Multi-Agent AI

[Group Coursework 1]

Aksel Cakmak
15047472
zcababc@ucl.ac.uk

Chong Yang
18089022
ucabcya@ucl.ac.uk

Chen Song
17040773
ucabcs3@ucl.ac.uk

1. INTRODUCTION

Real-time bidding (RTB) offers advertisers a way to adaptively bid for ads on a per-impression basis, in real time. It uses data from the user's context (cookies, meta-data like the browser used, the website visited, etc..) to formulate a bid price for a certain slot.

If the auction is won, the bidder's ad is then displayed in the slot. Afterwards, the user might or might not click on the displayed advertisement.

RTB differs from Sponsored Search Auctions, wherein an auction mechanism tries to match advertisers that bid on certain search keywords, not on specific impressions.

Bidders have a certain budget to allocate to a certain number of bids, and (in our specific setting) want to maximise the total number of clicks they get from all the ads they display.

For a specific slot, the bidder is usually interested in some specific KPI (key performance indicator) that they use to formulate their bid (in our case, we're interested in the predicted click-through-rate of a specific slot, or pCTR).

Then, a bidder has the following dimensions to consider for its bidding campaign:
- How to predict the KPI (here, pCTR), the choice of a predictive model, its accuracy, etc..
- How the bid is formulated wrt a given pCTR.

The bidder wants to optimise the number of clicks it gets by choosing a good KPI predicting model and a proper bidding function (in our specific setting). In addition to this, one other constraint there is is that the whole pipeline of KPI prediction + bid function has to be calculated with minimal latency (bids often have to be submitted within a short timeframe, like 100ms) [8, 2].

2. DELIVERABLES

Our deliverables are as follows:

- Our bidding prices on the testing set, testing_bidding_price_criterion_1.csv
- testing_bidding_price_criterion_2.csv that account for the different winning criterions.
- The Github repo that contains the source code of our project: https://github.com/JustinYaaang/COMPG0124.

3. APPROACH AND RESULTS

3.1 Problem 1: Data Exploration and Literature Review

3.1.1 Literature Review

The performance of a CTR prediction model has a direct impact on the final number of clicks generated by a campaign, which is why a lot of work has been made on how to predict the CTR of a given slot.

Regularised Logistic Regression models have been commonly used for the task of predicting CTR [1], but are lacking in that they require some feature engineering work and are not as accurate as newer Deep Learning-based methods. [3]

As a consequence, more Deep Learning-based methods have been proposed recently, where the input features are fed into neural networks which learn the implicit nonlinear relations between the different features, and consequently report enhanced accuracies. [4, 5, 7]

Once the pipeline has a model to predict the CTR, there exist different approaches to calculating a bid wrt this pCTR.

Some very simple approaches include bidding a constant value or choosing a random bid based on some range. Quite naturally, these approaches don't fare as well as other more sophisticated methods, that bid a variable amount using the pCTR and the contextual information of a specific slot. [9] One more advanced approach is linear bidding: the bidding value is an affine function of the predicted CTR (pCTR). This method fares far better than the aforementioned but is outclassed by non-linear methods like [9].

One challenge of this particular setting is that the bidding is budget constrained: the advertiser (in our setting) wants to maximize the number of clicks it gets from its impressions, for some number of slots it bids for. Knowing how fast one should burn through the given budget is a difficult problem[10]: If the strategy bids relatively high values, the budget might be spent early, and some potentially valuable slots might be missed. If the strategy is relatively more conservative, then the budget might not be fully spent, and some valuable slots might be underbid on. The problem is even more complex when considering the fact that there are a number of heterogeneous and unpredictable other agents with their own bidding strategies competing against a given bidder [10].

3.1.2 Data Exploration

We analysed some aspects of the data with respect to the weekdays, the CPC, CTR, and payprice.

As one can see on the figures below, CPC, CTR, and

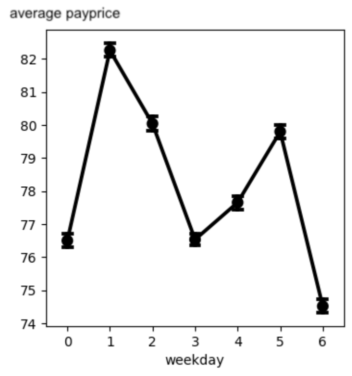

Figure 1: Average Payprice wrt Weekday

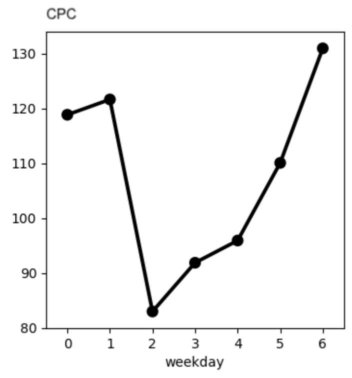

Figure 2: Average CPC wrt Weekday

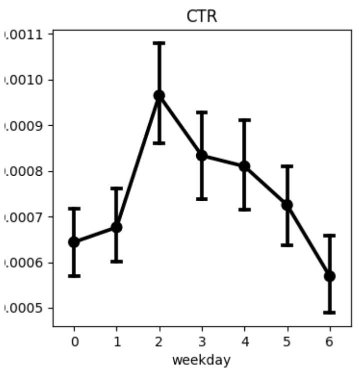

Figure 3: Average CTR wrt Weekday

payprice are extremely variable throughout the week. The day with the lowest average CPC of 81 is day 2, and the highest CPC of 130 is attained on day 6. The CTR per weekday falls within the range of 0.0006 (at day 6) and 0.00095 at day 2. Finally, the payprice per weekday also belongs to a notable range, from 74 (on day 6) to 82 (on day 1).

From further analysis, we observed that the CTR per slot and payprice per slot are positively correlated; This makes sense for our setting: assuming reasonable KPI predictive models, a slot with a high CTR would also tend to have a high pCTR, meaning it would also have a higher private value to bidders, and from this a higher payprice from the higher bids.

3.2 Problem 2: Basic Bidding Strategy

In this section, we analyse two basic bidding strategies, which are Constant Bidding Strategy and Random Bidding Strategy, and evaluate their performance based on the number of clicks within a limited budget of 6,250 CNY fen.

Evaluation function: For the single-agent basic bidding strategies, the main metric to rank the strategies are based on the clicks from winning impressions.

3.2.1 Constant Bidding Strategy

In order to find an optimal constant value, we loop the constant bid prices within a feasible range, from 0 to 300, which are the minimum bid price and the maximum bid price of all the slots in the dataset, to find out the constant bid price with the highest clicks from winning impressions. Specifically, for each constant price, we retrieve the columns of 'pay price' and 'click' for all the bids in the training set. Then we compare our constant bid price with the 'pay price' for each bid and add up the click into our total clicks if our constant bid price is great than or equal to the 'pay price' while the total spend is calculated at the mean time. Afterwards, we remove the clicks from bottom to top where the total spend has been over our limited budget. In order to use the training set to find out a good price for validation set, initially, we need to normalize the budget based on equation 1.

$$budget_{train} = \frac{sizeOfTrain}{sizeOfValidation} * budget_{validation} \quad (1)$$

Analysis: Figure 1 shows how the value of click changes based on the increment of the constant bidding price. The clicks increase dramatically when the constant bidding price increases from 1 to 78 and the climax of clicks is 643 when the constant bidding price is 77. Then the clicks drop smoothly when the bidding price increases from 78 to 300.

Figure 2 shows the changes in click value depending on bid price in the validation set with the standard budget. We could see the clicks are relatively high when the bid price is between 70 to 100. Surprisingly, the value of the click is maximised as 68 when the bidding price is 77 or 79. Therefore, the findings in our training data match the validation set.

3.2.2 Random Bidding Strategy

In order to find the optimal bidding range for random bidding, we step through a range of lower bound and a range of upper bound to find out the bid price with the highest clicks from winning impressions. Similarly, we use the same

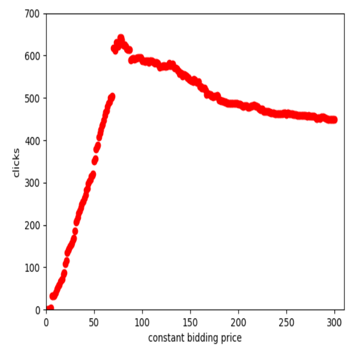

Figure 4: Training set - clicks wrt constant bid price

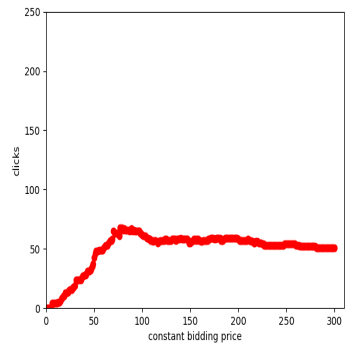

Figure 5: Validation set - clicks wrt constant bid price

Table 1: Optimal Bounds and Clicks

agent number	50	60	70	80	90	100
lower bound	90	90	110	70	130	50
upper bound	300	300	300	300	300	300
clicks	202	202	202	202	202	202

method as the one in Constant Bidding Strategy to calculate the clicks.

Analysis: The highest clicks are 628 generated from range 10 to 130 in training set with the limited normalised budget. In the validation set, we find the highest clicks are 76 generated from the range 20 to 150. It is acceptable that the best range in these two sets is not far from each other.

3.2.3 *Competition among homogeneous random bidding agents*

Competition among homogeneous random bidding agents: In the sub-problem, we first step into different combinations of lower bound and upper bound. And for each pair of the lower bound and upper bound, we generate n agents who are using random bidding strategy with the price with the selected bounds. Then we add the click to the winner following criterion 2 for each bid. Our criterion for bound comparison: the average click among the agents is the criterion of winning bounds. And the optimal bounds should generate the highest average clicks (total clicks). Table 2 shows optimal bounds and total clicks of different numbers of agents.

Consequently, the optimal upper bound is 300 for all of the group of multiple agent bidding, which is much higher than the single agent bidding. The reason for this high upper bound is that the strategy of multiple agent bidding is to maximize the total number of clicks, and the higher upper bound is, the larger opportunity that the total clicks could be maximised. In other words, the strategy for multi-agent random bidding is to maximise the benefits of the group.

3.3 Problem 3: Linear Bidding Strategy

In order to apply CTR estimation to for a linear bidding strategy, we initially retrieve these features as independent variables X: day, hour, region, ad exchange, slot width, slot height, advertiser, slot visibility, slot format, OS, browser, and slot price from the data set. Specifically, we categorise the slot price to five categories based on the price values, and we extract the OS and browser from the column useragent. The rest of the features is simply fetched from the data. The click from the data is our predictor Y. Afterwards, we use a Logistic Regression model, with a cross-entropy loss function, L2 regularisation and train it with the independent variables and predictor from the training set. Then we use the trained model to predict the click of test data and validation data separately. The pCTR of the validation data could be calculated with the equation 2.

$$pCTR = \frac{numOfClicks}{numOfWinningImpressions} \tag{2}$$

The bid price for each bid is calculated as equation 3. As shown in Figure 3, the total clicks increase sharply when the base bid increases from 1 to 20 and drop smoothly after then. The value clicks is maximised as 39 when the base bid is 20.

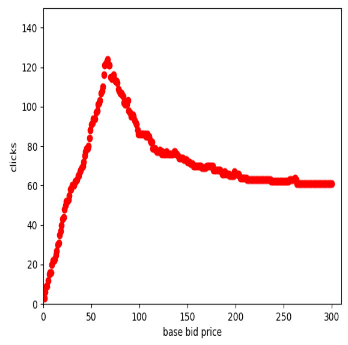

Figure 6: Validation set - base price and clicks

Table 2: Behaviour Comparison

	Constant Bidding	Random Bidding	Linear Bidding
price	79	range(20, 150)	67(base)
clicks	68	76	124

$$bidPrice = \frac{baseBidPrice * pCTR}{avgCTR} \quad (3)$$

Comparison:

Obviously, the behaviour of the linear bidding strategy is much better than random bidding and constant bidding strategies. The optimal value of clicks in linear bidding is 124 while the ones of constant bidding and random bidding are merely 68 and 76 separately.

3.4 Problem 4: Non-Linear Bidding Strategy

We calculate the non-linear bidding price based on ORTB (equation 4), and we step through some combination of c and lambda to find the optimal pair generating the optimal bid prices. We find the value of clicks is 39 when c equals to 39 and lambda equals to 1.31072e-05. Therefore, our non-linear bidding strategy just generates the same result as linear bidding strategy.

$$bidPrice = \sqrt{\frac{c}{\lambda} * pCTR + c^2} - c \quad (4)$$

3.5 Problem 5: Multi-agent Bidding Strategy

In order to find a good bid strategy for Winning criterion 2, we train a multi-agent model where 80 agents attend to the competition by using different strategies:

1) There are 20 agents using constant bidding strategies with constant bidding prices from 110 to 300 intervals 10.

2) There are 20 agents using the same random bidding strategy found in previous sub-problem 2.2.3 (multi-agent random bidding strategy).

3) There are 20 agents using linear bidding strategies with the base bid prices from 110 to 300 intervals 10.

4) In order to find a good strategy for non-linear bidding, we run a twenty-agent competition with different combination of parameters (c and lambda are equal to 50 and 2.048e-07 separately) to find the optimal non-linear strategy. Then there are 20 agents using this same non-linear bidding strategy.

Table 3: Best performance

	Constant	Random	Linear	Non-linear
highest click	3	2	20	12

Then we concatenate the bid prices of these 80 agents together, assign each agent limited budget, and start the competition. The agent wins in a bid if his or her bid in this row is greater than or equal to all the other agents as well as the pay price in the data. And this agent could gain a click if there is a click in this winning bid. Moreover, this agent needs to pay the second highest price and his budget will decrease by that price. Table 3 show the agent with the highest number of clicks in each strategy.

Analysis:

1) As expected, both constant bidders and random bidders have a poor performance in this competition since they do not adjust their price based on data (like pCTR) strategically.

2) Surprisingly, the best performer amongst the linear bidders gains a higher number of clicks than the best performer amongst the non-linear bidders. In our opinions, the reason for this behaviour is that we did not implement enough training on the parameter discovery (in other words, we didn't test enough combinations of c and lambda). Ideally, we are supposed to attempt more combination of parameters and implement the competition with more agents that just 20 (we chose 20 agents because of time constraints).

4. CONCLUSION

We recap on what we did for each problem.

We started off by building a simple agent that bid a constant value for all slots, then found the optimal value of the bid to maximize the number of clicks earned by our agent. Then we built a bidder that chose a random bid from within a certain range for each slot, and found an optimal range. We then modelled 50-100 of these agents and studied the performance of the random bidding model in this setting. For the next part, we used a Logistic Regression model to predict the CTR of ad slots, and bid an amount linearly proportional to the predicted CTR. Then, keeping our predictions from the previous part, we built a non-linear bidding function. Finally, we simulated a number of heterogeneous agents to compete for the slots and analysed our model's performance in this setting.

We would briefly like to mention the relative performance of our models on the last problem. When we compare our four strategies for a single agent bidding, the performance rankings from high to low are Non-linear bidding strategy, Linear bidding strategy, random bidding strategy, and constant bidding strategy. However, surprisingly, in our multi-agent competition, the linear bidding strategy performs better than the non-linear bidding strategy. We conclude that this phenomenon is due to the lack of training in the optimal parameter exploration for non-linear bidding. It takes on average four hours for a competition of twenty agents non-linear bidding with five hundred combination of parameter c and lambda. If we had time to improve our non-linear model, we would have made a grid search considering a higher range of parameters.

5. REFERENCES

[1] H. Brendan Mcmahan, H. Brendan Holt, et al. 2014. Ad Click Prediction: a View from the Trenches. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 1222-1230.

[2] Weinan Zhang et al. 2014. Optimal real-time bidding for display advertising. Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining.

[3] Guorui Zhou et al. Deep Interest Network for Click-Through Rate Prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems. ACM, 191-198.

[5] Cheng H. et al. 2016. Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. ACM.

[6] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep Crossing: Web-scale modeling without manually crafted combinatorial features.

[7] Shuangfei Zhai, Keng-hao Chang, Ruofei Zhang, and Zhongfei Mark Zhang. 2016. Deepintent: Learning attentions for online advertising with recurrent neural networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 1295-1304.

[8] Shuai Yuan, Jun Wang, and Xiaoxue Zhao. Real-time bidding for online advertising: measurement and analysis. In Proceedings of the Seventh International Workshop on Data Mining for Online Advertising, page 3. ACM, 2013.

[9] Weinan Zhang, Shuai Yuan, and Jun Wang. Optimal real-time bidding for display advertising. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1077-1086. ACM, 2014.

[10] Di Wu et al. Budget Constrained Bidding by Model-free Reinforcement Learning in Display Advertising. ACM, 2018.

[11] Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. 2008. Budget constrained bidding in keyword auctions and online knapsack problems. In International Workshop on Internet and Network Economics. Springer, 566-576.