Web Economics: Group 09 Report

Devin Kuokka
Affiliate Student, Computer Science, University College London
devin.kuokka.16@ucl.ac.uk

Stylianos Rousoglou
Affiliate Student, Computer Science, University College London
stylianos.rousoglou.16@ucl.ac.uk

Michael Whitman
Affiliate Student, Computer Science, University College London
zcabwhi@ucl.ac.uk

1. INTRODUCTION

Real-Time Bidding (RTB) is an increasingly popular approach to online advertising that has evolved into a multi-billion dollar industry. An efficient approach to quickly and accurately optimizing an advertiser's bidding strategy in ad auctions is paramount to the success of the advertising party, as a good solution can save vast amounts of money and lead to higher conversion rates than competitors.

The challenge in automating ad auctions and bidding in real-time is the necessity to predict, with the highest possible accuracy, the likelihood that the ads displayed to a particular user will actually be of interest to them. Using those predictions, hereby referred to as pCTR (or predicted Click-Through Rate), a strategy can be developed to optimize the number of won impressions (which the advertiser pays for) that have the potential of becoming conversions. Naturally, the accuracy and sensitivity of the pCTR values in central in developing a successful bidding strategy.

Given real-life ad impression data, a machine learning approach is often preferred in tackling this problem. First, a large data set with *training* data is used to build and optimize some type of machine learning classifier. After the machine learning model is trained, it is used to make predictions about data in a *test* data set, for which user feedback is not provided, thus attempting to predict unknown users' behavior. The success of the classifier is evaluated by different metrics that reflect its accuracy in anticipating an unknown user's responsiveness to an ad. The machine learning approach undertaken in my personal strategy is *logistic regression*.

Therefore, in attempting to predict user behavior, a thorough and educated study and use of the training data is required for accurate and useful results. Our individual reports present some data analytics that result from a basic exploration of the training data set provided for the assignment.

In this paper, different approaches to automating the generation of bid prices will be undertaken, and each will be evaluated using common evaluation metrics and comparatively to others. First, two naive strategies with little optimisation will be implemented, and the results will be presented and discussed. Subsequently, two machine learning approaches will be presented, one using a linear model and one using a non-linear one. Our approach and steps in building our machine learning classifier and developing a bidding strategy will be detailed. The techniques employed throughout our group strategies will be discussed, and different design decisions made will be defended. Finally, the results of our best strategy will be presented and commented on using relevant evaluation metrics.

2. RELATED WORK

The extensive literature review we performed covers a range of academic papers and articles. The first academic paper, *Real-time bidding benchmarking with ipinyou dataset*, was used as a model for both the presentation of the data and the preliminary statistical analysis performed on the dataset, both to be found in our individual reports. The second relevant publication, *A logistic Regression Approach to Ad Click Prediction*, provided valuable insights into building a logistic regression classifier, which is the machine learning model used in our group solution, as well as into techniques for data pre-processing, data clensing and data reduction. Other related work submitted in the past 5 years to KDD (Knowledge Discovery and Data Mining), a community producing scientific work in data mining and data analysis, was also reviewed for additional observations and expertise.

3. DATASET

The dataset consists of three data files, namely the training, validation, and test files. Machine learning classifiers are first trained using the training data set, *train.csv*. Subsequently, the *validation.csv* file is used to evaluate the performance of different classifiers and suggest which algorithm is more accurate in its predictions. While the aforementioned files include user feedback information about each ad impression, which is necessary for supervised learning as it is used to train, optimize and evaluate the classifier (in the case of RTB, user feedback refers to whether the ad was clicked or not, i.e. a binary variable), the third file, *test.csv*, is only used for testing the developed model, and thus does not contain the *click* data field. Auction pricing information is also not included in the testing dataset, as the solution is required to optimize bidding prices and come up with the best possible strategy to win as many likely-to-be-clicked impression auctions as possible, thus maximizing the conversion rate of ad impressions.

3.1 Data format

The dataset includes thousands of impressions, one per line, in comma-separated csv files, with detailed information about the advertiser, the advertising context, the spatial and temporal context, and the user, for each impression. The format of the data varies across fields, with integers, strings, as well as special words and symbols used. In instances where data is missing, the word *null* is used. Fields such

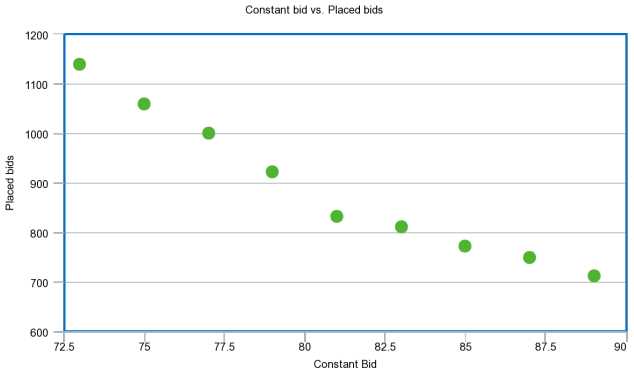


Figure 1: Constant bid vs. Placed bids

as *weekday* and *hour* have been mapped to integers, while others, such as *useragent*, have been left as string. It is clear that some data manipulation has to be performed prior to the data beind used for training a classification model.

Note that, although the dataset uses the currency RMB and units of Fen x 1000 in all figures pertaining to money, all monetary results in this report (such as cost, effective CPC, etc.) have been adjusted to units of Fen.

4. APPROACH & EVALUATION

There were several non-trivial approaches our group experimented with, including linear and non-linear solutions. Below, the constant and random bidding strategies are first discussed, and then a linear and a non-linear approach are presented.

Python was used for developing the main functionality of all the solutions and learning algorithms, as well as for common helper code used for loading the data, storing the output in a convenient format, and displaying a summary of the results in the terminal. The *scikit-learn* [1] Python library was used for the actual logistic regression functionality. Other Python libraries, such as *pandas*[2], a data analysis software library, and *numpy*[3], a scientific computing software package, were also utilized to facilitate data analysis and manipulation.

4.1 Basic Strategies

4.1.1 Constant Bid

The first strategy was a rather simple one; the solution had to bid the same constant value in every auction. Given the budget constraint of 25,000, the task can be treated as an optimization problem with the following limiting cases: the algorithm's bids are too high, which results to all auctions being won and the budget rapidly running out; or the algorithm's bids are too low, which results to too few auctions being won and the budget not being spent.

Clearly, the optimal solution lies somewhere in between the two limiting cases. This approach would in theory help us win more bids than choosing a constant bid by chance, and assuming that impressions that will be clicked are uni-

[1]http://scikit-learn.org
[2]http://pandas.pydata.org/
[3]http://www.numpy.org/

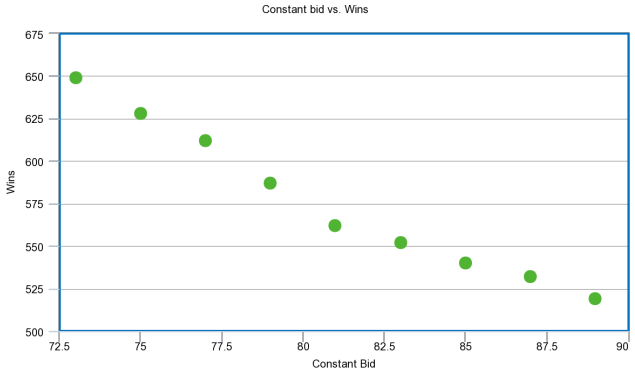
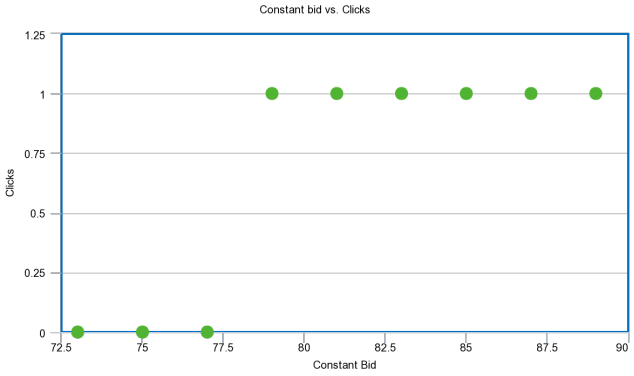
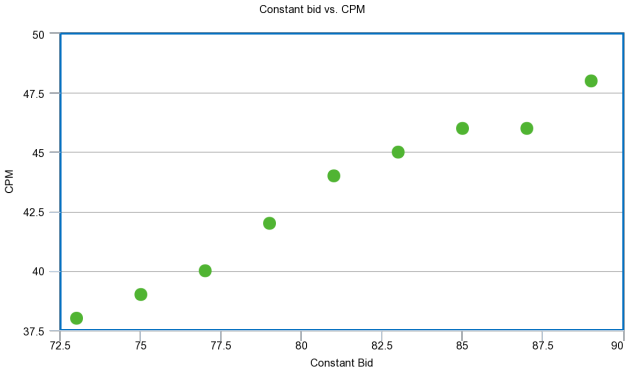


Figure 2: Constant bid vs. Wins



h

Figure 3: Constant bid vs. Clicks



h

Figure 4: Constant bid vs. CPM

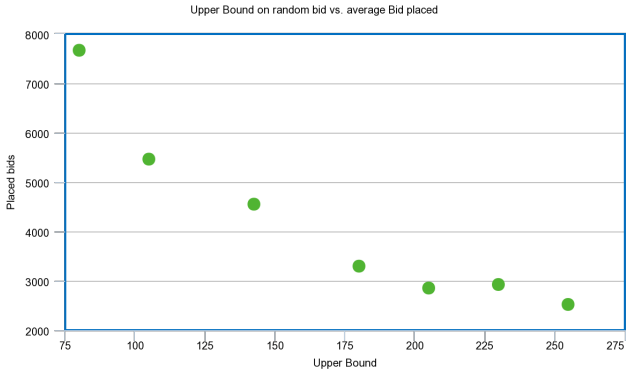


Figure 5: Upper bound vs. Placed bids

formly distributed in the dataset, it also increases the expected CTR, since the budget will all be spent, but not immediately and on consecutive bids. Therefore, we proceeded to calculating the *mean* payprice in the training dataset and using that to define a range of values that we would explore. To approximate the optimal constant bidding amount, we plotted that range of probable values around the mean against several evaluation metrics, with the results presented above.

Figures 1 and 2 show the constant bid value plotted against the bids placed and the bids won by the solution respectively. The results are perfectly consistent with the notion that a high constant bid value places less bids and wins less auctions that a lower one. Additionally, the fact that significant differences in bids and wins are observed for slight variations of the constant bid suggests that the payprice values in the dataset are concentrated around the mean value (which is around 80).

Figure 3 plots the bid value against the number of clicked impressions. As expected, the Constant Bid solution is performing very poorly, and the Click results are not of much insterest since we only see values of one or no clicks at all, which offers no valuable insights on how CTR varies with those changes in the constant bid. For the data points of 1 click, we can conclude that the solution starts consistently winning one patricular auction that leads to a conversion, the price of which is over 77.3, or approximately 79.

Figure 4 shows the bid value against the Cost per million impressions, which expectedly increases since a higher constant bid implies more expensive auctions being won, and thus more money being spent per million impressions.

The CTR is 0% for runs where no impressions were clicked, and about 0.00033% for the latter runs with one clicked impression. This CTR is significantly low, considering that the percentage of clicked impressions in the training dataset is more than 100 times greater, specifically about 0.077%.

4.1.2 *Random Bid*

The second simple strategy we explored introduced randomness in the strategy's bidding choices by basing the bid prices on chance. Our approach in exploring a range of possible upper bounds was the following: after calculating the mean impression pay price from the training dataset, we created a range of values by adding fractions of the standard deviation to the mean. To make our results more reliable, we ran the evaluation process on validation.csv *five times* for each value in the explored upper bound range. The vertical

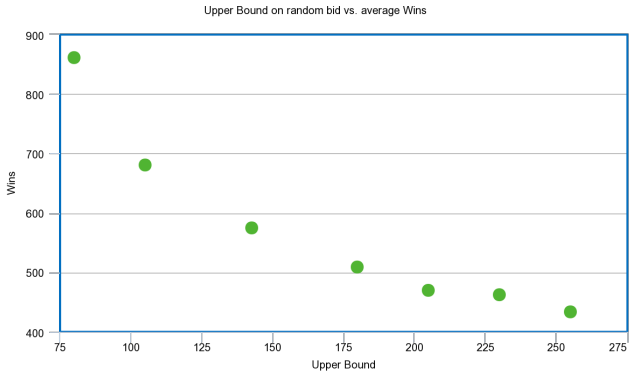


Figure 6: Upper bound vs. Wins

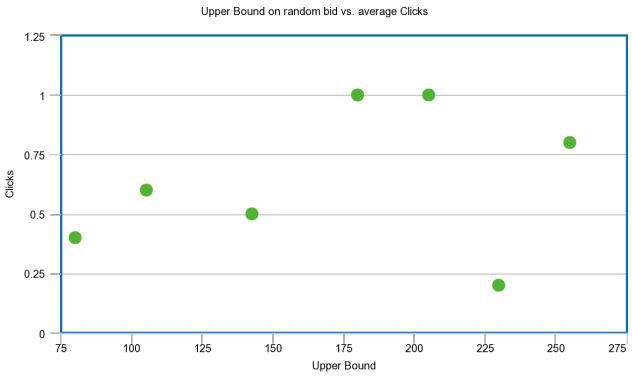


Figure 7: Upper bound vs. Clicks

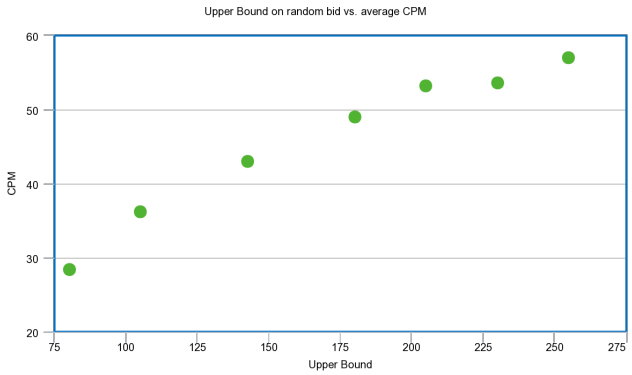


Figure 8: Upper bound vs. CPM

axes then plot the *average* values obtained from five distinct trials with each given upper bound.

To enforce the upper bounds, we used a Python function to generate pseudo-random numbers in the range $[0, 1]$, and then multiplied them by our upper bounds, thus getting a series of pseudo-randomly generated real values in the range $[0, UpperBound]$.

Figures 5 and 6 show the number of placed bids and wins respectively, as a function of the upper bound fed into our pseudo-random number generator. As expected, increasing the upper bound of the random bids leads to an increase in the expected average number of bids placed. Therefore, the trend observed is similar to that of Figure 1, where the actual average number of bids was inveresely proportional to the value of the constant bid.

Figure 7, which plots the upper bound against the number of clicked impressions, reveals why a random-based approach to solving the problem is bound to not perform reliably well. Although repeating the experiment infinite times would give expected values that follow a certain increasing trend, five trials were clearly not enough to demonstrate any consistent trend or correlation between the two values. Since the bidding values are randomly chosen, there is no guarantee on the performance of the solution on any given run.

Figure 8 plots the solution's CPM as a function of the upper bound of our pseudo-randomly generated bid values. The trend observed is similar to the trend in Figure 4, which makes intuitive sense since increasing the expected average bid value should lead to more expensive auctions being won and paid for.

The calculated average CTR of this solution was again about 0.00033%, reflecting the inadequacy of this solution and its bad results, which are similar to those of the constant bid approach.

4.2 Machine Learning Strategies

Before training the machine learning classifiers to make CTR predictions, there was a significant amount of work to be done in order to prepare the data for "learning". First, a high-level analysis (presented in detail in our individual reports) was useful in deciding on the good data features to be used for classification, i.e. the ones that do correlate with differences in the Click-Through Rate. Accordingly, several data fields that either had unique or highly differentiated values, such as the *bidid* and the *userid*, would not have been useful for classification and thus had to be removed. Therefore, some **data selection** was performed to remove those fields, as well as *IP* (unique to each user), *logtype* (1 in this dataset), *domain*, *url*, *urlid*, *creative*, *keypage*, etc.

Given the large volume of data with negative user feedback (click = 0) and the scarcity of impressions that led to conversions (click = 1), **undersampling** was also used to increase the ratio of clicked to non-clicked impressions in the training data. The small bias towards clicked impressions in training helped correct the bias in the original dataset, where only 0.077% of all impressions were clicked.

Several regression and classification machine learning algorithms were considered by our team. The problem lends itself well for supervised learning; the training dataset does contain auction results as well as user feedback, and the goal is to build a classifier that estimates the CTR of any given impression. Our initial instinct was to use Linear Regression, a quite familiar and conceptually simple approach. Linear Regression uses the general linear function $y = a_0 + \sum a_i x_i$, where the dependent variable y is *continuous*, as is *usually* the case for the independent variables x_i as well. It is tempting to use the linear model result as a probability, but there is a fundamental problem with that: the output of the linear model *can be less than 0 or greater than 1*, rendering the output meaningless.

To solve this problem, *Logistic Regression* was introduced, with a mathematical model that actually restricts the dependent variable value in the range $[0, 1]$. In reality, the model is estimating *the probability* of a categorical outcome, i.e. the likelihood of one discrete output result versus another. The general Logistic Regression formula for two outcomes (0 or 1) can be written as

$$P(Y = 1) = \frac{1}{1 + e^{-(a_0 + \sum a_i x_i)}}$$

Though the independent variables may either be continuous or discrete, the result of the model is a probability of the dependent output taking the given discrete value.[4]

Logistic Regression was the approach finally undertaken by our group. Logistic regression is a *binary classification* algorithm, which means that the result of its predictions is discrete by nature. This caters perfectly to our problem, namely deciding whether a user will click (click = 1) an ad or not (no click = 0). Also, all impression data used for learning is categorical, which is acceptable since predictor variables in Logistic Regression dependent variables may be both continuous or categorical. The underlying mechanism of Logistic Regression is Maximum Likelihood Estimation (MLE), a widely used statistical model for estimating the values of parameter variables that maximize the likelihood that the output takes the observed value.

[talk about data selection and overfitting]

4.2.1 Linear Bidding Strategy

[TODO]

4.2.2 Non-linear Bidding Strategy

[TODO]

5. CONCLUSION

[TODO]

6. REFERENCES

[1] Xuehua Shen, Jun Wang, Shuai Yuan, and Weinan Zhang. Real-time bidding benchmarking with ipinyou dataset. *arXiv preprint arXiv:1407.7073, 2014.*

[2] Gouthami Kondakindi, Vinit Parakh, Sai Kaushik Ponnekanti, Aswin Rajkumar, and Satakshi Rana. A Logistic Regression Approach to Ad Click Prediction. *Mach Learn Class Project, 2014.*

[3] Cheng Li, Yue Lu, Qiaozhu Mei, Dong Wang, and Sandeep Pandey. Click-through Prediction for Advertising in Twitter Timeline. *University of Michigan, KDD 2015 Sydney.*

[4] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar

[4] https://www.quora.com/What-is-logistic-regression

Hrafnkelsson, Tom Boulos, and Jeremy Kubica. Ad Click Prediction: a View from the Trenches. *Google, Inc. KDD 2013 Chicago.*