on average. (Fortunately the ad image files are as small as 30KB in average. The bigger the image size, the longer processing time.) So it took about 40 hours to process all the image files. The program was periodically terminated by the server, but it can continue from the terminate point after the restart.

2.3 Source code and output files

The files could be found on the design center servers (linux.scudc.scu.edu): /web-pages/she/GetImageInfo/quick

The Java source code: `http://linux.students.engr.scu.edu/~she/images/ImageInfoRetrieval.java`

To compile and run the source code, you need to download and install Java Http Client library from:

`http://hc.apache.org/httpclient-3.x/`

Output:

`http://linux.students.engr.scu.edu/~she/images/result`

The information of each columns are: <image id >, <image size >, <R >, <G >, <B >, <H >, <S >, <V >.

2.4 Notes and work in the next step

1. One image file was corrupted, so there are only 24124 rows in the result file.

2. The results will be sorted by the image ID in the future, which can provide better performance.

3 Classification of Titles

It is obviously that the titles of the advertisements can be divided into different categories. And different users may have different preference to certain category of advertisements. Such preference absolutely effects the CTR rate.

So we need to classify all the titles into different categories and then put this as a factor of our pCTR model. BTW, just like the age and gender, this factor is also not a linear factor but a category factor.

3.1 Manual classification

At the first step, we manually scan the titles.txt with our eyes, and identify there are about 6 main categories: Mens wear, Womens wear, Gaming, Beauty, Household, Education, etc. For those not belong to these categories, we marked as "Other". Then we identified some keywords in each category. You can refer those keywords from:

`http://linux.students.engr.scu.edu/~she/titles1/keywords.xlsm`

In the first worksheet of this file, we put some of the keywords we manually picked out from the titles. In the second worksheet, we wrote a macro to generate a matrix between the keywords and the categories: 1 means if a title contains this keyword that means it may belongs to the category; else 0.

We wrote another Java program to scan through each title, and keep weight of each Category that the keywords are from. The category with the highest keywords weight will be the label for that title.

However we found that this method is not so good: 1) We need to spend lots of time on identifying the keywords; 2) In the result, we found that most of the titles have equal weight for more than one category, so it is not easy to classify which category it really belongs to; 3) The method is not generic – if we get a new set of titles, we need to identify the keywords again – we could not take advantage of "Machine Learning" in tihs classification.

The files could be found on the design center servers (linux.scudc.scu.edu): /webpages/she/titles1/

Source code:

`http://linux.students.engr.scu.edu/~she/titles1/Classify.java`

Input: Keywords-Category Matrix in UTF-8 encoding and Unicode:

`http://linux.students.engr.scu.edu/~she/titles1/keywords.txt`
`http://linux.students.engr.scu.edu/~she/titles1/key_unicode.txt`

Titles:

`http://linux.students.engr.scu.edu/~she/titles1/titles.txt`

Result:

`http://linux.students.engr.scu.edu/~she/titles1/titles.classify`

3.2 Automatic classification

This section is the most innovative part of this report.

General idea: To solve the problems in the "manual classification", we need to discover the frequently appeared keywords, and their relationship with the titles, and then use E-M algorithm to divide those frequently appeared keywords into several groups based on their occurence in the titles.

Approaches:

1. Scan the characters (include Chinese characters, Latin characters, and special characters) in all titles, and count their appearance.
2. Combine the characters to form words, and try to form as long as possible words which appeared in the titles.
3. Update the appear frequency of each word (substract the apearence from their sub words or contained characters).
4. Pick out the words which appear more than 5 times.
5. Create a keyword-title Matrix to represent the occurence of the keywords in the titles.
6. Run E-M algorithm to divide the keywords into 6 categories until it can be used for classifying the titles.

The files could be found on the design center servers (linux.scudc.scu.edu): /webpages/she/titles/

Source code: Keywords analysis:

`http://linux.students.engr.scu.edu/~she/titles/AnalysisKeywords.java`

The E-M algorithm of auto classification:

`http://linux.students.engr.scu.edu/~she/titles/AutoClassify.java`

Input:

`http://linux.students.engr.scu.edu/~she/titles/titles.txt`

Keywords parsing results:

`http://linux.students.engr.scu.edu/~she/titles/titles.classify`

Individual Keywords Appear Frequency:

`http://linux.students.engr.scu.edu/~she/titles/title_keywords.txt`

The keywords classification:

`http://linux.students.engr.scu.edu/~she/titles/kwds_clsfy.txt`

The titles classification:

`http://linux.students.engr.scu.edu/~she/titles/ttl_clsfy.txt`

4 Processing User Data

The user dataset contains more than 20 million rows of user information, include user ID, age, and gender. To make the prediction more accurate, we plan to sort the user IDs, and then group the user IDs by age and gender into small groups.

4.1 Sort User Data by user ID

From the users.txt file provided, we sorted by built binary search trees of the user data. At the beginning, we created an array of binary search trees. Then we read each row of user data from the file, and check the uid and add it to one of the binary search tree whose index is (UID/10000). So each binary search tree contains no more than 10000 nodes and will not cause stack overflow in tree visiting because of inbalance of tree shape due to the huge amount of nodes.

4.2 Generate Hash Value of Gender and Age

As we know, the gender and age of users have significant impact with the preference of advertisements. We have generated a hash function to generate a pseudo rating from the user's gender and age. So that we can split the training data, testing data and validation data into groups by gender and ages and build different models for them.

During the data processing, we found that in some rows the gender values or the age values are missing. So we considered those special cases in our hash function:

$$f(gender, age) = \begin{cases} 28, & \text{if no gender, no age} \\ 29, & \text{if male, no age} \\ 30, & \text{if female, no age} \\ 31, & \text{if } age \leq 15 \\ age * 2 + gender, & \text{if } 15 < age \leq 65, \text{ (male = 1, female = 0)} \\ 131, & \text{if } age > 65 \\ 132, & \text{if no gender, } age \leq 15 \\ age + 117, & \text{if no gender, } 15 < age \leq 65 \\ 183, & \text{if no gender, } age > 65 \end{cases}$$

4.3 Source code and Output

The Java source code:

`http://linux.students.engr.scu.edu/~she/sort_users/SortUser.java`

The original user.txt:

`http://linux.students.engr.scu.edu/~she/sort_users/users.txt`

Output:

`http://linux.students.engr.scu.edu/~she/sort_users/sortedusers.txt`

The first column is sorted user id, and the second column is the hashed gender and ages.

5 Splitting Training Data and Calculate CTR

5.1 Splitting Training Data by Gender and Age

When splitting the training data, we decided to split by gender-age hash number generated by the user in the previous section. We will scan the training dataset line by line and use the UID paramter to split into separate files. The splitted files will be grouped together by the gender-age hash number (ie, '31.txt' will contain all users that have gender-age hash number of 31). The tree array built in previous section will be used to search by UID for the matching gender-age hash.

The Java source code:

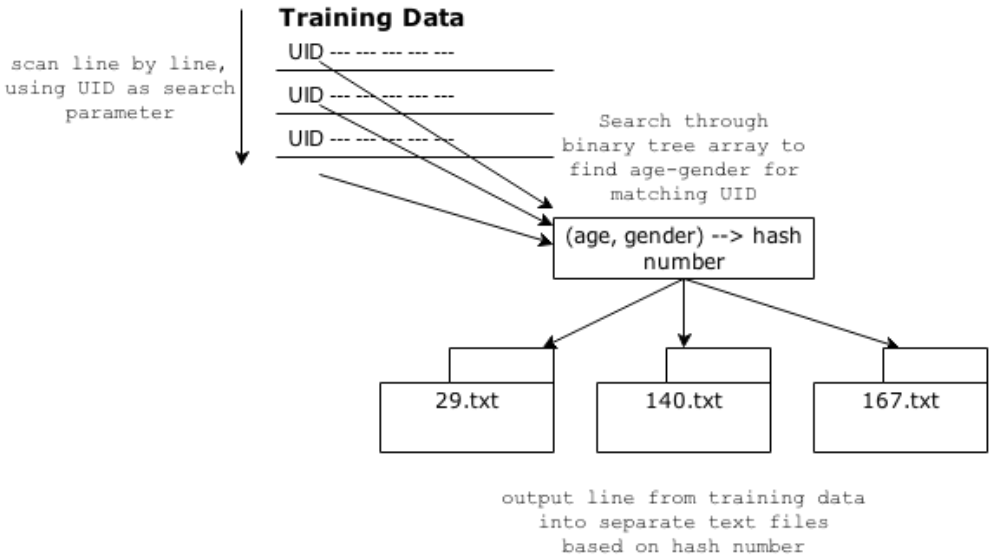
http://linux.students.engr.scu.edu/~she/Split/SplitData.java

The input:

http://linux.students.engr.scu.edu/~she/split/training_dataset
http://linux.students.engr.scu.edu/~she/split/evaluation_dataset
http://linux.students.engr.scu.edu/~she/split/validation_dataset
http://linux.students.engr.scu.edu/~she/sort_users/sortedusers.txt

Output:

http://linux.students.engr.scu.edu/~she/split/splited.html



5.2 Calculate the CTR

Once the training data splitting is complete, we will go through each file and merge the results by AdID. We will add up all the impressions and clicks of matching AdID and sum them up. Then CTR can be calculated by dividing number of total clicks into total number of impressions.

(This step is still on-going.)

6 Build the Model and Make Prediction

6.1 Build the Model

We will organize each hash group by title category, then we will use a logistic regression model to simulate pCTR using calculated CTR of each image as the y variable and the image features as the x variables. After logistic regression there should be a weight attached to each variable, and they will be used to calculcate pCTR of the testing data.

$$Logistic\,Model(Hash\,\#,\,AdCateogry) = Image\,Features(f_1, f_2, f_3, ..., f_7) \backsim CTR_{calculated} \Rightarrow (w_1, w_2, ..., w_7)_{Hash\,\#, Ad\,Category} \tag{1}$$

For each Gender/Age Hash Group:

Title Category	Ad Image ID	Parameters(X_1, X_2, X_3, ...X_7)	CTR (Calculated)
1	$image_1$1	file size, RGB, HSV, …	CTR_1
	$image_1$2	file size, RGB, HSV, …	CTR_2
	...	file size, RGB, HSV, …	CTR_i
2	$image_2$1	file size, RGB, HSV, …	CTR_1
	$image_2$2	file size, RGB, HSV, …	CTR_2
	...	file size, RGB, HSV, …	CTR_i
...			
...			

6.2 Validate Model and Make Prediction

To validate the model and make predictions, we plan to apply the Model we made to the splited evaluation dataset and validation dataset, and worked out the pCTR for different advertisements (based on title category and image features) and users from different gender and age groups, and then make a summary to test whether the pCTR is accurate.

(This step is still on-going.)

7 Software Used

The project will use, primarily, Java, Microsoft Excel, MySQL, and MatLab. Depending on future considerations for calculation efficiency and convenience, software capable of heavy mathematical modeling and computation will be required. Java will be used mainly for application-side utilities.

8 References

Bayesian Statistics

http://www.maths.nott.ac.uk/personal/tk/files/talks/nott_radiology_01_11.pdf

Gaussian Process

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.1226&rep=rep1&type=pdf

Nearest Neighbor Algorithm

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.1422

Machine Learning

http://www.cs.nyu.edu/~mohri/mlbook/