User Behavior Modeling

Avery Wu, Tsung-Yi Huang, Minghong Zheng,
yw2928@columbia.edu, th2668@columbia.edu, mz2597@columbia.edu,
Data Science Institute, and Department of Computer Science,
Columbia University, New York

***Abstract*—Ad click through rate(CTR) can mainly affect advertisers revenue, and thus it provides the most relevant metrics of return on advertising investment. Therefore, targeting right users can efficiently improve ad CTR. In this paper, given users interests and activities of past several months from Yahoo, we analyzed the data and built three models (Näive Bayes, SVM and Logistic regression) with different assumptions to predict user behavior to improve ad CTR as well as ad revenue.**

***Keywords*—Ad CTR; user activities; user interests**

I. Introduction

Internet has emerged as an important medium for advertising. In the year of 2011, the companies in U.S. have spent $31.7 billion on online advertisements. For most companies, what they care about in online advertising is not how many posts exist on web pages, but rather how many people are clicking on their advertisements. Motivated by maximum efficiency, Overture from Yahoo! invented the first performance-based pricing model that charges according to *cost per click* (CPC); this approach is quickly adopted by Google and Yahoo! in 2002, and is currently the most widely used pricing model for online advertising [1].

Using the user-advertisement behavior big data provided by Yahoo! [2], we are able to scrutinize the variation of user's personal interest towards advertisements across various fields. Understanding their interrelationships would greatly help in suggesting advertisements to online users, and therefore benefits CPC estimating and company profit maximizing.

In this project, we model millions of users' personal interest through their ad-clicking history, and estimate their attitude towards online advertisements that belong to new fields, such that the user have never seen advertisement in that field before. From this project, we aim to dig up more hidden profit in the business areas, using the combination of machine learning algorithms and big data system techniques.

II. Related Works

Machine learning has been working effectively in predicting user behavior based on their past actions. In *Machine learning for targeted display advertising: Transfer learning in action* [3], C. Perlich traced user's online activities as the target dataset, and learned multiple functional mappings in parallel to reduce the high-dimensional feature data space. Similarly, in our project, the reduction is achieved using Principal Component Analysis (PCA), which applies orthogonal transformation to deduce a set of linearly uncorrelated variables as modeling features. C. Perlich then built a linear model on the reduced data, using logistic regression trained by stochastic gradient descent (SGD) to classify the type of user's online activity. However, this method is not entirely suitable for our project, since big data is abundant in size and SGD would require many more loops to update the parameters using the randomized data. Therefore, instead of that, we employ a sample data to estimate the model parameters before running logistic regression on the bigger data set.

Aside from classifying online activities, researches on the click-through rate modeling also provided guidance to this project. In *The Business Next Door: Click-Through Rate Modeling for Local Search* [4] from AT&T Labs-Research, S. Balakrishnan models the click-through rates (CTR) of advertisements by Bayesian relational dependency networks including ordered Gibbs sampling. For our case, since Näive Bayes on multinomial variables is sufficient for deriving a precise prediction method, we leave the Bayesian relational networks for future works.

III. System Overview

A. Data

The data we employed in our project is a sample data of several months of user activities on their interest areas at Yahoo web pages [2]. Each user is represented as one feature vector and its associated labels.

- Feature Vector:
  - 13346 Features
  - Each dimension of a feature vector quantifies a user activity with a certain interest category during a training period of 90 days, which is derived from user interactions with classified pages,ads,and search results
  - Users clicking ad of a category usually have feature values at the same columns
  - Features values:
    - values measures users' engagement to Yahoo's web pages.
    - They are calculated from page views, search queries, search result clicks, sponsored search clicks, ad views,and ad clicks
    - Larger feature values indicates that users were more active at the web page
- Label:
  - 380 Labels
  - Label ID represents a interest category ID
  - There exists a hierarchical structure among the labels.
  - Label values:
    - -1: if a user saw an ad but did not click
    - 1: if a user saw and clicked on the ad
    - 1 and -1 propagate upward till root
    - 1 will overwrite/block -1 along the path

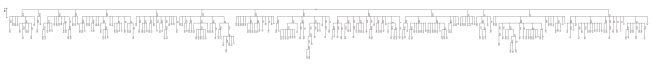

Fig. 1. The taxonomy tree structure of labels

B. System Design

In this project, we use HDFS and HBase as underlying storage and database, Spark and scikit learn as machine learning related tasks processing and modeling engine, and Tableau as visualization tool.

Using HBase instead of other database such as mySQL and Hive give us the advantage of quickly selecting the features from users that we are interested in. Usually when training models targeting some label IDs, we only want to use the feature vectors that have values on those labels as input, instead of using all feature vectors. Being a column oriented storage system, HBase can pick out and filter a particular column quickly without the need to traverse through each column.

We have considered three ways to transfer and parse the features. The most directive one is to read the raw feature file in file system and use MapReduce to parse each line of features, before selecting out the features we want using tools such as sparkSQL. This is not a good way because it transfers all features into the environment and sparkSQL will have to iterate through each and every feature value just to reach a particular column and decide if this row should remain. A better way is to use HBase, and transfer the feature vectors after selecting them as mentioned above. The drawback of the method is that although HBase can find out the feature vectors we want very fast, transferring each feature value still requires HBase to iterate through each cell of a feature vector. Being a column oriented NoSQL database, HBase has no advantage over other databases in this kind of task. The third way is a mixture of these two methods. In HBase, We stored the labels as usual, but we stored the features in raw format, and use MapReduce to parse the features into feature vectors after selection. This way HBase only have to handle the features as a single column instead of more than 10 thousand columns. The trade off for speed is more memory consumption because the raw feature are stored as strings instead of parsed sparse vectors.

Some HBase query result is shown as in Figure 2.

IV. Algorithm

A. Linear SVM

The linear SVM is a standard method for large-scale classification tasks. It is a linear method that

```
hbase(main):011:0* scan 'train', {LIMIT => 5}
ROW                                   COLUMN+CELL
 1                                    column=f:10003, timestamp=1, value=27
 1                                    column=f:10005, timestamp=1, value=11.13274
 1                                    column=f:10026, timestamp=1, value=2.30243
 1                                    column=f:10051, timestamp=1, value=0.68255
 1                                    column=f:10094, timestamp=1, value=3.03412
 1                                    column=f:10104, timestamp=1, value=60
 1                                    column=f:10113, timestamp=1, value=18
 1                                    column=f:10127, timestamp=1, value=3
 1                                    column=f:10159, timestamp=1, value=54
 1                                    column=f:10176, timestamp=1, value=14
 1                                    column=f:1018, timestamp=1, value=67
 1                                    column=f:10208, timestamp=1, value=5.36329
 1                                    column=f:10238, timestamp=1, value=27
 1                                    column=f:10244, timestamp=1, value=75
 1                                    column=f:10262, timestamp=1, value=6
 1                                    column=f:10271, timestamp=1, value=48
 1                                    column=f:10272, timestamp=1, value=75
 1                                    column=f:10298, timestamp=1, value=202.6624
 1                                    column=f:10317, timestamp=1, value=6.20604
 1                                    column=f:10334, timestamp=1, value=11
 1                                    column=f:10340, timestamp=1, value=1.16233
 1                                    column=f:1035, timestamp=1, value=12
 1                                    column=f:1039, timestamp=1, value=41
 1                                    column=f:10398, timestamp=1, value=1.7435
 1                                    column=f:10410, timestamp=1, value=8.8412
 1                                    column=f:10424, timestamp=1, value=3
 1                                    column=f:10435, timestamp=1, value=3
 1                                    column=f:10445, timestamp=1, value=75
 1                                    column=f:10494, timestamp=1, value=17.36714
 1                                    column=f:10498, timestamp=1, value=11
 1                                    column=f:10515, timestamp=1, value=37
 1                                    column=f:10546, timestamp=1, value=0.88638
 1                                    column=f:10556, timestamp=1, value=2.8238
 1                                    column=f:10558, timestamp=1, value=28
 1                                    column=f:10565, timestamp=1, value=0.50999
 1                                    column=f:10586, timestamp=1, value=6.19721
 1                                    column=f:10588, timestamp=1, value=60
 1                                    column=f:10639, timestamp=1, value=12.20142
 1                                    column=f:10667, timestamp=1, value=1.16003
 1                                    column=f:10709, timestamp=1, value=4.76753
 1                                    column=f:1073, timestamp=1, value=75
 1                                    column=f:10754, timestamp=1, value=3.85022
 1                                    column=f:10756, timestamp=1, value=7.62311
 1                                    column=f:10770, timestamp=1, value=11
 1                                    column=f:10813, timestamp=1, value=53
 1                                    column=f:10820, timestamp=1, value=0.87738
 1                                    column=f:10823, timestamp=1, value=54
 1                                    column=f:10832, timestamp=1, value=1.43553
 1                                    column=f:1084, timestamp=1, value=13
 1                                    column=f:10860, timestamp=1, value=0.88638
 1                                    column=f:10869, timestamp=1, value=75
 1                                    column=f:10870, timestamp=1, value=2.90583
 1                                    column=f:10909, timestamp=1, value=32
```

Fig. 2. Features of the first user in the training data set.

solve the optimization problem $\min_w f(w)$ where w is in R^d and the objective function is of the form

$$f(w) := \lambda\, R(w) + \frac{1}{n}\sum_{i=1}^{n} L(w; x_i, y_i)\ .$$

Here $\lambda \geq 0$ is the regularization parameter defines the trade-off between the two goals of minimizing the loss L(i.e., training error) and minimizing model complexity R(i.e., to avoid over-fitting), the vectors x_i in R^d are the training data examples, for $1 \leq i \leq n$, and y_i in R are their corresponding labels, which we want to predict. We used linear SVM with L2 regularization and the loss function in the formulation given by the hinge loss:

$$L(w; x, y) := \max\{0, 1 - yw^T x\}.$$

Since our features are sparse, we apply PCA (Principle Component Analysis) to the feature vectors before training the SVM model to reduce dimensions and condense information. PCA is a statistical method to find a rotation such that the first coordinate has the largest variance possible, and each succeeding coordinate in turn has the largest variance possible. The columns of the rotation matrix are called principal components. Here we retained the first 100 components and use them to produce new feature representations in R^{100}.

As a whole picture, for each interest category, we apply PCA to feature vectors that have a label value of that category, and train a SVM model using the PCA transformed feature vectors.

B. Multinomial Näive Bayes

In order to predict if a user is also interested in a new field and will click on its ads, we employ the Multinomial Näive Bayes algorithm to build a classification model.

We apply the Näive Bayes classification algorithm following the primary hypothesis that a user's personal interest in ads under label x is independent from his or her personal interest in ads under label y. In algebraic expressions, if we label variable representing the personal interest of user i to an ad under label j as $\mathbb{I}_{i,j}$, then $\mathbb{I}_{i,x} \perp\!\!\!\perp \mathbb{I}_{i,y}$, where x and y are in the range of labels. We adopt Multinomial Näive Bayes since we have ads under nearly 400 labels, and for each pair of user i and label j the interest value $\mathbb{I}_{i,j}$ is guaranteed to be an integer.

Therefore, a model to classify the user's interest in ads under label y is trained on users' interest on all other labels. The goal of training is to derive $\theta_y = (\theta_{y,1}, \theta_{y,2}, ..., \theta_{y,y-1}, \theta_{y,y+1}, ...)$, in which $\theta_{y,k} := \mathbb{P}(x_k|y)$, x_k is the interest on ads under label k over all users in the training data set. In estimation, $\theta_{y,k} = \frac{N_{yi}+\alpha}{N_{yi}+\alpha \cdot n}$, with $N_{yi} = \sum_{k \in \text{labels}} x_k$, $N_y = \sum_{k \in \text{labels}} N_{yk}$.

In this model, we set the smoothing parameter $\alpha = 1.0$ to use Laplace smoothing in calculating the maximum likelihood of θ_y, given our training data set.

C. Logistic Regression

As you can see from Fig 2, more users chose not to click on an ad given the average feature values and LabelID. Thus, we used logistic regression algorithm from scikit-learn package in python to find the relationship between users' click actions and their interest/engagement to a webpage to improve ad CTR. According to the logistic regression formula: $\log(\frac{Y}{Y-1}) = b_0 + b_1 X_1 + b_2 X_2$, X_1 is LabelID, and X_2 is user's average feature values. X_1 is a categorical variable from 1 to 380, and X_2 is the continuous

values from 0 to 1 since I normalized the average feature values. The reason why I used normalized average feature value instead of feature vector is that there exist outliers among values in feature vectors, so taking the average on feature values and normalizing the values can reduce the affects from outliers. In addition, the feature values represent how users are active at a webpage. Based on our assumption: labels are independent, so we can apply logistic regression model since every observation is independent.

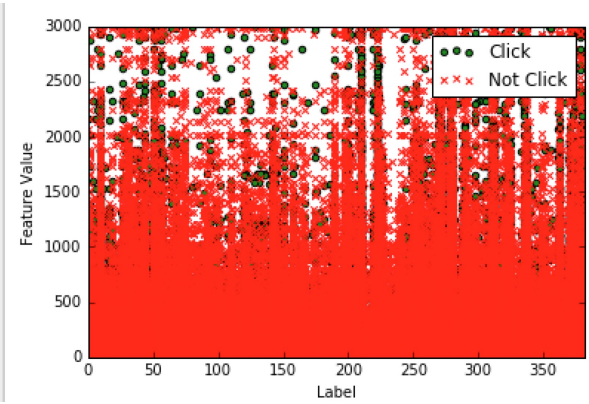

Fig. 3. Label vs. Feature values for training data set.

V. Software Package Description

The package is composed of two part, storage part and modeling part.

The data cleaning and processing pipeline can be found in $generatedata.py$. It can produce various files including labels and features file in LibSVM format, dense format data file, HBase format labels and features file, etc. It can also parse the taxonomy tree file provided in the data set for human readability and modeling convenience. $hbase.py$ will bulk load the data in batch into HBase, provided that the corresponding table has been created.

The file $svm.py$ contains the SVM pipeline as described above. The code of Multinomial Näive Bayes classification model and Logistic regression model pipeline can be found in $model.py$.

VI. Experiment Results

A. Linear SVM

On test data, the average error rate of SVM model is 27.31%, where the average error rate of leaf nodes is 30.85% while the average error rate of top level node (all of them are not leaf) 22.94%. This is

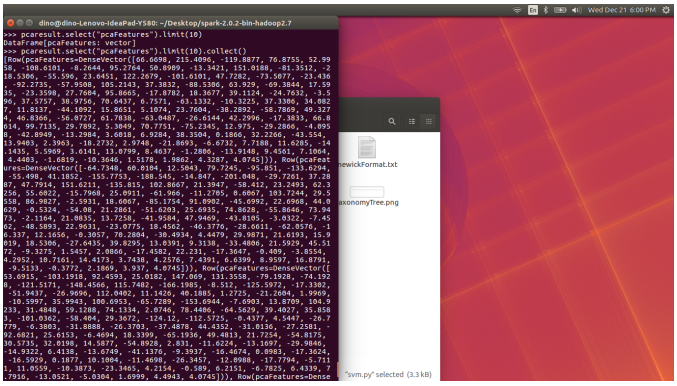

Fig. 4. Some PCA transformed feature of users.

probably the reflection of the fact that positive labels will block and overwrite all negative labels on its way of propagation to the root. So nodes closer to the root will have a lot more chance to have positive value than negative value, therefore easier to predict. The error rates are summarized in the table below.

Type	Avg Error Rate(%)
All	27.31
Leaves	30.85
Top Level	22.94

TABLE I
Performance of SVM model

B. Multinomial Näive Bayes

After a 5-fold cross validation on the training data, we derived a Multinomial Näive Bayes classification model with 25.3% as the testing error, as shown in Table II.

CV Round	Training Error	Testing Error
1	26.1%	25.3%
2	23.9%	
3	21.4%	
4	22.7%	
5	25.4%	

TABLE II
5-fold cross validation training and testing errors using Multinomial Näive Bayes

Given that the smoothing parameter is set to be $\alpha = 1.0$ to apply Laplace smoothing in calculating the maximum likelihood on the training data, the cross validation error rates suggest that the Multinomial Näive Bayes model on the third round yields

the lowest training error (21.4%) on the hold-out subset. Hence, to evaluate the testing error using our best model, we apply the model trained using the first, second, fourth and fifth partition of the training data, as is done in the third cross validation round, on the entire testing data set. The result error rate (25.3%) proves that our model is significantly better than random guessing.

C. Logistic Regression

After a 5-fold cross validation on the training data, we derived a Logistic regression model with 13.3% as the average testing error as shown in Table II. b_1 =-1.5 and b_2=3.5 indicates that a user's click action has a negative relationship with Label ID but a positive relationship with average feature values. The positive relationship between a user's click action and average feature values approved our assumption: a larger average feature value means that a user is more active at a webpage.

CV Round	Training Error	Testing Error
1	15.5%	12.5%
2	12.3%	14.5%
3	16.4%	13.1%
4	14.5%	13.6%
5	17.4%	12.8%

TABLE III
5-FOLD CROSS VALIDATION TRAINING AND TESTING ERRORS USING LOGISTIC REGRESSION MODEL

VII. CONCLUSION

By comparing the predicted values to known target values in a set of test data, our test errors for linear SVM, Näive Bayes and logistic regression algorithm are acceptable (i.e 0.273, 0.253 and 0.133). Näive Bayes can accurately learn users unexplored interests by giving their interests so that we can target potential qualified users. Also, We developed a logistic regression model to predict whether a user will click on an ad if we know his/her interests and feature values (i.e. engagement). Therefore, we can efficiently target right users. For future work, we hope that we can access the demographic data to improve our models. Since in order to protect users privacy, all user identifiers were removed. Thus, demographic data is not accessible. Demographic data contains general information about groups of people such as age, gender, and place of residence, as well as social characteristics such as occupation, family status, or income. It can provide a deeper insight into a websites target users to give a more accurate result. For example, users from different ages may be interested in different ads. Therefore, we can add more variables (e.g. age, gender, income) to logistic regression to explore whether any demographic variables affect ad CTR.

Contributions of each team member in percentage: each member have contributed the same amount of work (1/3).

ACKNOWLEDGMENT

The authors would like to thank heartily to Prof. Ching-Yung Lin for his instruction and encouragement throughout the course.

The authors are also very grateful to the teaching assistants, including but not limited to Eric Johnson (*efj2106*), Munan Cheng (*munan.cheng*), Kushwanth Shantharam (*kk3098*), Rohan Kulkarni (*rohan.kulkarni*), Gautam Sihag (*gautam.sihag*), Peiran Zhou (*pz2210*), Emily Yao (*dy2307*), Chuwen Xu (*cx2178*), for their helpful advice and comments on homework and this project.

REFERENCES

[1] Y. Hu, J. Shin, Z. Tang, *Performance-based Pricing Models in Online Advertising: Cost per Click versus Cost per Action*, Yale School of Management, Oct 2013.

[2] Yahoo! Research, *Yahoo Data Targeting User Modeling, Version 1.0 (Hosted on AWS)*, Retrieved from https://webscope.sandbox.yahoo.com/catalog.php?datatype=a&did=78.

[3] C. Perlich, B. Dalessandro, O. Stitelman, T. Raeder, F. Provost, *Machine learning for targeted display advertising: Transfer learning in action*, Dstillery, 2014.

[4] S. Balakrishnan, S. Chopra, I. D. Melamed, *The Business Next Door: Click-Through Rate Modeling for Local Search*, AT&T Labs-Research, Machine Learning in Online Advertising, Dec 2010.

[5] M. Aly, A. Hatch, V. Josifovski, V. K. Narayanan, *Web-Scale User Modeling for Targeting*, Yahoo! Research, Lyon, France, Apr 2012.

[6] B. Zhao, Y.K. Li, J. C.S. Lui, *Mathematical Modeling of Advertisement and Influence Spread in Social Networks*, NetEcon, 2009.