Programmatic Buying Bidding Strategies with Win Rate and Winning Price Estimation in Real Time Mobile Advertising

Xiang Li and Devin Guan

Drawbridge Inc, 2121 S. El Camino Real, San Mateo, CA, 94403, USA
{xiang,devin}@drawbrid.ge

Abstract. A major trend in mobile advertising is the emergence of real time bidding (RTB) based marketplaces on the supply side and the corresponding programmatic impression buying on the demand side. In order to acquire the most relevant audience impression at the lowest cost, a demand side player has to accurately estimate the win rate and winning price in the auction, and incorporate that knowledge in its bid. In this paper, we describe our battle-proven techniques of predicting win rate and winning price in RTB, and the corresponding bidding strategies built on top of those predictions. We also reveal the close relationship between the win rate and winning price estimation, and demonstrate how to solve the two problems together. All of our estimation methods are developed with distributed framework and have been applied to billion order numbers of data in real business operation.

Keywords: Mobile advertising, programmatic buy, real time bidding (RTB), win rate estimation, winning price estimation, bidding strategy.

1 Introduction

A recent trend in mobile advertising is the emergence of programmatic buying in real time bidding (RTB) based marketplace, where each advertiser bid on individual impression in real time. Unlike the conventional mediation type marketplace with pre-negotiated fixed clearing cost, the clearing price in the RTB marketplaces depends upon the bid that each advertiser submits. In second price auction [14], which is commonly used in the mobile advertising RTB marketplace, the clearing price is actually the second highest bid. While at one hand this unique behavior in RTB marketplaces gives advertiser greater flexibility in implementing their own biding strategy based on their own need and best interest, it also makes the market more competitive and demand solutions to some new problems that don't exist in mediation type marketplaces. Some of the specific problems are predicting the win rate given a bid, estimating the most likely clearing price in the second price auction setup, and optimizing the bid based on various estimates. In this paper, we describe our approaches of estimating the win rate and winning price given the bid and correspondingly how do we carry out bidding strategies in different RTB auction setups.

V.S. Tseng et al. (Eds.): PAKDD 2014, Part I, LNAI 8443, pp. 447–460, 2014.

The general workflow in a RTB marketplace is the following: As an end-user launches a mobile app, which we call *"property"*, a request will be sent to the ad-exchange or marketplace, and it will be further passed to all the bidders who trade in that ad-exchange. Those bidders get the information regarding the property, the available ad space to display on the property, and some basic information regarding the device, e.g. device type, os version etc., and they need to decide if they want to place a bid for this particular request, and if so, how much to bid. Once ad-exchange receives all the response from those bidders, it will pick the winner with the highest bid, determine the cost and notify the winner. As an analogy to financial market, an ad-exchange is on the "sell" or "supply" side, and a bidder is on the "buy" or "demand" side.

Win rate estimation refers to the problem of estimating the likelihood of winning an incoming auction request given a specific bid price. It imposes different level of difficulties to the ad-exchange on the sell side and the bidder on the buy side. For the ad-exchange, it has the complete observations regarding all the bids that it receive from all the participants in the auction and it can hereby construct a win rate estimate fair easily by taking a histogram on the winning bid. The task of win rate estimation is more challenging to a bidder in the RTB, in the sense that he only knows his own bid and the outcome of the auction, and he has no idea about other people's bid. In other words, there is a missing attribute situation to the bidder in RTB. Being able to utilize a huge quantity of data with partial missing attribute is the key to a successful demand side win rate estimation.

Winning price estimation is another unique problem in the second price auction type RTB, and is indeed an even more challenging problem to the bidder. In most of the RTB exchanges, a bidder will only know the winning price information if his bid win the auction. Most RTB exchanges, especially those with big volume, are highly competitive in terms of number of bidders participating in the auction, and that consequently push down the win rate for each individual bidder. For each bidder in the auction, his average win rate in general decreases as more competitors join the auction, and such decrease of the win rate further reduces the amount of positive data available to him for the winning price prediction and that makes his estimation more difficult, which could further reduces his win rate. Such vicious circle of win rate and winning price makes the winning price prediction a critical component in the RTB auction.

Despite all the above challenges on the win rate and winning price prediction tasks, there is one good news regarding those two problems, in the sense that those two are closely related to each other. With some statistical transformation, the solution to the one problem can be automatically applied to the other. Specifically, we will illustrate in section 4 that win rate is essentially the cumulative distribution function (cdf) of the corresponding winning price distribution. Solutions of one problem bring the solutions to the other. In this paper we choose to approach the problem from the win rate side, first model the win rate using a logistic regression model, and then take the derivative of win rate estimation to generate the distribution of the winning price, and use the expected value of the distribution under the bid price as the winning price estimate.

While win rate and winning price model enables a RTB bidder to predict the likelihood of success and the associated cost, the actual programmatic bidding has to be done through a bidding strategy. A bidding strategy is actually an optimization function, takes the input of expected revenue if winning the auction (i.e. ecpm estimate), win rate and winning price estimate, and generate the final bid price according

to some pre-defined objective functions. Some specific strategies could be to maximize the actual revenue or the profit from the operation, or the combination of both revenue and profit.

The paper will be organized as the following. Section 2 reviews some of the existing works in the field. In section 3 and 4 we describe our effort of estimating the win rate and the winning price given a bid, respectively; and in section 5 we explain the various bidding strategies that we have tested. The experimental setup and results are illustrated in section 6, and we conclude in section 7 with our contributions.

2 Related Work

Our win rate estimation techniques are based on logistic regression methods, and there are some existing works of estimating various probabilistic events in advertising using logistic regression models. For example, there are those of using logistic regression models to estimate click through rate [9][10][12], television audience retention [11], contextual effect on click rates [12] and bounce rate in sponsored search [13] etc. To estimate the winning price in auction, there are in general two high level approaches: machine learning based approaches as in [3][5][6] and historical observation or simulation based approaches [5-8]. R. Schapire et al. [3] used boosting based conditional density estimation in Tac-2001 competition [4]. They treated the price estimation as a classification problem, discretized the price into different buckets, and used boosting approach to estimate the selling price. For bidding strategies, many existing work [3-8] are closely related to the corresponding winning price estimation tasks, mostly due to the fact that they are motivated by series of TAC competitions [4].

Despite all those existing work, we see very few publications regarding the estimation and bidding strategies in the setup that we are facing in our everyday business operations. For example, many of the existing winning price estimation work are based on the assumption that a buyer has complete observation regarding the past auction outcome [3], while this assumption is apparently not valid in our daily operation.

3 Win Rate Estimation

3.1 Logistic Regression Based Win Rate Estimation and Corresponding Features

The likelihood of winning an auction given a bid price depends on two high level factors. One is the characteristics of the incoming request, and the other is the bid that a bidder willing to pay. Not all the incoming requests are with the same value, and there are many different attributes that affect the quality of each incoming request. For two requests with the same bid price, the win rate could be drastically different depending on the attributes like the nature of the mobile app (property), the time of the request, the size of the available ad-space, the geo location of the user, and many other attributes. The second factor that affects our win rate is apparently the bid price itself: for the same request, the higher our bid is, the more likely we will win the auction. Those two factors have to be included in the win rate estimation as features.

As a probability term bounded between 0 and 1, win rate makes itself a perfect candidate of using logistic regression model [10], as in Eq. 1:

$$winRate = \frac{1}{1+e^{-z}},\ where\ z = \beta + \sum_{i=0}^{n} \theta_i * x_i \quad (1)$$

, where *winRate* is the estimated win rate, β is the intercept, x_i is individual feature extracted from the request, and θ_i is the corresponding model weight.

With no surprise, the list of the features that we have constructed resonates with the two high level factors we mentioned above. We extracted various features regarding the nature of the request, our bid, and we combine them to make the win rate prediction. Some of the features are *"stand-alone"* features that describe single attribute of the request that we received, e.g. the name of the app; and others are what we called*"cross"* features that describe the inter-action between individual *"stand-alone"* features. For example, we have a *"cross"* feature to describe mobile app name and day of the week that we receive the request. With *"stand-alone"* and *"cross"* features all together, there are about 1 Million total features in our win rate model.

3.2 Scaling Up Win Rate Prediction with Distributed Machine Learning

One of the challenges we are facing with win rate prediction, as with many other machine learning tasks, is the scale of the data that is available to us. The number of daily requests that we receive from the ad-exchanges is at the order of billions, and literally for every request that we submit our bid, it can be used as training data for our win rate prediction model, either as negative or positive data, depending on if our bid win the auction or not. With such huge amount of available data, it will be a crime to down-sample and use only a small percentage of the data in order to fit into existing non-distributed machine learning toolkit. Instead, we choose to utilize as much of the data as possible to build our model, and as we will illustrate in section 6, more data does help on the prediction accuracy.

Our approach of utilizing such huge amount of data is to use distributed machine learning algorithm and toolkit, for example Mahout[1] and Vowpal Wabbit(VW) packages[2]. We have tried both packages, and we end up using the VW. VW is more specialized in the classification tasks using general linear learner, while Mahout focuses more on the recommendation, clustering and general machine learning tasks. From our own observations, VW is faster than Mahout, especially for the large-scale sparse training data that we have used. On average our win rate model utilizes about 1 or 2 billion of records as training data for each model update, and the training process can be finished in couple of hours using VW. VW is faster because it uses true parallel processing with message passing interface, while Mahout is built on top of the MapReduce framework.

3.3 Feature Selection, Regularization and Missing Feature in Win Rate Prediction

While the distributed machine learning algorithm and tools give us the capability of utilizing the huge amount of available data, we still need to answer some of the

traditional machine learning questions before we can build an accurate prediction model. The first question is how to deal with large number of features that can be extracted from the available data. As mentioned earlier, the original number of unique features is in the order of million, and we need to avoid directly feeding all those features into model building process. Typically there are two high level approaches to address this high feature dimension problem: Feature selection and regularization. Feature selection techniques can effectively reduce the number of unique features before model building actually take places, but it has to be done off-line first and it's quite expensive. On the other hand, regularization techniques mix the feature selection process with model building process and are more efficient than the separate feature selection approaches. Within regularization, L1 regularization automatically decides the feature to get non-zero value during the model building process, and L2 regularization can put more emphasis into more discriminant features. In our case, we decided to do regularization directly during model training without a separate feature selection process.

The second question we need to answer is regarding the new attributes. Regardless of how frequently do we update the model, and even if we use online model updating process, it's still quite often that we will see new attribute values pop up from the request, and this is especially true in large ad-exchanges since they keeps on adding new mobile apps to their inventory. Almost every week, we see new app become available for bidding, and all the features associated with that new app will become undefined. This significantly affects the accuracy of our prediction model.

Our solution to address this new attribute problem is to add *"filler"* feature into the model building process. During the model training, we will remove certain features and replace those removed features with corresponding *"filler"* features to build model. When we make predictions for new attribute values, e.g. a new mobile app (property) name, we just use *"filler"* feature to represent the value and generate win rate estimate.

4 Winning Price Estimation

The cost in an auction depends on the format of the auction. In the first price auction, winning price is exactly the same as the bid. In the second price auction [14], winning price is the second highest bid that the ad-exchange receives. Due to business constrains, ad-exchange only notify the bidder what the winning price is if the bidder actually win the auction. If a bidder loses the auction, all the information he would know is that the winning price is at least the same as his bid, since otherwise he would win the auction. Also, as we mentioned earlier, because there are many bidders in some of the largest mobile ad-exchanges, the typical win rate for a bidder in such large mobile ad-exchange is in the order of single digit. All the above factors translate into two problems for machine learning based winning price estimation: 1) unbalanced distribution among positive and negative training data, and 2) missing value issue in the negative training data.

4.1 Linear Regression Based Winning Price Prediction

Maybe the simplest way of estimating the winning price is to use linear regression. Specifically, we could extract features from all the auctions that we have won, and try to fit a linear regression function as in Eq. (2). This approach has immediate problems in the followings three ways: 1) it assumes the relationship between winning price and all the features, including price, to be linear, which is hardly satisfied. 2) It only uses the positive data, the observed winning price to make prediction, and throw away all the negative data for the cases that we didn't win. While those negative data point don't contain as valuable information as the positive data point, they still provide us some information, e.g. the winning price is at least the same as our bid. 3) It's a deterministic process, for the request with same attribute values and same bid, it will always return one fixed winning price. In reality, winning price is determined based on the behavior of our competitors. Behavior change of our competitor causes the winning price to fluctuate, and we want a way to model this fluctuation if possible. Due to those limitations of linear regression, it was no surprise to see that it didn't perform well in winning price estimation task.

$$winningPrice = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{2}$$

4.2 Solving Win Rate and Winning Price Estimation Problems All Together

Fortunately, there is one good thing about winning price estimation, in the sense that it's not a separate problem and it actually relates to the win rate estimation problem. Imaging that we have a probability density function of the winning price distribution, as in the Fig. 1, then for every bid *b* as the vertical line in the Fig. 1, the probability of winning corresponds to the situation where the winning price is less than or equal to the bid *b*, which is the marked area on the left side of the bid line.

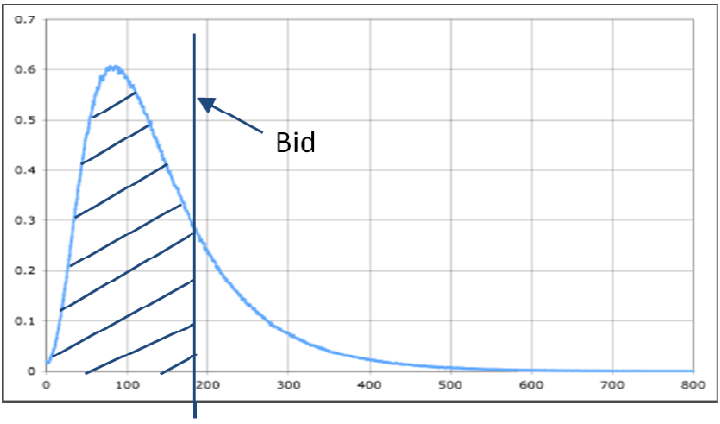

Fig. 1. Win rate given a bid can be calculated from the winning price distribution

In other words, for each bid *b,* the win rate is the same as this probability

$$P(winningPrice \leq b) \tag{3}$$

, which is the cumulative distribution function (cdf) of the probability density function (pdf) of the winning price. In other words, if we have a winning price distribution, and we take the integral up to the bid price, we will have the win rate; if we have the win rate distribution and we take the derivative with respective to the price, we have the winning price distribution. They are dual problems that can be solved with one uniform solution. Assuming the win rate estimation is based on logistic regression as in Eq. (1), we can re-format the Eq. (1) as Eq. (4)

$$winRate = \frac{1}{1+C*e^{-\theta_b*bid}} \tag{4}$$

C is a constant factor that covers the exponential term of all the features that are unrelated to the bid price, and $\theta_b$represents the model weights associated with bid price.

If we take the derivative of Eq. (4) with respect to the bid price, we will get:

$$\frac{d(winRate)}{d(bid)} = \frac{1}{(1+C*e^{-\theta_b*bid})^2} \cdot C * e^{-\theta_b*bid} * \theta_b \tag{5}$$

, and that is exactly the probability density function (pdf) of the winning price distribution given all the attributes of an incoming request.

$$f(x) = \frac{1}{(1+C*e^{-\theta_b*x})^2} \cdot C * e^{-\theta_b*x} * \theta_b \tag{6}$$

Having the closed form solution on the winning price distribution, we compute the actual winning price for each given bid by taking numerical approximation of the mean value in the region left to the bid *b* as winning price, as in Eq. (7). *f(x)* is the pdf of winning price distribution as in Eq. (6), and *E{}* is the expected value of a distribution.

$$winningPrice = E\{x * f(x)\}, where\ x \leq b \tag{7}$$

5 Bidding Strategy

Having the estimate of win rate and the winning price, we can derive our bidding strategy to automatically calculate the bid for each incoming request. Bidding strategy is indeed an optimization function, it takes as input the monetization capability on the individual request, the win rate estimation function and the corresponding winning price estimation for each bid, and produces the final bid based on specific business objectives as output. In the remaining of this section, we will illustrate some of the different bidding strategies that we use to drive different business objectives. For the annotation, we use *pRev* and *eRev* to represent the expected revenue given that we win the auction and expected revenue when we places the bid (those two revenue terms are different), *winRate(bid)* as win rate estimate for the specific bid, and *cost(bid)* as the cost for winning the auction. *pRev* is the "effective cost per-thousand

impression" (CPM), i.e., what we charge our client for serving thousand impressions for them; *cost(bid)* is either the winning price in the second price auction, as discussed in section 4, or the bid price itself in the first price auction. Among those terms, *pRev* is a constant term independent of how do we construct the bid, *eRev, cost(bid)* and *winRate(bid)* are all monotonically non-decreasing functions of the bid price.

5.1 Strategy That Maximizes the Revenue

The first specific strategy is to maximize the revenue. If we assume that the term *pRev* is accurate, then whether or not we can realize this revenue *pRev* solely depends on if we could win the auction and serve the ad. In other words, the expected revenue for each incoming request *eRev* can be formulated as:

$$eRev = winRate(bid) * pRev \tag{8}$$

The corresponding bidding strategy that maximizes the revenue can then be formulated as:

$$bid^* = argmax\{winRate(bid) * pRev\} \tag{9}$$

Since *winRate(bid)* is a monotonically non-decreasing function with respect to the bid, the *bid** that maximize the Eq. (9) is simply the one that maximizes the win rate, which is *pRev*.

Bid with *pRev* is the optimal revenue generating strategy in the first price auction scenario, as we pay what we bid and we can maximize our bid to the extent that we don't lose money. On the other hand, most of the RTB ad-exchanges that we participate operate on the second price auction mechanism [14], bid with *pRev* may indeed lose opportunities since the cost of winning most of time is less than our bid, the *pRev*. In this case, we would bid at a higher bid price whose corresponding winning price is the same as the *pRev*. This can be described as in Eq. (10):

$$bid^* = argmax\{winRate(bid) * pRev\}, where\ argWinning(bid) \leq pRev \tag{10}$$

argWinning(x) is a function that returns the bid whose corresponding winning price is the input price point *x*. We can use a linear search algorithm to approximate this function.

5.2 Strategy That Maximizes the Profit

While the strategy in section 5.1 maximizes the revenue, the daily operation of the business is to make profit, and it quite often that we want to have a bidding strategy that maximizes the profit produced from the business operation. Unlike the revenue, which is a monotonically increasing function with respect to our bid, profit depends on the difference between the revenue and the cost. *pRev* is a constant term, but the cost increases with our bid. At one hand, if we win the auction, the higher our bid the less profit we would realize; on the other hand, the higher our bid the higher likelihood of us winning the auction, and if we don't win the auction, the cost and revenue

will all be 0. The optimal bid is the one that maximizes the joint effect of the two above factors, which can de described as in Eq. (11):

$$bid^* = argmax\{winRate(bid) * (pRev - cost(bid)\} \tag{11}$$

Depends on how the cost is constructed, *cost(bid)* would be either the *bid* as in first price auction or the winning price estimate of the bid as described in section 4.

One thing worth noting is that the strategy of maximizing the profit is different from the one that maximize the profit margin. Regardless whether profit margin is defined as the ratio of revenue minus cost over revenue, or directly the revenue over cost, it all has the fixed revenue term as either the denominator or the numerator, and the bid that maximizes the profit margin will simply be the one that minimize the cost, and that translates into 0 as the bid. While this yields the maximum value of profit margin in theory, we won't be able to realize this margin, since we won't win any auction with 0 bid price. This is another reason why we don't use profit margin as the objective for our bidding.

5.3 Strategy That Maximize the Combined Profit and Revenue Goal

Unlike the previous two bidding strategies that solely focus on either the profit or revenue, we could also combine those two factors together and maximize a combined objective during bidding. Specifically, we can mix the two objective functions together and use a weight alpha to control the blend of two strategies. If the alpha term is applied to the profit based objective function, then we can make the combined strategy as the following:

$$bid^* = argmax\begin{Bmatrix}\alpha * winRate(bid) * (pRev - cost(bid)) \\ +(1-\alpha) * winRate(bid) * pRev\end{Bmatrix} \tag{12}$$

The *winrate(bid)*pRev* term in the profit and revenue strategy component will indeed cancel the effect of alpha, and we will have the mixed bidding strategy as in Eq. (13):

$$bid^* = argmax\{winRate(bid) * (pRev - \alpha * cost(bid))\} \tag{13}$$

This mixed strategy covers both the revenue and profit objective during business operation. We can adjust the alpha value that controls the relative importance of profit v.s. revenue when we bid. When alpha equals to 0, this combined strategy becomes the one that maximize the revenue, as in section 5.1; when alpha becomes 1, this strategy falls back into the profit optimization strategy. In addition to this flexibility, this bidding strategy gives us another advantage in the sense that we can dynamically adjust the value of alpha in real time based on the performances of the bidding system so far. For example, we can set a goal on either the revenue or profit metric, check the progress of the bidding system toward the goal at fixed time intervals, and adjust the alpha value accordingly to hit the pre-defined revenue or profit goal.

6 Experimental Setup and Results

6.1 Evaluation Metrics

We tested the performance of our various estimation methods and bidding strategies using one of the leading mobile ad exchange platforms that we participate. On everyday we bid on more than a few billion requests in that ad exchange.

The performance metrics we used to evaluate our methods are the followings:

- For win rate estimation: We used the log-loss of predicted results collected from bidding data.
- For winning price estimation: We used two metrics, RMSE and ratio-RMSE. RMSE is computed based on predicted winning price and actual observed winning price; and ratio-RMSE is computed by first taking the ratio of predicted winning price over observed price, then compare it against value 1 and compute the corresponding RMSE. ratio-RMSE was introduced to offset the scale difference between winning prices. For the same 1 cent difference between the predicted and actual winning prices, the level of accuracy is completely different between the case with base price of 10 dollar and the case with base price 10 cents, ratio-RMSE normalizes the level differences between different data points.
- For bidding strategy: We looked at the revenue and profit margin per unit percentage of traffic. We took the baseline revenue and profit margin figures from maximizing revenue strategy as 1, and computed relative metrics on the revenue and profit margin from other bidding strategies.

6.2 Experiment Setup and Results for the Win Rate Estimation

Since we conducted the experiments using domain specific real business operation data, all the numbers reported here were relative performance metrics. Nevertheless, the findings from the experiments are still meaningful, especially for the purpose of comparing different approaches and identifying better modeling techniques.

In the win rate and winning price estimation, we conducted the experiments by splitting the data into training and testing set. Testing set has the data collected from 1 week time period during January 2013. We built models on the training set and tested the model on the testing data, all done in off-line fashion.

Our win rate estimation baseline approach was to simply use the historical observed win rate. We sliced all the auction winning and auction not winning instances observed in the past time period according to combination of their attributes. For each slice of data, we first segmented the bid price into discrete chunk, then under each chunk, we collected all the auction winning instances, divided by the total auction instances to produce the win rate for that specific price chunk given the combination of attributes.

We tested the baseline method with two different historical time windows. One was *baseline_7* and the other was *baseline_14*. *baseline_7* uses the past 7 days training data and *baseline_14* covers 2 week period instead.

One issue with our baseline approach is that we won't know the win rate for the new apps (new property) or new ad dimension. We used the fall back approach to handle that. Every time we can't find the historical win rate based on the feature combination look up mentioned above, we fall back to a combination with one less factor and check if we could find the win rate, and continue to fall back if the historical win rate is still missing.

The first experiment with our logistic regression model was to use the *"standalone"* features only. We built the win rate model using 7 days and 14 days sliding window historical data, and the performance is labeled as *logistic_standalone_7* and *logistic_standalone_14* respectively.

We then tested the performance of adding *"cross"* features into win rate estimation. The performances were summarized as *"logistic_all_7"* and *"logistic_all_14"* respectively.

Table 1 lists the performance metrics of all the win rate estimation methods.

From table 1, it's clear that our win rate model performs better than baseline. The best performing model is the all feature model using 14 days of historical data. Indeed, it almost reduced the log-loss of *baseline_7* by half. All the log-loss results were statistically significant.

Table 1. Performance comparison between various win rate estimation methods

Win rate estimation method	Log-loss
baseline_7	0.156
baseline_14	0.142
logistic_standalone_7	0.118
logistic_standalone_14	0.109
logistic_all_7	0.091
logistic_all_14	0.087

6.3 Experiment Setup and Results for the Winning Price Estimation

Experimental results for winning price estimation are listed in table 2.

The baseline results was achieved by slicing the historical data of winning price based on the same way that we sliced the data to get the baseline results for the win rate. We also applied the same fall back logic if there was any attribute value missing. Results are labeled as *price_baseline_7* and *price_baseline_14* in table 2.

The second result sets were based on linear regression approaches. The features used in linear regression winning price model were the same as those used in *logistic_all_7* (or *logistic_all_14*), and the linear regression coefficients were computed using 7 and 14 days of data as well. *linear_all_7* and *linear_all_14* in table 2 are the corresponding results.

The third set of results, *logistic_price,* came from logistic regression based methods. We calculated the expected value of winning price based on the distribution from the price point of 0 all the way up to the bid price, and used it as the winning price. *logistic_price_7* was generated using the 7 days of training data, and *logistic_prce_14* was generated using 14 days of training data.

Two baseline approaches using different time window yield very similar results, probably due to the fact the mean value of winning price for each combination didn't change that much from 1 week to 2 weeks time period. *"logistic_price"* based winning price out-performed the other two approaches, but the lift wasn't substantial. We believe that was due to the intrinsic high variance among the winning prices. Naturally, the winning price depends on our competitor's bidding behaviors, and their bidding behaviors can be heavily influenced by their business needs. We see quite often that the winning price for the same segment of traffic changes drastically from one individual request to another, in a very short time period of minute. Indeed, we computed an "oracle" experiment using the mean value of winning price observed per traffic segment in the testing data and measured its performance on the same testing set (in other words, train and test on the same testing set). The RMSE and ratio-RMSE was 22.8 and 0.34 respectively. This "oracle" experiment performance can be treated as the upper bound of all winning price estimation methods.

Table 2. Performance comparison between various winning price estimation methods

Winning price estimation method	RMSE	ratio-RMSE
price_baseline_7	33.76	0.59
price_baseline_14	33.74	0.59
linear_all_7	38.71	0.62
linear_all_14	37.76	0.59
logistic_price_7	31.72	0.51
logistic_price_14	31.57	0.52

6.4 Bidding Strategy Experiment Setup and Results

The evaluation of bidding strategy is slight different from the win rate and winning price. While all win rate and winning price prediction approaches can be evaluated offline using previously collected historical data, for bidding strategy it has to be evaluated in online environment against live traffic data. In our experiment, we created multiple testing buckets with the same percentage of traffic, and let each single bidding strategy drive the bid inside one bucket. We ran multiple of those testing buckets in parallel for one week to offset "day of the week" effect, and compared different testing buckets based on their relative revenue and profit measurements with respect to the strategy that maxsimize the revenue, and the results are summarized in table 3.

Table 3. Performance comparison between various bidding strategies

Bidding strategy	Normalized revenue	Normalized profit margin
Maximizing revenue	1	1
Maximizing profit	0.9	1.35
Combined strategy, alpha=0.3	0.97	1.1
Combined strategy, alpha=0.8	0.95	1.23

It's clear from table 3 that each strategy does what it supposed to do. We obtained the maximum revenue, with the sacrifice on profit margin using the revenue maximizing strategy, and vice verse for profit maximizing strategy.

It's also interesting to compare the relative gain on the revenue and profit margin between different strategies. In general, we observed that it was easier to gain on the profit margin side than to grow the revenue. In our specific example below, in order to grow the revenue by 10% relative from the profit maximizing strategy to the revenue maximizing strategy, we need to sacrifice nearly 35% of the profit margin. This is not a surprise to us. We need to bid more to get higher revenue scale, and a higher bid will result in higher cost per unit traffic. In other words, revenue grows linearly with the traffic, while cost grows faster than linear with the traffic scale.

7 Conclusions

We described our effort of estimating win rate, winning price and corresponding bidding strategies in real time bidding (RTB) based ad-exchange in mobile advertising. We explained our effort of building large scale logistic regression based win rate model, which is capable of handing order of billions real data and provide accurate win rate estimation. We have also demonstrated the dual relationship between win rate and winning price in the second price auction scenario, revealed that the two problems can be solved with one solution, and proposed a corresponidng winning price estimation method. Based upon the win rate and winning price estiamtion, we outlined various bidding strategies that we used in our daily operation. Comparison data from real business operation confimed the superiority of our proposed methods against various baseline approaches.

References

1. Apache software foundation, Scalable machine learning and data mining (2013), http://mahout.apache.org
2. Langford, J.: Wowpal wabbit (2013), https://github.com/JohnLangford/vowpal_wabbit/wiki
3. Schapire, R., Stone, P., McAllester, D., Littman, M., Csirik, J.: Modeling auction price uncertainty using boosting-based conditional density estimation. In: ICML 2002 (2002)
4. Wellman, M., Greenwald, A., Stone, P., Wurman, P.: The 2001 trading agent competition. Electronic Markets 13, 4–12 (2003)
5. Wellman, M., Reeves, D., Lochner, K., Vorobeychik, Y.: Price prediction in a trading agent competition. Journal of Artificial Intelligence Research 21, 19–36 (2004)
6. Putchala, R., Morris, V., Kazhanchi, R., Raman, L., Shekhar, S.: kavayaH: A trading agent developed for TAC-02. Tech. Rep., Oracle India (2002)
7. He, M., Jennings, N.: SouthamptonTAC: An adaptive autonomous trading agent. ACM Transactions on Internet Technology 3, 218–235 (2003)
8. Greenwald, A., Lee, S., Naroditskiy, V.: RoxyBot-06: Stochastic prediction and optimization in TAC travel. Journal of Artificial Intelligence Research 36, 513–546 (2009)

9. Hosmer, D., Lemeshow, S.: Applied logistic Regression, 2nd edn. John Wiley and Sons (2000)
10. Richardson, M., Dominowska, E., Ragno, R.: Predicting clicks: estimating the click-through rate for new ads. In: Proceedings of WWW 2007, pp. 521–530 (2007)
11. Interian, Y., Dorai-Raj, S., Naverniouk, I., Opalinski, P., Kaustuv, Zigmond, D.: Ad quality on tv: Predicting television audience retention. In: ADKDD (2009)
12. Becker, H.: Modeling contextual factors of click rates. In: Proceedings of AAAI Conference on Artificial Intelligence, pp. 1310–1315 (2007)
13. Sculley, D., Malkin, R., Basu, S., Bayardo, R.: Predicting bounce rates in sponsored search advertisements. In: SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp. 1325–1334 (2009)
14. Edelman, B., Ostrovsky, M., Schwarz, M., Fudenberg, T., Kaplow, L., Lee, R., Milgrom, P., Niederle, M., Pakes, A.: Internet advertising and the generalized second price auction: selling billions of dollars worth of keywords. American Economic Review 97 (2005)