COMPGW02/M041: Web Economics Project

Group submission report

Kamakshi Bansal
Data Science, UCL
ucakkba@ucl.ac.uk

James Shields
Data Science, UCL
james.shields.16@ucl.ac.uk

Said Kassim
Data Science, UCL
ucaksas@ucl.ac.uk

ABSTRACT

In this report, we present a study of real-time bidding (RTB), bid optimization and bidding strategies for online display advertising. We have implemented, optimized and evaluated a breadth of bidding strategies from a constant bidding baseline model to non-linear sophisticated bidding models, with each model evaluated and analyzed. The project can be divided into two stages: 1) estimating predicted click through rate (pCTR) for impressions based on features and; 2) designing and optimizing a bidding strategy which maximizes performance against key performance indicators (KPI's). We have implemented a range of models for both tasks and present a detailed analysis of all approaches. This report finds that pCTR can be calculated with an (AUC) accuracy of roughly 83% and that non-linear bidding strategies can substantially outperform baseline bidding strategies by a 310% increase in click-through rate (CTR) and and 50% reduction in effective cost per click (eCPC) for online display advertising.

1. INTRODUCTION

Online display advertising exchanges and associated supply and demand platforms are the ecosystem where online display impressions are bought and sold. Individual desktop and mobile display advertisements are exchanged at an impression level in real-time global auctions. Advertisement exchanges (Ad-exchanges) host impression auctions and can clear billions of transactions daily. Each auction occurs online and without a noticeable delay for the advertisement receiving user.

Advertisement agencies and other entities wishing to bid in these auctions and buy online display advertising impressions require a real-time bidding (RTB) strategy to be able to compete in the live auctions hosted by the exchanges. Buyers have bidding strategies which are hosted by demand side platforms (DSPs) and interface with the exchanges to submit bids. Each bidding strategy is designed to the buyers budget, appetite and expectations (all of which are constraints to the bidding strategy's KPI). All these create an interesting and commercially rewarding machine learning challenge.

The key metric used to evaluate bidding performance, click-through-rate (CTR), is whether the advert has been clicked by the end user. In online display advertising this is typically very low (0.05-0.5%) as only a small fraction of impressions are clicked on. This can mean advertising campaigns see very little activity and at an expense to the buyer.

There is substantial variability between impressions, be it size, position, user device, time of day, agency. These are listed as an impression's features and affect the likelihood of an advertisement to be clicked. As each impression is sold at auction the market price is also variable and so is the cost of each impression.

These challenges create a dynamic environment which provides potentially high rewards for optimizing strategies. This is the aim of this project. In this report, we first review related work on the problem (section 2), we then implement and optimize baseline bidding strategies followed by enhanced strategies (section 3), we analyze the results (section 4) and finally we give our conclusion and possible future work (section 5).

2. RELATED WORK AND BACKGROUND

The relevant literature in the field of real-time bidding, bidding strategy has been reviewed and is summarised below.

2.1 Display advertising

Display advertising infrastructure consist of an ecosystem of exchanges and demand and supply side platforms which facilitate the buying and selling of advertisement display at an impression level. Exchanges host bidding auctions for display impressions. Supply and demand side platforms connect with the exchanges on either side to list impressions and make bids respectively. Exchanges operate globally and there are many, transacting billions of impressions daily [1]. Supply side platforms (SSP's) enables web publishers and content providers to list their available ad inventory. Demand side platforms (DSP's) support advertisers by placing bids and managing their real-time bidding strategies.

2.2 Display advertising campaign KPI

Key performance indicators (KPI's) are defined by the buyers or other stakeholders who are funding the advertisement campaign. These depend on the buyers requirements e.g. budget, time, audience, etc. A campaign is defined as a deployment of a bidding strategy with a specific KPI. Below is a review of the prominent KPI's for online display advertising bidding strategies.

Click-through-rate (CTR) is the percentage of users that click on an impression.This is equal to the number of clicked impression won divided by the total number of impressions won. The apriori likelihood of an impression being clicked is low due to the unbalanced and sparse nature of the data set.

Number of clicked bids is the count of clicked impressions. This is an important indicator to consider as very conservative bidding strategies (ones that win few impressions) can have a high pCTR but if that is paired with a low click count then its is an unrealistic strategy.

Another important consideration is the budget of the campaign and the total money spent on the won impressions. This will typically govern the end of a campaign i.e. when the allocated budget has been used up. Bid strategies are often evaluated against a range of budgets [5] so as to analyze a bidding strategy's performances.

2.3 Predicted click-through rate

Predicted click-through rate (pCTR) is the expected probability of an impression being clicked by a user. pCTR is a key component for optimized real-time bidding strategies [10]. The performance of pCTR models can be a main source of competitive advantage in impression auctions.

pCTR is learned from training samples which contain features and correct labels of whether the impression was clicked or not. The impression features are the input and whether the impression has been clicked is the label. With this, there are a plethora of supervised learning approaches that can be used to determine pCTR such as naive Bayes, logistic regression and deep learning [10].

pCTR is typically evaluated by the 'area under the curve' (AUC) metric which is a good measure of classification prediction accuracy [10]. Studies have shown, that on display advertising data, AUC accuracies of up to 70-80% [6] can be obtained.

2.4 Real-time bidding strategies

Real-time bidding (RTB) is the format of the impression auctions hosted by ad exchanges. Auctions occur at impression level and in real-time. Advertisers through DSPs, use real-time bidding strategies which automatically submit bids for impressions based on the impressions features and campaign constraints i.e. budget, eCTR, CPC.

Basic bidding strategies use no prior information or impression features to formulate bids. Typically bids are either a random or constant value for each impression.

Linear bidding is a substantial improvement to the basic bidding models. It requires an underlying predictive model which first calculates pCTR based on an impression's features and then applies a linear function to the pCTR to generate a bid price. Linear bidding strategies are well studied and are regularly used[3]. They have been found to substantially outperform basic bidding strategies by up to 50 times[5].

Non-linear bidding is less well studied but has shown improved results compared to linear bidding in [5]. Non-linear bids consider the winning probability of the impression and its bid price to form a non-linear function to produce its bid. With this, there are also also additional non-linear parameters which are learned based on the data and bidding strategy which affect the value of the bid.

Deep learning models have far superior expressive power to traditional machine learning models and as such have surpassed their predictive performance in many other areas[7]. Deep learning can be applied to pCTR estimation and has been shown to successfully learn data patterns and improve pCTR performance compared to traditional models[6]. We were able to find little or no research in the use of deep learning for bidding strategies.

Recent papers have shown that reinforcement learning can be applied to online display advertising [4]. Bidding strategies form fairly natural reinforced learning problems because of the contained 'environment' (impressions, features and suppliers), the limited actions for the 'agent' (bid or not bid, bid high or low), and simplicity of reward (impression clicked).

2.5 Bid optimisation

Bid optimisation has been studied extensively [3]. In the case of bidding strategies, this involves tuning its parameters to make it perform well when measured against defined KPIs.

For simpler models, such as basic and linear models, there are few parameters to tune (baseline bid, minimum and maximum bid value). For more complex models finding optimal parameter values is more difficult. Parameter optimisation is performed typically via a parameter grid search on a validation set. This involves iterating over the range of parameters and running the model with the specified bidding strategy on the validation set. The parameter or combination of parameters which performs optimally against the KPI is then stored to be used. This process scales badly as the number and range of parameters increases and is typically limited by computation power.

3. APPROACH AND EXPERIMENTS

3.1 Problem 2: Basic Bidding Strategies

In this section, we explain the work we have done in implementing a constant and random bidding strategy and then discuss the results which represent a baseline for the subsequent bidding strategies.

3.1.1 Constant Bidding

Constant bidding is a basic bidding strategy where the only parameter is the base bid price. We chose the base bid from the range 2 to 300 with steps of 2. The bid price was then compared to the pay price of each impression and if our constant bid price was greater than the pay price, the impression was won.

The set budget was 6250 CNY fen (6250000 CPM). The higher the CTR and lower the CPC and CPM, the better our bidding strategy.

The optimal base bid we found was 152 with 105309 impressions won and 625000.4 amount spent, getting the number of clicks as 85, with CTR as 0.08, CPM as 59349.53 and CPC as 73529.88 as can be seen from the figures 1 and 2.

	constants	imps_won	total_spend	clicks	CTR	CPM	CPC
44	90	130728.0	6250054.0	85	0.07	47809.6	73530.05
74	150	106459.0	6250030.0	85	0.08	58708.33	73529.76
75	152	105309.0	6250040.0	85	0.08	59349.53	73529.88
78	158	103767.0	6250033.0	85	0.08	60231.41	73529.8
79	160	103404.0	6250004.0	85	0.08	60442.57	73529.46

Figure 1: The data analysis for the bids giving the highest number of clicks for the constant bidding strategy.

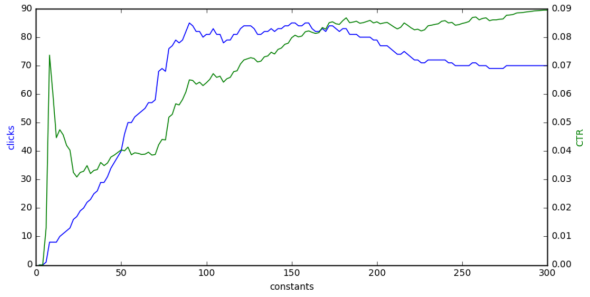

Figure 2: Distribution of clicks and CTR for the constant bidding strategy.

3.1.2 Random Bidding

For this bidding strategy, the upper bound is fixed and a bid price is selected randomly from 1 to that upperbound.

The optimal upper bound found was 240 with 110428 impressions won and 6250044.0 amount spent, getting the number of

clicks as 86, with CTR as 0.08, CPM as 56598.36 and CPC as 72674.93.

Due to the randomness of the bids, the clicks and CTR are not smooth as can be seen in the spikiness shown in figure 3.

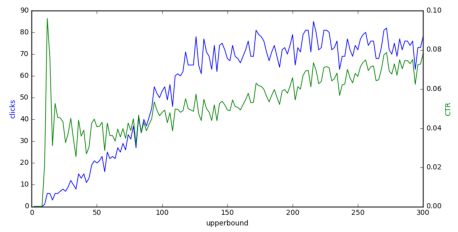

Figure 3: Distribution of clicks and CTR for the random bidding strategy.

3.2 pCTR estimation

pCTR is a valuable component of modeling complex bidding strategies. To perform pCTR estimation we implemented three different supervised learning models using engineered features and click labels from the training data. Each is described below.

3.2.1 *Feature engineering*

The first stage of building a pCTR model was to decide which features we would use for our models. The dataset consisted of 26 features (columns) corresponding to either user features or contextual features. We first pruned out the features that were either unique or not adding any meaningful signal.

We then transformed the remaining features into binary features using Scikit-learn's OneHotEncoder [13] After exploring the data we found one of the features to be a concatenation of two so we decided to split that feature into two. This is the 'useragent', which we found to consist of both the Operating System and the Browser. See figure 4 for the features we pruned and those we binarized.

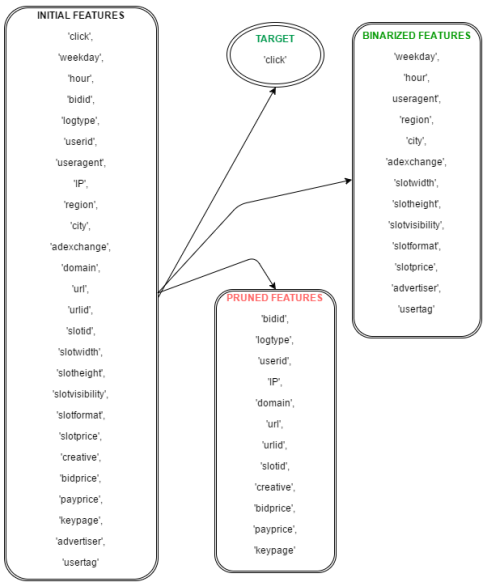

Figure 4: Feature Engineering

3.2.2 *Logistic regression*

Once we had our features we could feed them into our machine learning models.

Parameter tuning. We first tuned the logistic regression parameter C using a grid search with 10 fold cross validation and found the best C to be 0.001 as shown in figure 5 below.

	rank	C	mean time	std time	mean AUC	std AUC
0	1	0.001	60.332912	2.825690	0.829984	0.001177
1	2	0.01	117.853973	7.627416	0.829016	0.000698
2	3	0.1	177.978387	54.880737	0.826775	0.001130
3	4	1	318.417584	25.438356	0.825664	0.001370
4	5	10	543.256229	55.809014	0.825096	0.001132

Figure 5: Logistic Regression Parameter Tuning

Balancing. Once we had the best C, we used that in our model to find pCTR values. However there is the problem of imbalanced data which means the two classes into which we are trying to classify are not equally represented. We observed that in this particular dataset the ratio of clicks to no-clicks was 1325/1

There are several ways we found to balance data such as down sampling which refers to reducing the more frequent class, over sampling is when you do the opposite and sample more of the less frequent class. Other methods are such as generating synthetic data or collecting more real data which were not viable options.

An advantage of Scikit Learn's Logistic Regression [14] is that it automatically balances the data for you if you set the class weight parameter as balanced.

Once we had the data balanced we had to recalibrate the predicted probabilities so as to accurately represent the sparseness of the clicks. We used the following formula which is described in [15] to recalibrate pCTR.

$$q = \frac{p}{p + (1-p)/w} \tag{1}$$

where q is the recalibrated pCTR, p is the prediction in balanced space and w the balancing rate. To find the balancing rate w, we used the following formula which is described in the logistic regression docs in scikit learn[14]

$$ratio(y = 1|0) = nsamples/(nclasses * count(y = 1|0))$$

$$q = ratio(y = 1)/ratio(y = 0)$$

Feature Selection. We performed feature selection using Scikit Learn's Random Forest Classifier feature importances [16] with the results shown in figure 6. We found that the 6 features were not important (see second figure on the right).

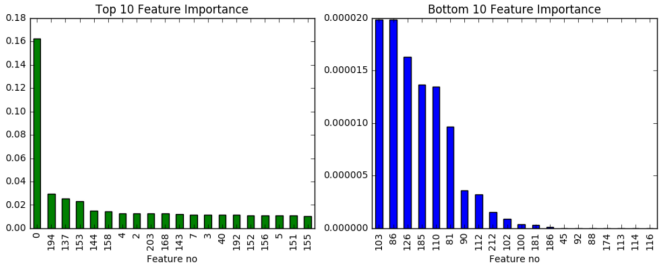

Figure 6: Feature importances

Prediction Accuracy. Next we then estimated the accuracy using the AUC score for the full subset of the features and for the reduced subset after feature selection. We found that removing the 6 unimportant features reduced the accuracy slightly (0.000002) even though they apparently had 0 importance. So we used all the features. The accuracy achieved was 0.8293.

3.2.3 Naive Bayes

Naive Bayes is one of the simplest methods that is widely used in practice to classify data. Its approach, counts the frequency of occurrence for each category, treats each category as independent and then applies Bayes Theory on the conditional probabilities [17].

We implemented Naive Bayes models using the Scikit Learn module [18]. We tested the model over a range of engineered features (section 3.2.1) and including the features of the best performing logistic regression model 3.2.2, so that we were able to compare the models.

Once the model had been fit, we used it estimate the pCTR for the validation set and measure the accuracy using the AUC metric. The results are as shown in section 4.1. As can be seen, with a optimal AUC accuracy of 0.7554, the approach did not perform as well as the logistic regression models.

3.2.4 Deep learning

Our final approach to calculate pCTR was to implement a mulit-layer (deep) perceptron neural network (NN). The NN models were implemented using the Python's TensorFlow module [19]. The input features used were the same as the best performing logistic regression model 3.2.2 so that we were able to compare the models.

The NN was configured as a typical multilayer perceptron. It received as input the 219 dimensioned engineered feature. It consisted of multiple hidden (1-3) layers with sigmoid activation functions on each node. The output was a softmaxed two dimensional vector which represented each category of either click and not clicked.

The NN was trained using 'uneven batching' (UB) which inputs data in batches containing at least a minimum of one clicked data instance per batch (with batch size varying from 10-100). Whilst batching is common in NN training to reduce computation time, uneven batch like this is a novel method presented by this study. Its purpose was to balance the data so that the NN didn't receive iteration after iteration of data without clicks, which would have occurred due to the sparsity of the data. Such sparsity would effectively reset the weights and bias to all configure probability 0 and NN would not be able to learn on the rare clicked impressions.

3.2.5 XGBOOST

XGBoost is similar to gradient boosting but more efficient. It is ten times faster as it does parallel computation. It has both a linear solver and tree learning algorithms. It is versatile as it supports classification, ranking and regression. It's a highly sophisticated algorithm, powerful enough to deal with all sorts of irregularities of data.
The most important factor behind the success of XGBoost is its scalability in all scenarios. Surprisingly, XGBoost didn't perform well with this dataset giving the accuracy score of 0.7284.

The results and analysis of each approach for pCTR is in section 4.1.

3.3 Problem 3: Linear Bidding Strategy

For the linear bidding strategy we had to employ the following formula for bidding:

$$bid = base\ bid \times \frac{pCTR}{average\ pCTR} \tag{2}$$

where base_bid is the optimization parameter and can be inferred from the training data statistics, pCTR is the pCTR values obtained from the learning model and avgCTR is the average CTR for the training data.

3.3.1 Linear bidding strategy results

The results we obtained from using the linear bidding strategy with a budget of 6250 CNY fen (6250000 CPM) are shown in figures 7 and 8 shown below.

From figure 7 we can see base bid 104 produced the best results as it not only had the highest clicks obtained (168) but also had the best CPC and CPM in the bunch while scoring a good CTR of 0.15%

Figure 8 shows us the relationship between the base bid and the clicks obtained. We observed that the mid range base bids had the highest clicks and this makes sense as the linearity of the bidding strategy ensures that the more the base bid the more clicks you get, but this then gets capped due to the budget constraint and thats why it starts to go down.

	bid	bidding_strategy	imps_won	total_spend	clicks	CTR	CPM	CPC
51	104	linear	113686.0	5547127.0	168	0.15	48793.4	33018.61
52	106	linear	115334.0	5655144.0	168	0.15	49032.76	33661.57
53	108	linear	116974.0	5766663.0	168	0.14	49298.67	34325.38
54	110	linear	118570.0	5873097.0	168	0.14	49532.74	34958.91
55	112	linear	120180.0	5977030.0	168	0.14	49733.98	35577.56
56	114	linear	121715.0	6084171.0	168	0.14	49987.03	36215.3
57	116	linear	123172.0	6183731.0	168	0.14	50204.03	36807.92

Figure 7: Bids with the highest click count obtained

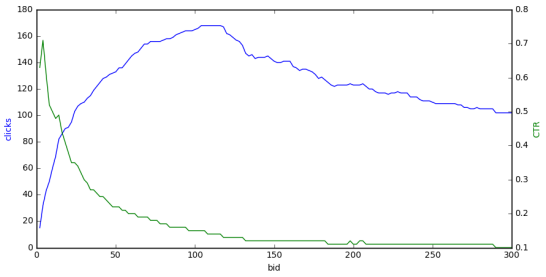

Figure 8: Distribution of clicks and CTR for the linear bidding strategy

3.4 Problem 5: Combined Bidding Strategy

For this part, we had the choice to combine our 3 individual bidding strategies to form our final strategy or to pick the best of the 3 as our final strategy.

We individually came with with 3 different strategies with varying performance on the validation set against the predefined KPIs and set budget of 6250 CNY fen (6250000CPM). The strategies are explained below.

The first strategy is the exponential strategy and does the following:

$$bid = base\ bid \times \exp\left(\frac{pCTR}{average\ pCTR}\right) \quad (3)$$

The second is the squared and does the following:

$$bid = base\ bid \times \left(\frac{pCTR}{average\ pCTR}\right)^2 \quad (4)$$

The third strategy we implemented which we called the 'gate' is a logical gate which checks whether the pCTR norm i.e. (pCTR / avgCTR) is greater than a certain threshold, where this threshold is the parameter to the function. If it is greater than the threshold then we bid the maximum bid and if not we do not bid.

We tried to combine the strategies but observed that they would perform poorly when combined versus on their own. So we analyzed the results of the 3 strategies and chose the best performing one as can be seen in figure 9.

	imps_won	total_spend	clicks	CTR	CPM	CPC
exponential	138421	6199575.0	171	0.1235	44787.82	36254.82
squared	95651	6063093.0	173	0.1809	63387.66	35046.78
gate	68788	6234152.0	173	0.2515	90628.48	36035.56

Figure 9: Best performing individual bidding strategy

We found the best strategy was the gate strategy as it had both the highest click count and the highest CTR which are our 2 main KPIs. Once we had decided on our best bidding strategy, we made our predictions on the test set using this strategy.

4. ANALYSIS

4.1 pCTR Results

pCTR is a an important determinant in real-time bidding strategies. As such, a substantial component of this study was in the producing of models to find accurate estimated pCTR. Of the four separate approaches we designed, implemented and optimized for pCTR estimation, the logistic regression model performed the best and was used in the subsequent bidding strategies. Table 1 shows the comparative pCTR performance of each of the models measured against the AUC metric.

pCTR Model	Parameters, Features	AUC
Logistic Regression	C=0.001, full set of engineered features	0.8299
Deep Learning 1	1 layered (sigmoid activation function) MLP with softmax cross entropy loss	0.7812
Deep Learning 2	2 layered (sigmoid,sigmoid activation) MLP with softmax cross entropy loss	0.8012
Deep Learning 3	3 layered (tanh,sigmoid,sigmoid activation) MLP with softmax cross entropy loss	0.7311
Naive Bayes 1	Default parameters, 5 features (advertiser, weekday, hour, adexchange, slot visibility)	0.7134
Naive Bayes 2	Default parameters, 10 features (advertiser, weekday, hour, adexchange, slot visibility,city,slot height, slot width, usertag, useragent)	0.6723
Naive Bayes 3	Default parameters, Full set of engineered features	0.7554
XGBoost	Default parameters, full set of features encoded with DictVectorizer	0.7284

Table 1: pCTR model summary

Below is our analysis of each approach and a discussion of the results pCTR.

4.1.1 Feature engineering

Crucial to the logistic regression and other pCTR models success was the feature engineering. Whilst label-encoding and other preprocessing where required to input the data, we were also able to enrich the data through feature engineering . We were able to extract additional information from some features which supported our inference and improved our pCTR.

As can be shown from table 1, by comparing the models which contained the full set of engineered features versus the models with only the features in their 'vanilla' form, we were able to increase relative performance.

4.1.2 Logistic regression

The AUC accuracy for the logistic regression model of 0.83 is a reasonable level of accuracy. However, it still indicated that there is substantial inaccuracy or noise in the data.

The noise is likely to be caused by factors which are not captured by the impression level features. These include a whole range of factors ranging from impression's properties not listed (the display content and aesthetics) the user's behavior (mood and location or accidental clicking). In this context, the data we have been provided with is actually a very small subset of the total determinant variables on whether an display advertisement impression is clicked on.

We believe that this noise would account for a substantial part of the errors in pCTR and thus with the provided data and in this context, an accuracy of 0.83 can be considered satisfactory.

4.1.3 Naive Bayes

Comparatively, the Naive Bayes approach performed poorly. It was the worst of the three approaches, over 7% below best performing model.

This is likely because of Naive Bayes' struggles where there is very small data counts, like the sparse number of clicks in the data. This mean that, for some categories, there will be very few or zeros clicks. This means in the Naive Bayes there could be a zero probability for some combination of categories. The model adjusts for this by applying smoothing, however this is creates inaccuracy.

The benefits of the NB approach is that the average training was very quick. This may be advantageous for RTB in very dynamic environments where online retraining is required.

4.1.4 XGBoost

The XGBoost did not perform as well as we expected it too. In theory it should do better than the logistic regression model as it can better model the nonlinearity in the data. An explanation for this might be that, the parameters used were not ideal and more needed to be done in order to find the optimal parameters. Also, instead of using discreet features XGBoost would have benefitted from engineering features as continuous values using creative approaches.

4.1.5 Deep learning

The deep learning model performed only slightly behind the logistic regression model and better than the other models. This result is disappointing, as the superior expressiveness of a NN was unable to outperform the comparatively restrictive logistic regression model. But it is also encouraging as we were able to implement a 'state-of-the-art', highly complex model for pCTR estimation.

In terms of implementation time (coding, training, parameter optimisation), the approach is far more expensive than the

others. TensorFlow and other modules help by providing out-of-the-box back-propagation, to reduce coding time, however the build time is still long as is optimization. Processing cost which is effectively unbounded is incomparable to the simpler models.

There is little literature available on NN design for pCTR [6], but this study shows that the data can be modeled with deep learning. In particular, the uneven batching method worked well in the condition of extremely sparse data. It enabled the MLP to learn rapidly and infer to a near optimal performance. Given the complexity of initializing and training a neural network, this is a good achievement. Whilst it has not outperformed the logistic regression model here, its capacity and potential with bigger data sources is higher.

In this study, we attempted 3 optimized deep learning models but given the enormous and uneven parameter space, it is impossible to say that the models are fully optimized and it is certainly possible that were they fully optimized they would have outperformed the logistic regression model.

The second issue that we hypothesize is that there was insufficient data to properly train the NN. Usually, 2 million data points would be sufficient to train a NN, however because there were only 2000 clicked impressions, this in principle made the data much smaller. Due to the way NN's learn, with forward and back propagation, this a bigger problem than for logistic regression models.

4.2 Real-time bidding strategy

After doing all the different bidding strategies, the results of their performances are shown in figures 10 and 11. We implemented the basic bidding strategies (constant and random), the linear strategy, the ORTB strategy [5] and then our 3 individual strategies.

	imps_won	total_spend	clicks	CTR	CPM	CPC
constant	103404	6250004.0	85	0.0822	60442.57	73529.46
random	121232	6250056.0	85	0.0701	51554.51	73530.07
linear	113686	5547127.0	168	0.1478	48793.4	33018.61
ortb	144370	6133571.0	168	0.1164	42485.08	36509.35
exponential	138421	6199575.0	171	0.1235	44787.82	36254.82
squared	95651	6063093.0	173	0.1809	63387.66	35046.78
gate	68788	6234152.0	173	0.2515	90628.48	36035.56

Figure 10: Comparison of all bidding strategies

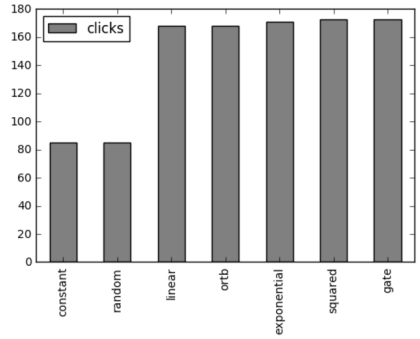

Figure 11: Performance of all bidding strategies on click KPI

As is clear from the experiments and the results we obtained, the basic bidding strategies i.e. constant and random bidding were not the best performing. They act as good baselines for the more advanced strategies. The linear bidding performed much better than the basic, while the non linear strategies were the best performing albeit the difference between non linear and linear strategies were not that big.

Compared to optimized constant or random bidding strategies, by including pCTR in the linear and subsequent strategies, we have nearly doubled all of the key performance metrics (CTR, CPC, count-per-click). This a substantial performance gain and in practice these improved bidding strategies would save buyers money and improve their display advertisement exposure. It also further validates the importance of the pCTR.

Below is an analysis of our three most interesting bidding strategies.

4.2.1 *Linear bidding*

Linear bidding strategy was an easy to optimize model, and as an initial bidding strategy using pCTR provided substantial gains compared to the baseline models (constant and random bidding).

4.2.2 *ORTB*

We implemented one of the ORTB non linear functions from [5]. The formula is shown below:

$$w(b(\theta)) = \frac{b(\theta)}{c + b(\theta)} \tag{5}$$

The results were not as expected as we were not able to surpass the performance of the linear bidding strategy as shown in figure 10. We had hoped to improve on the linear bidding strategy as was shown in the paper [5]. The reasons could be either one or both of the 2 parameters, C and λ were not adequately tuned but this is not a very good explanation as a thorough grid search was performed with a combination of 20 candidates for each of the 2 parameters with the best parameters emerging as C(6) and $\lambda(5\times10^{-7})$. Another possible explanation could be the formula's lack of versatility with different types of datasets but we did not explore this any further.

4.2.3 *'Gate'*

The gate is a very simple approach that is governed by one idea: if the pCTR is above a threshold value, the impression will be clicked and it should be won. As such, it relies entirely on the pCTR value and an optimized threshold. There is no consideration to price and it carries an additional risk exposure.

Due to the scarcity of the clicks and that the key metrics of a bidding strategy is CTR, the payprice of the auction is not really a factor. The average cost per click, is over 100 times the maximum auction price. As such, a greedy algorithm that sees a impression it predicts likely (to be clicked) it should win it. And by bidding nothing for the impressions it thinks are unlikely, it does not win impressions it predicts unlikely.

It would be more difficult to implement this strategy in practice or in other contexts, as the maximum bid would not be capped at 300 as in this study. You may have to bid even higher to win those pCTR rated impressions which increases the cost of impressions won and risk of won impression not being clicked.

5. CONCLUSION

In this report we analyzed different bidding strategies for the problem of RTB. A clear finding is that effective bidding functions rely on accurate estimated pCTR. The increase in performance between models with pCTR versus those without is substantial (+310% CTR and -50% eCPC).

Its believed that further improvements in pCTR either through increased training data or enhanced impression features could

see further improvements in the performance of bidding strategies. Beyond increasing the data supply, this study has also shown that deep learning can be used to find pCTR.

This study has shown that the impression level data can be enriched by effective feature engineering. This not only makes the data suitable for modelling but also extracts additional information like the two components of 'useragent' which leads to improved pCTR accuracy.

This study suggests that there is a limit to the accuracy of the estimate pCTR available from the features in the dataset. A large amount of clicks or non-clicks can be attributed to noise in the data which is not able to modelled. This will have contributed to the limit of pCTR accuracy of around 80% AUC.

Beyond pCTR, this challenge is further complicated by factors affecting bidding strategies and bidding strategy optimization. Firstly, the variety of KPIs makes generalization of optimization difficult. Ultimately, the performance measures determine the optimal bidding function and this means that even though you might get the best CTR using a certain strategy you still have to consider other things such as the number of clicks as well as the CPC and CPM. This makes the problem a subjective one with the outcome depending on the advertiser's perspective and campaign goals.

We found that basic bidding strategies such as constant and random bidding perform conservatively well but are not the best and this makes sense as they do not take into consideration the user or context features of bid requests. Linear bidding does much better than basic bidding and non-linear bidding strategies perform the best of all.

In future, we would like to look at better feature representations such as using word2vec [20] or GloVe [21] embeddings for the features. With these, we would vectorize and embed impression features which would provide better representations for modelling. This could lead to bettering the pCTR and the deep learning will surely improve giving us a better pCTR estimation and consequently better bidding performance.

Also, we would like to look at other non-linear bidding strategies by adding different forms of non linearity to the bidding strategy. Finally we would like to define pCPM estimators as well and use them in bidding strategies alongside pCTR so that we can increase clicks while reducing spend.

With deep learning there is certainly potential for further work on pCTR by implementing enhanced models and using bigger and richer datasets. We would also attempt a regression neural network model which would learn on the click divided by payprice (click/payprice). This would add the cost of the click into the prediction and the model would then be able to identify not only if the model was clicked but also what price to bid, which would be useful in more dynamic bidding environments.

Further to this, reinforcement learning (section 2.4.5) would be an interesting approach. By designing agents which learn on live data and then bid based on these observation, bidding strategies could become more dynamic and be flexible to temporal market conditions (i.e. times of high CTR or high payprice).

6. CODE DESCRIPTION

All the source code from this study can be found in the group's collaborated GitHub repository at the following address: https://github.com/SaidAbdullahi/web_econ_coursework. See web_econ_group_part.ipynb.

7. REFERENCES

[1] http://www.measurementnow.net www.measurementnow.net. Making Measurement Make Sense. Cited 15 March 2013.

[2] S. Yuan, J. Wang, and X. Zhao. Real-time bidding for online advertising: measurement and analysis. In ADKDD, 2013.

[3] Perlich, C., Dalessandro, B., Hook, R., Stitelman, O., Raeder, T., Provost, F. (2012, August). Bid optimizing and inventory scoring in targeted online advertising. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 804-812). ACM.

[4] Cai, H., Ren, K., Zhang, W., Malialis, K., Wang, J., Yu, Y., & Guo, D. (2017). Real-Time Bidding by Reinforcement Learning in Display Advertising. arXiv preprint arXiv:1701.02490.

[5] Zhang, W., Yuan, S., Wang, J. (2014, August). Optimal real-time bidding for display advertising. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 1077-1086). ACM. Chicago

[6] Zhang, W., Du, T., Wang, J. (2016, March). Deep learning over multi-field categorical data. In European Conference on Information Retrieval (pp. 45-57). Springer International Publishing.

[7] Bengio, Y.: Learning deep architectures for ai. Foundations and trendsR in Machine Learning 2(1), 1-127 (2009)

[8] Wang, X., Li, W., Cui, Y., Zhang, R., Mao, J.: Click-through rate estimation for rare events in online advertising. Online Multimedia Advertising: Techniques and Technologies pp. 1-12 (2010)

[9] J. Wang, S. Yuan, X. Shen, and S. Seljan. Real-time bidding: A new frontier of computational advertising research. In CIKM Tutorial, 2013.

[10] Lee, K.c., Orten, B., Dasdan, A., Li, W.: Estimating conversion rate in display advertising from past performance data. In: KDD. pp. 768-776. ACM (2012)

[11] A. Broder, M. Fontoura, V. Josifovski, and L. Riedel. A semantic approach to contextual advertising. In SIGIR, 2007.

[12] J. Jaworska and M. Sydow. Behavioural targeting in online advertising: An empirical study. In WISE. 2008.

[13] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

[14] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[15] He, Xinran, et al. "Practical lessons from predicting clicks on ads at facebook." Proceedings of the Eighth International Workshop on Data Mining for Online Advertising. ACM, 2014.

[16] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[17] Barber, David. Bayesian reasoning and machine learning 203-213. Cambridge University Press, 2012.

[18] http://scikit-learn.org/stable/modules/naive_bayes.html

[19] https://www.tensorflow.org/api_docs/

[20] https://code.google.com/archive/p/word2vec/

[21] https://nlp.stanford.edu/projects/glove/