Multi-Agent Intelligence Group Coursework Report

Haoxuan Wang
ucabhw2@ucl.ac.uk

Wen Yang
wen.yang.18@ucl.ac.uk

Chenhao Lu
ucabcl7@ucl.ac.uk

ABSTRACT

This report described our work in the COMPG0124 Multi-agent Artificial Intelligence Group Coursework, where we tested the performance of different bidding strategies for online Real-Time Bidding (RTB) based on a part of the iPinYou dataset. We firstly reviewed existing researches on RTB strategies and explored the dataset, based on which we developed 3 machine learning (ML) models, using respectively Logistic Regression (LR), Gradient Boosting Regression Tree (GBRT) and multi-layer perceptron neural network (MLP), to predict the probability of an impression to be clicked, i.e., Click-Through-Rate (CTR) Prediction. After that, we compared Constant, Random and Linear strategies, and 3 nonlinear strategies including the Quadratic strategy, Optimal RTB (ORTB) strategy, and the Bidding Machine, on maximizing the total number of clicks received given a limited budget. and found out the combination of GBRT with Quadratic strategy performed best against the original pay prices in both the validation and test splits of the dataset. As a result, we got 185 clicks on criterion 1 of the leaderboard (LB1) and ranked No. 3.

We simulated the online bidding environment, i.e., the competition on criterion 2 of the leaderboard (LB2), where 30 agents could competitively bid for every impression. With it we demonstrated certain phenomenons on the leaderboard, such as the emergence of market segmentation. We also produced the market landscape, located other groups in the market, and explained the stats in LB2. Supported by our understanding of the competition, we achieved 12 clicks on LB2 and ranked No. 5.

Keywords

Real-Time Bidding (RTB), Demand-Side Platform (DSP), Bidding Function, Click-Through-Rate (CTR) Prediction.

1. INTRODUCTION

This report described our work in the COMPG0124 Multi-agent Artificial Intelligence Group Coursework, where we tested the performance of different bidding strategies for online Real-Time Bidding (RTB) based on a part of the iPinYou dataset.

RTB is a popular and fast-growing regime for advertisement (ad) allocation in online advertising. In RTB, the ads in a web page are determined at the time the page being loaded, to maximize the possibility to be clicked by the visitor. When a user try to load a web page that has ad-slots, potential impressions are formulated. Ad Exchange platforms would aggregate some context information about the customer who request the web page. These information such as the page URL, the size and format of ad-slots, and the user's IP address, would be sent to Demand-Side Platforms (DSPs). After receiving a bidding request from AD Exchange platform, the DSP must maximize the total clicks on the ad impression campaign on behalf of its advertisers who have limited budgets. Then after receiving bid prices from DSPs, the Ad Exchange would run a second price auction, and the web page would be loaded with those ads from those winning advertiser.

Our work aimed at developing an optimal bidding system for a DSP in competitive environment. We conducted our work in steps and reported them respectively in the following sections. Section 2 summarized related researches. Section 3 reported the exploratory data analysis of the dataset we used, which was a part of the iPinYou dataset containing 3 million impressions won by iPinYou, the largest DSP in China. Section 4 compared two basic strategies, namely Constant and Random strategies. Section 5 described how we built ML models for predicting CTR, i.e., the probability for an impression to be clicked. And the performance of Linear bidding strategy based on the CTR prediction. Section 6 described the construction of 2 nonlinear CTR prediction models, and compared the performance of 3 nonlinear bidding strategies. In Section 7 we developed a multi-agent simulation of competitive market environment, investigated market landscape and equilibrium, and proposed valuable strategies for such environment. Section 8 is the conclusions.

2. RELATED WORK

A bidding system usually contains two essential steps, a CTR prediction model, and a bidding strategy which takes the estimated CTR and returns the bid price. Moreover, there are also researches on prediction of market prices.

H. Liao et al. who released the iPinYou dataset also provided detailed explanations of each feature [2]. Based on that, W. Zhang et al. conducted benchmark studies on the iPinYou dataset [6], where they did insightful exploratory data analysis, developed Logistic Regression (LR) and Gradient Boosting Regression Tree (GBRT) models for CTR prediction, and compared the performance of Constant, Random and Linear bidding strategies on the dataset. In their study, GBRT had better CTR prediction performance with the area under ROC curve (AUC) score of 0.852. And the combination of Linear strategy and LR prediction obtained the most clicks.

Meanwhile, X. He et al. [1] from Facebook investigated the performance of different CTR prediction models including LR, a hybrid model of LR and Boosting Trees, and Naive Bayes based on Facebook's own RTB data. They found that the hybrid model, where the Boosting Trees acted as a feature encoder for the input of LR, performed best. Additionally, they investigated various affecting factors of CTR prediction accuracy, including data freshness, learning rate of ML models, tree depth and number of trees, and data subsampling rate. In particular, they demonstrated that, given the huge imbalance between positive and negative (i.e., clicks and non-clicks) samples in RTB impression data, downsampling the negative data only could lead to better performance of the CTR prediction model.

W. Zhang et al. [3] proposed an integrated bidding system named Bidding Machine, which combined the CTR prediction, market price prediction and bidding strategy into one mathematical model so that they can be optimized as one entity. To achieve this, they introduced the utility of an impression as the defined value of reward if being clicked minus the cost of winning the impression. By optimizing w.r.t. the utility instead of binary value of click/non-click, the model made more reasonable bids that were closer to the market price and less extreme bids. They found Bidding Machine outperformed simpler systems stacked by separable subsystems, such as in [6].

J. Xu et al. proposed Lift-based CTR prediction method, where they defined Lift as the incremental value to the advertiser for a user seeing an ad given the user's historical behaviour. [4] However, in our dataset very few users appeared more than once, thus this method was not applicable. Instead, we tested the 2 nonlinear Optimal Real-Time Bidding (ORTB) strategies proposed by W. Zhan et al. [5]

3. EXPLORATORY DATA ANALYSIS (EDA)

The dataset we used was split at the ratio of 8:1:1 into train, validation and test sets. The train and validation sets had 25 columns, whereas in the test set the columns of true clicks and market prices (pay prices) were removed. The key stats of the 3 splits are summarized below in Table 1.

Split	Impressions	Clicks	CTR	CPM	CPC
Train	2430981	1793	0.0738%	78.151	105.959
Validation	303925	202	0.0665%	78.234	117.709
Test	303375	N/A	N/A	N/A	N/A

Table 1: Stats of the train, validation and test sets.

W. Zhang et al. [6] did a quite insightful and comprehensive EDA on the whole iPinYou dataset showing correlations between different advertisers and other feature columns. We repeated their work but due to space limitation we would like to show insights that hasn't been discussed by them here.

- One of the 9 advertisers, advertiser 2997, only displayed on mobile operation systems, and was only associated with Ad Exchange null.
- The other 8 advertisers were all associated with Ad Exchange 1, 2, and 3. But only advertiser 2821 was associated with Ad Exchange 4.
- The impression shares of Ad Exchange 1 to 4 were respectively 29.5%, 31.4%, 33.7% and 3.3%.
- The slotid of all Ad Exchanges were non-comprehensible numbers except for Ad Exchange 3, whose slotids were comprehensible strings such as "QQlive_SP_HP_bottom_Width" (we guess this Ad Exchange is Tencent). The first word represented the sector, which had 29 types such as "Baby" and "Comic", and the last word was associated with the size of the ad-slot.
- There were 21 different slot-widths and 14 different slot-heights, but only 29 combinational slot-sizes. 13 of them were banners with aspect ratios larger than 5:1, others were squares or fat rectangles.
- There were 11 different slot-visibilities and 4 different slot-formats, but only 16 combinations excluding "N/A".
- There were in total 68 types of usertags. Each user had on average 4.51 tags, while the maximum was 48 tags and the minimum was 0.
- There were in total 131 different ad creatives, but only 96 of them had ever been clicked in the train set. The number of creatives of each advertiser ranged from 3 to 25.
- Each ad creative is associated with only one slot-size. Thus an advertiser promotes its campaign on a set of slot-sizes.
- The distribution of impression frequency against the number of feature instances with that frequency was long-tailed for many feature columns. For example, 10% of all domains contributed for over 95% of total impressions. And top 5 slot-sizes accounted for 78% of all impressions.

4. BASIC BIDDING STRATEGIES

This section discusses two basic bidding strategies, the Constant and Random bidding strategies, which does not involve any predictions. Then a competition involving multiple homogeneous agents running Random strategy was held to find how the related parameters would affect the number of total clicks of all the agents.

4.1 Constant Strategy

We experimented with a range of constant bid values (value $\in \{v|v \in [1, 99], v \in \mathbb{Z}^+\}$) on both train and validation sets, where the outcome stats such as the number of clicks, number of impressions, CTR and spend were recorded. In terms of the number of clicks, an empirically optimal constant bidding price on train data set (shown in figure 1) is about 23 or 24 (both achieved a same highest number of clicks) with 134 clicks whereas the optimal price on validation data set is about 77 or 79 (both achieved a same highest number of clicks) with 68 clicks (shown in figure 1). An interesting fact is that the price is fairly close to the average pay price of validation data set.

The number of clicks initially increases as the constant bidding price increasing until a point and then it drops as the constant bidding price continuing to increasing. The point is near somewhere that the cost reaches it limited value (in our cases, the 6250 budget). This is true for both train and validation data set.

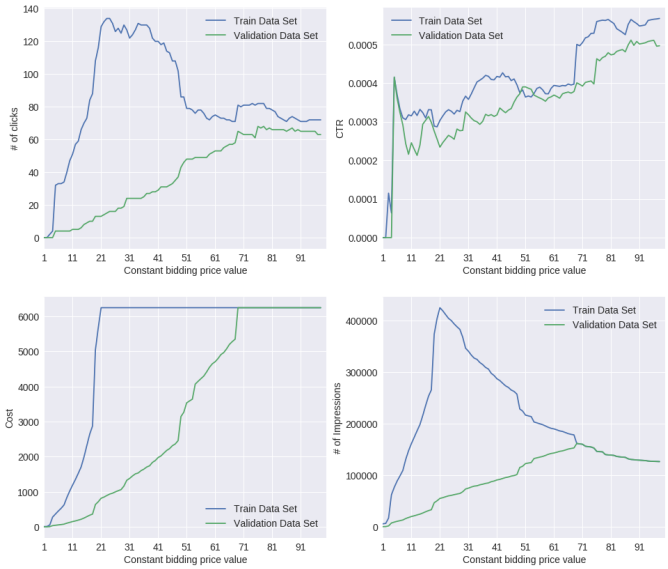

Figure 1: Statistics as a function of constant bidding price value

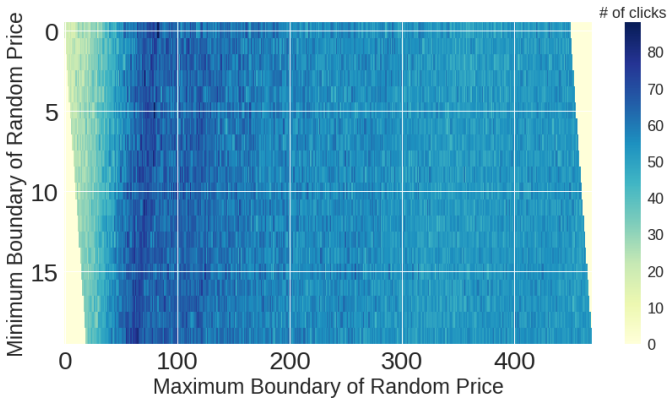

Figure 2: Number of clicks as upper and lower random bidding price boundary varying (on validation data set), Note that we set some non-covered (non-tested) range (left-bottom and right-upper part) to 0.

4.2 Random Strategy

To assist the further analysis of the question, we perform a random bidding strategy to see if any interesting points could be found. Figure 2 presents the results of a grid search for upper and lower boundary value for the random price number generation. We do a grid search with lower bound ranging from 1 to 20 (with a step of 1) and a window (the upper bound equals to lower bound plus the window) ranging from 50 to 500 (with a step of 1). The colour in the figure represents the number of clicks it gained on the validation data set and the darker colour represents more number of clicks. To ensure a consistent result, we also set the random seed to value 1. The deepest colour most located at the range that has maximum random boundary value from 70 to 100. The maximum number of clicks is 88 that achieved with a lower boundary of 1 and upper boundary 134. This is slightly better than the 68 clicks achieved by constant bidding strategy.

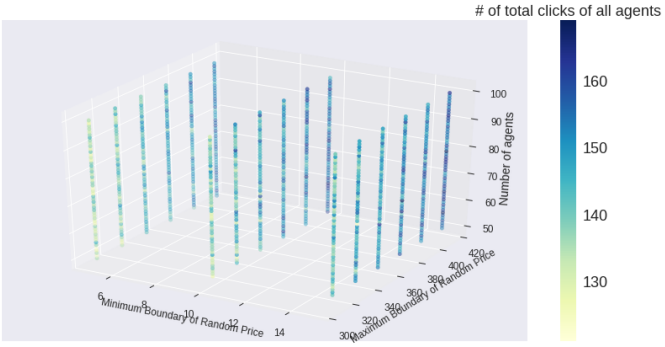

Figure 3: Number of total clicks as upper and lower random bidding price boundary and the number of agents varying (on validation data set) for Homogeneous Random Strategy.

4.3 Homogeneous Random Strategy Competition

For preparing multi-agent strategy, we built a simulator to simulate how agents' population would affected as the number of them increasing using random bidding agents. It could be noticed that there is no obvious changes and changing patterns from graph 3 and 4 in terms of total clicks or optimal random upper and lower bound. Nonetheless, this means a decreasing for number of clicks obtained of each agents when the their number increases. That is, when the competition growing up, everyone in the competition will get less clicks than before. The optimal value for different number of agents is shown in graph 4.

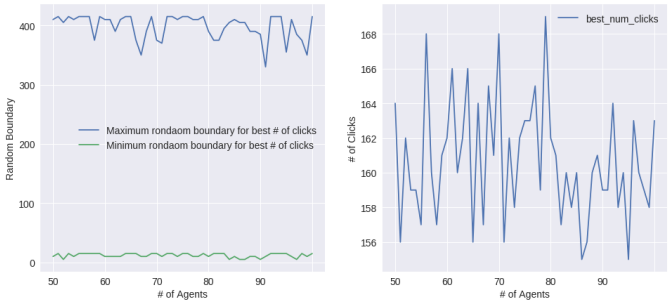

Figure 4: Optimal upper random boundary, lower random boundary and best clicks under the condition of different number of agents for Homogeneous Random Strategy

5. LINEAR STRATEGY

There are two subsequent components for a bidding strategy, i.e. predicting CTR and mapping the predicted CTR to a bidding price. A linear strategy uses a logistic regression

CTR predictor and a linear bidding function.

5.1 Linear CTR Prediction

We did the following feature engineering to the dataset. Firstly, we dropped the columns "bidid", "userid", "urlid", "IP", "url", and "keypage", since they either contained too many feature instances with statistically insufficient impression frequency, or irrelevant to CTR prediction. Before dropping the column "url" we merged it with the column "domain" for the rows where "domain" was null. We split "useragent" into "useros" and "userbrowser", and "usertag" into 68 binary columns each for a different usertag, and then dropped the original columns. Moreover, we merged "slotwidth" with "slotheight", and "slotformat" with "slotvisibility". We added a column "sector" extracted from "slotid" where applicable (Ad Exchange=3), and "N/A" otherwise.

Since the dataset was extremely imbalanced, we wanted our models to focus on learning about how to success instead of how to fail, so for each remaining column, we only kept the feature instances that were associated with a click in the train set, and ignored other instances. This left in total 1751 binary feature instances.

We developed LR, GBRT and MLP models for CTR prediction trained on these features. We also investigated in dropping extra columns with many feature instances, such as "domain", "slotid" and "creative", and found that this led to inferior performances of LR and MLP but better performance of GBRT.

Strictly speaking, only the combination of LR and the linear bidding function may be regarded as a pure linear strategy and discussed here. The other models and their combinations with the linear and different nonlinear bidding functions would be discussed in section 6.

5.1.1 Negative Down-sampling and Re-calibration

As did by X. He et al. [1], we adopted negative down-sampling on the training data to cope with the imbalanced dataset and limited computational resources. Similar to them, we also compared the performance of each CTR prediction model w.r.t. a range of negative down-sampling rates. They reported the negative down-sampling rate that gave the best GBRT performance was 0.025. However, we found that the optimal down-sampling rate for our GBRT, LR and MLP were respectively 0.01, 0.125 and 0.125. We believe this was due to different feature engineering and potentially different stats of the iPinYou and Facebook datasets.

Following the negative down-sampling method, we also did the method from [1] to enable the empirical CTR have a proper ration on the original (not down-sampled) data space. Specifically, we use $q = \frac{p}{p+(1-p)/w}$ where p is the predicted CTR on down-sampling space and w is the negative down-sampling rate. We found that down-sampling and re-calibration effectively help us on improving our final results.

5.1.2 Logistic Regression

We trained the basic LR with the 1751 binary features using Tensorflow Keras framework with Adam optimizer and binary cross-entropy loss. As W. Zhang et al.[6], we used AUC as metrics. Our LR obtained AUC of 0.8441, slightly higher than 0.8307 obtained by them, possibly due to our cleaner feature engineering and/or smaller dataset. Combined with the linear bidding function, it achieved respectively 148 and 166 clicks in the validation and test sets.

5.2 Linear Bidding Function

The simplest bidding strategy function that could involve CTR prediction is the linear bidding strategy. It formulates the bid price by combining the predicted CTR (pCTR), average CTR and a scaling parameter base_bid:

$$bid = base\_bid\frac{pCTR}{avgCTR}$$

5.2.1 Base Bid Tuning

For each major version of each CTR prediction model, we performed a grid search on base_bid over the whole validation set from the price 50 (less than the average CPM of validation set at 78) to price 1000 (12 $\times$ the average CPM) with step size of 1.

6. NON-LINEAR STRATEGY

6.1 Non-linear CTR Estimator

6.1.1 Multi-layer Perceptron Neural Network (MLP)

We built a MLP by adding 1 hidden layer of 2000 neurons with ReLU activation to LR. The model achieved AUC of 0.8816. Combined with the linear bidding function, it obtained respectively 162 and 177 clicks in the validation and test sets. The output of this combination also consistently stayed in top 10 of the second leaderboard (LB2) and peaked at the 2nd place for some hours.

6.1.2 Gradient Boosting Regression Tree (GBRT)

The Gradient Boosting Regression Tree is an iterative decision tree algorithm used for regression prediction. It is a widely used non-linear model and it is based on the boosting idea in ensemble learning. Specifically, each iteration constructs a new decision tree in the direction of reducing the residual gradient.

The advantage of GBRT over LR (Logistic Regression) is that GBRT has strong ability of learning the non-linear features while the linear model has limited learning ability. It requires a lot of work in feature engineering to analyze the effective features and feature combinations in advance, so as to indirectly enhance the nonlinear learning ability of itself.

With the appearance of XGBoost[1], GBDT (Gradient Boosting Decision Tree) has been implemented in an efficient system way. W. Zhang et al. [6] leveraged it for CTR estimation based on iPinYou dataset. And we reproduced the CTR estimator with XGBoost.

Before describing the parameter tuning process, we give a simple explanation of the XGBoost algorithm. Compared to GBRT, XGBoost explicitly adds the complexity of the trees as regularization to the objective function which can be described as

$$\sum_{i=1}^{n} l\left(y_i, \hat{y}_i\right) + \sum_{k} \Omega\left(f_k\right), f_k \in \mathcal{F}$$

where $\hat{y}_i = \sum_{k=1}^{K} f_k\left(x_i\right)$. The first term on the right side of the this function represents the training loss of the model, and the second term stands for the complexity of the trees.

[1]https://github.com/dmlc/xgboost

The loss function deployed in our CTR estimator is defined as

$$l\left(y_i, \hat{y}_i\right) = y_i \ln\left(1 + e^{-\hat{y}_i}\right) + (1 - y_i) \ln\left(1 + e^{\hat{y}_i}\right)$$

And the regularization term $\Omega\left(f_t\right)$ is defined as

$$\Omega\left(f_t\right) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

where T represents the number of leaf nodes, w_j represents the weight of the j^{th} leaf node. λ is the L2 regularization term on weights which is written as **reg_lambda** in XGBoost.

Then by additive training, which is also known as boosting. The prediction at round t is

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k\left(x_i\right) = \hat{y}_i^{(t-1)} + \epsilon f_t\left(x_i\right)$$

where ϵ is the **learning rate** or step size. This shrinkage factor ensures that we do not do full optimization in each step and reserve chance for future rounds which helps prevent over fitting.

The way to find the best tree structure is greedy learning, which is depended on the value of the gain after node segmentation. The gain is defined as

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{\left(G_L + G_R\right)^2}{H_L + H_R + \lambda}\right] - \gamma$$

The **gamma** parameter in XGBoost defines the minimum loss reduction required to make a further partition on a leaf node of the tree.

The parameter range is described below in the Table 2. We use grid search for obtaining the best combination of hyperparameters step by step.

Parameter	Range
n_estimators	range(1, 500, 1)
max_depth	range(3, 10, 2)
min_child_weight	range(3, 20, 3)
gamma	[0.1,0.2,0.3,0.4]
learning_rate	[0.01, 0.02, 0.06, 0.1, 0.15]
subsample	[0.6, 0.7, 0.8, 0.85, 0.95]
colsample_bytree	[0.5, 0.6, 0.7, 0.8, 0.9]
scale_pos_weight	[0.2, 0.4, 0.6, 0.8, 1]
reg_lambda	[1e-5, 1e-2, 0.1, 1, 100, 1000]
reg_alpha	[1e-5, 1e-2, 0.1, 1, 100, 1000]
max_delta_step	[0, 0.2, 0.6, 1, 2]

Table 2: Grid Search for parameter turning

We finally obtain the model with best performance of 185 clicks with parameters shown in Table 3

n_estimators	max_depth	min_child_weight	gamma	learning_rate	subsample	colsample_bytree	scale_pos_weight	reg_lambda	reg_alpha	max_delta_step
320	7	9	0.1	0.06	0.95	0.7	1	1	1	0

Table 3: Parameters of Best Performance Model

6.2 Non-linear Bidding Function

6.2.1 Optimal Real-Time Bidding (ORTB)

In the paper [5], the author derived 2 different bidding functions.

$$b_{ORTB1}(\theta) = \sqrt{\frac{c}{\lambda}\theta + c^2} - c$$

where θ is predicted CTR in our scenario, λ and θ are 2 adjustable parameters. Another bidding function is defined as

$$b_{ORTB2}(\theta) = c \cdot \left[\left(\frac{\theta + \sqrt{c^2\lambda^2 + \theta^2}}{c\lambda}\right)^{\frac{1}{3}} - \left(\frac{c\lambda}{\theta + \sqrt{c^2\lambda^2 + \theta^2}}\right)^{\frac{1}{3}}\right]$$

Unfortunately, we found that these two bidding functions not work well with our CTR estimators. There are 2 parameters, c and λ. We found that the value of λ has relatively larger affect on the final produced price. Using the result on validation set of Logistic Regression predictor, we fixed the value c to 50 (as it is one of suggested value in this paper) and did a grid search from $1e-9$ to $1e-1$ with a interval $1e-4$. For both of these 2 strategies, we found that they both achieved a best number of clicks on $1e-4$ with maximum number of clicks of 63 and 61 respectively. We finally did NOT use these 2 bidding functions in our later experiments because of the relatively poor result compared with other bidding functions. This may because we failed to find a best set of parameters for the 2 bidding function or the situation of our data set did not match the problem that the bidding functions attempts to resolve (for example a random guess could be that the 2 bidding price probably perform better if given a very limited budget).

6.2.2 Quadratic Bidding Function

We proposed an intuitive non-linear variant of the linear bidding function:

$$bid = base\_bid\left(\frac{pCTR}{avgCTR}\right)^2$$

Empirically combine this bidding function with GBRT CTR estimator, we achieved our best performance for criterion 1 on the test data set, with 185 clicks .

6.2.3 Models Involving Pay Price Prediction

We built two different models involving pay price prediction. However, due to the noise in the dataset neither of them performed well.

One model adopted the key idea of the Bidding Machine proposed by K. Ren et al. [3] by replacing the binary reward of an impression with a utility function: $Reward = Click\ Value \times Click - Payprice$, where $Click \in \{0, 1\}$, and $Click\ Value$ was the relative value of a click which we experimented with $\{300, 500, 700, 1000, 2000\}$. The utility was normalized and used to train the MLP we built before. However, combining the predicted utility with the linear bidding strategy achieved less than 150 clicks on the validation set, which decreased with $Click\ Value$. This was because, due to the wide range of feature variety in the very few positive samples and noise in the negative samples, all CTR prediction models had a considerably high false negative rate. With the inclusion of pay-price estimation, a false negative sample with high predicted pay-price could get lower predicted utility than a true negative sample with lower predicted pay-price, and thus the former became less likely to

be "saved" by the linear bidding strategy, or similar scaling strategies, to win the impression.

The other model tried to maximize:

$$\max_{B} \& \sum_{i=1}^{N} \sum_{M=0}^{B_i - 1} \hat{P}(M_i|C_i = 1)\hat{P}(C_i = 1)$$

$$s.t. \ \& \sum_{i=1}^{N} M_i \mathbb{I}(B_i > M_i) \leqslant Budget$$

where N is the total number of impressions, C, B, and M are respectively click, bid-price and pay-price. We built a separate MLP to estimate $P(M_i|C_i = 1)$, which divided pay-price into 30 bins of width 10, and achieved accuracy of approx. 0.4 and top-5-bin accuracy of approx. 0.7. We then approximated the optimization problem by linear programming for each batch of 1000 impressions and solved it using Scipy. Similar as the above, due to limited positive training samples and large noise in data, this model only achieved 135 clicks in the validation set.

6.3 Combination of CTR Estimator and Bidding Function

We tested various combinations of CTR estimators and bidding functions. For winning criteria 1, we have following submission results (given a CTR estimator model, we only record a best result that appeared in a combination of any bidding function the CTR Estimator.)

CTR Model	Bidding Function	Base_Bid	Clicks (Test set, LB1)	Clicks (Validation set)
LR	Linear	206	166	148
MLP	Linear	136	177	162
MLP	Quadratic	175	174	160
XGBoost (with sub-optimal parameters)	Linear	96	173	161
XGBoost (with sub-optimal parameters)	Quadratic	179	179	174
XGBoost (with optimal parameters)	**Quadratic**	161	**185**	171

Table 4: Some combinations of CTR estimator and bidding functions and their best clicks result for wining winning criteria 1 on test data set (with a total number of 223 clicks) and validation set (with a total number of 202 clicks), respectively

CTR Model	Bidding Function	# features	hidden layers neurons	# classifiers	max tree depth	learning rate (α)	down-sampling rate
LR	Linear	17000	N/A	N/A	N/A	0.001	0.125
MLP	Linear	1751	2000	N/A	N/A	0.001	0.125
MLP	Quadratic	1751	2000	N/A	N/A	0.001	0.125
XGBoost (sub-optimal)	Linear	886	N/A	347	5	0.1	0.01
XGBoost (sub-optimal)	Quadratic	886	N/A	347	5	0.1	0.01
XGBoost (optimal)	Quadratic	762	N/A	320	7	0.06	0.01

Table 5: Parameter configurations for each combination of CTR estimator and bidding functions

7. MULTI-AGENT BIDDING STRATEGY

7.1 Multi-agent Environment Simulation and Market Landscape Analysis

We simulated the multi-agent competitive bidding environment using evolutionary algorithm. For the particular experiment showed here, the environment had 30 agents and 100 rounds. All agents used linear bidding strategy, started with the same base_bid. In each round the whole validation set was looped through, and each agent bid based on a different modified version of our best pCTR until using up its budget. After each round the worst 40% agents were eliminated, while each of the best 20% agents gave birth 2 children whose base_bid was the parent's base_bid timed by a factor drawn from N(0,1). Figure 5 below showed the market landscape in the last round. For brevity we denote the dimension of increasing rows in the dataset as "time", along which we split the whole dataset into 30 bins. We then split each bin further into 10 bins along the dimension of pCTR, so that each bin contained about 1000 samples. We took average value in each bin for the interested quantities including number of clicks, historical pay-price, current pay-price, highest bid-price, and cost-efficiency of the bin (1/avgCPC).

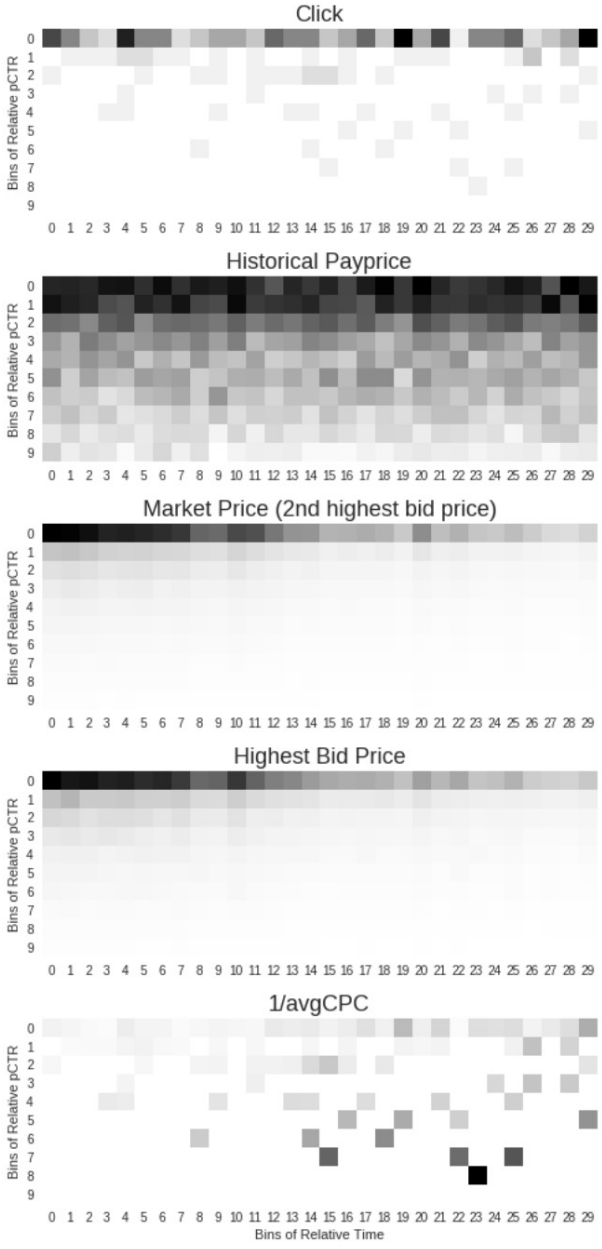

Figure 5: Market Landscape in the last round of the multi-agent simulation.

We can see that our pCTR model was generally good although there are some false negatives deeply submerged into true negatives. Historical pay-price in the dataset was posi-

tively correlated with the pCTR. The market segments that were early in time and high in pCTR were the most competitive, which contained the highest bid and pay prices. As time passed, increasingly more agents used up the budget, and the competition became less radical. It's interesting to note that, the most cost-effective segments actually located late in time and low in pCTR.

Theoretically, according to game theory, given perfect information, a newcomer would always pick the most cost-effective slots, making them more competitive and less favorable to players come after. Eventually with enough players the market landscape would become generally flat with consistent local fluctuations. Such behaviour aligned with the emergence of different market segments being occupied by different agents, as shown below in Figure 6. By differentiate their base_bid, the top-5 agents in each round (note the identities of these agents may have changed over rounds) separated their market occupation in both the pCTR and time dimensions, thus some clustering can be seen in Figure 6[Right]. Moreover, Figure 6[Left] showed interesting periodical back-and-forth fluctuation on the boundaries of the agents' territories. This indicated that the best agents tended to avoid direct collision with each other, whereas the worst agents tended to be eliminated due to poor market location and being smashed in collisions.

However, in the real world competition where groups used different strategies and pCTR models, there would be much more noise in the bid-price landscape, making the "slots with treasures" much harder to locate, and the market equilibrium less stable. The fact that the groups could only obtain the overall performance on the validation set, not feedback on individual impressions and clicks worsened this.

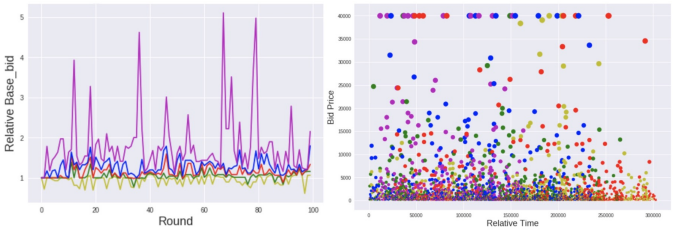

Figure 6: Left: Sorted relative base_bid value of the top 5 agents in each round. Right: Location of the top 5 agents' bids in the landscape in the last round.

7.2 Leaderboard 2 Analysis

Unfortunately, the discussion above only holds when all 30 agents entered the market at the same time. In the real world, group 1 created monopoly on the leaderboard 2 because they entered the market much earlier than other players, purchased a significant amount of high pCTR impressions at a very low pay-price (their very low CPM indicated that they mostly purchased from the original dataset pay-price) and meanwhile proposed very high bid-price. As a result, all other groups, including us, had to fight for their territories on the plateau created by group 1, which significantly limited the size of their territories. As shown in Figure 7, groups ranked 2-13 all used up their budgets with hundreds of CPC, in contrast to the low CPC and CPM of group 1. If we ignore group 1 and only consider the rest of groups, their CPM indicated that their behaviour and market occupation aligned with our discussion before.

We tried several methods to dodge direct collisions with other groups. Most of them, such as only focusing on segments late in time or medium-low in pCTR, didn't perform well due to the large noise and lack of information in the actual market landscape. However, we found that using MLP-based pCTR significantly outperformed using GBRT-based pCTR. We believe this was because the two models produced differently-shaped distributions in pCTR, and GBRT was popular among other groups but not MLP, so that we effectively dodged collisions with many groups.

Leaderboard - Criterion #2:

Ranking	Group	Clicks	CTR	CPC	Impressions	Cost
1	1	59	0.006673453229272707	7.473966101694938	8841	440.9640000000013
2	9	16	0.0019743336623889436	390.6248520064049	8104	6249.997632102479
3	18	14	0.001435897435897436	446.4249857507556	9750	6249.949800510578
4	6	14	0.0009744553490638268	446.4279892539893	14367	6249.99184955585
5	5	12	0.0014677103718199608	520.829331090868	8176	6249.951973090416
6	24	11	0.0006869847614289283	568.1813370494157	16012	6249.994707543572
7	13	11	0.0003484320557491289	568.181442832351	31570	6249.99587115586
8	4	9	0.005924950625411455	694.4440607877641	1519	6249.996547089877
9	17	9	0.002555366269165247	694.4443074891414	3522	6249.998767402272
10	15	8	0.001005151400929765	781.2466974861436	7959	6249.973579889149
11	3	8	0.00033886818027787193	781.245955945701	23608	6249.967647565608
12	11	8	0.0001380381330342507	781.2485204850566	57955	6249.988163880453
13	12	7	0.001488095238095238	892.8567306599291	4704	6249.997114619504

Figure 7: Leaderboard 2 screenshoted at 21.00 March 11.

8. CONCLUSIONS

We found that GBRT and MLP were effective CTR prediction models, which if tuned well could outperform LR. We found that constant bidding functions had limited performance and hence we did some non-linear strategies such as the Bidding Machine. On the other hand, we found that linear and quadratic bidding functions were both very effective for our problem. We achieved our best performance on LB1 and LB2 respectively by combining GBRT with quadratic strategy and MLP with linear strategy.

By analyzing the market landscape produced by our multi-agent bidding simulation environment, we showed the divergence of agents' focuses into different market segments in order to maximize cost-efficiency and avoid direct collision. The market equilibrium would be achieved when there are enough players in the market and no player could spare a large chunk of other player's territory without sacrificing its own territory. Besides, we explained how group 1 created monopoly on LB2. The only way to beat group 1 was that several groups collaboratively bid just below the bid price of group 1 to drain its budget. However, collaboration was not allowed by competition rule. Instead, in a pure competitive environment, players (including us) usually become nearsighted and greedy, and bid for immediate increase of their own rewards. In fact, bidding systems that focuses on long-term rewards in a competitive environment and agent collaboration are something we think worth to be looked at in the future.

9. CODE REPOSITORY

Our relative code could be found at https://github.com/ChenhaoLu/Multi-agent-Artificial-Intellgience.

10. REFERENCES

[1] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Q. Candela. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, ADKDD'14, New York, NY, USA, 2014. ACM.

[2] H. Liao, L. Peng, Z. Liu, and X. Shen. ipinyou global rtb bidding algorithm competition dataset. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, KDD '14, New York, NY, USA, 2014. ACM.

[3] K. Ren, W. Zhang, K. Chang, Y. Rong, Y. Yu, and J. Wang. Bidding machine: Learning to bid for directly optimizing profits in display advertising. *IEEE Transactions on Knowledge and Data Engineering*, 30(4):1–18, APRIL 2018.

[4] J. Xu, X. Shao, J. Ma, K.-c. Lee, H. Qi, and Q. Lu. Lift-based bidding in ad selection. *Association for the Advancement of Artificial Intelligence (www.aaai.org)*, 2016.

[5] W. Zhang, S. Yuan, and J. Wang. Optimal real-time bidding for display advertising. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1077–1086, New York, NY, USA, 2014. ACM.

[6] W. Zhang, S. Yuan, J. Wang, and X. Shen. Real-time bidding benchmarking with ipinyou dataset. *UCL Technical Report*, 2014.