Multi-Agent Artificial Intelligence Project (Group 3)

François-Xavier Aubet * Sylvanus Quek * Sorawit Saengkyongam *

1. INTRODUCTION

Since 2009, computational advertisement has increasingly been conducted through Real-Time Bidding (RTB). [2] This system allows publishers to sell individual ad impressions to advertisers using real-time auctions. The advertisers all receive a description of the ad impression available for auction which allows them to evaluate and decide how much to bid on it. The ad exchange gathers the bids and selects as winner the one with the highest bid price. The winning advertiser is then permitted to show its ad. In this work, we only consider second-price auctions where the winning pays the price of the second highest bid. The main goal for an advertiser is to maximize the number of clicks on the ads it displays. This demands an accurate assessment of the impressions' descriptions and estimation of how likely the ad would be clicked on, i.e. Click Through Rate (CTR) [12].

In this work, we use ad impressions collected from the iPinYou logs [20] to design and evaluate different bidding strategies. We evaluate the strategies on the validation set using a fixed budget of 6,250 CNY fen under two distinct settings. First, the algorithms compete against the pay prices of the impressions collected in the data set giving us winning criteria #1: auction is won if $bid \geq payprice$. Second, the algorithms compete against one another using winning criteria #2: auction is won if $bid \geq \max(payprice, otherSubmittedBids)$.

We first review related work and perform a detailed analysis of the data set. Following that, basic baseline strategies are created and we investigate the multi-agent setting for one of them. In the later sections, we develop and evaluate linear and non-linear bidding strategies. Finally, we design a multi-agent and compare all of the strategies in a multi-agent environment. The implementation of the algorithms created in this paper is made publicly available.[1]

2. LITERATURE REVIEW

In this section, we review various previous work on aspects of RTB. The early works in RTB focused mainly on utility estimation and utility-based bidding strategies. Utility estimation often boils down to a user response prediction such as CTR and conversion rate (CVR) predictions. CTR and CVR predictions are typically framed as binary classification problems. Several machine learning algorithms for classification such as logistic regression [7], Gradient Boosting Trees [14] and Factorization Machine [6] have successfully been applied in utility estimation. With the estimated utility, utility-based bidding strategies then exploit the utility prediction to perform a bid. Auction theory suggests that truthful bidding, where we bid according to the true utility value of an impression, is the optimal strategy in second price auction [7]. However, truthful bidding can be sub-optimal if we consider budget and volume (number of impressions) constraints [19]. The non-truthful linear bidding strategy was then proposed by [19] where the bid price is calculated by CTR/CVR estimation multiplied by a constant parameter which is tuned according to the budget constraint. Later on, [19] suggested a non-linear bidding strategy. The authors derived a closed-form solution in which the bidding function is a concave function instead of a linear function. A lift-based bidding strategy was proposed in [16] where the bid price is decided based on the utility lift rather than absolute utility value.

Another line of works involve strategies that rely on both utility and cost estimation. Cost estimation aims to predict the market price distribution given features of an impression. The challenge is that advertisers only gain market price information from impressions won. Censored regression has been adopted to address this censoring problem [15]. The authors in [8] combined cost and utility estimations to define an efficiency (*return on investment*) of an impression and used it to come up with the final bid price. [13] proposed a joint optimization approach which jointly learns utility estimation, cost estimation and bidding function as a whole.

All of the aforementioned works assume that the environment is stationary. In recent years, the works in RTB have pivoted from stationary to non-stationary environments where the changes in the strategy of one advertiser would affect the strategies of other advertisers. A game-theoretical approach has been recently proposed by [5]. The authors solved RTB with multi-agent reinforcement learning by modeling interactions of bidding agents. More recently, [3] formulated RTB as a generalized mean-field-game and proposed an alternating approximating Q-learning algorithm with Boltzmann policy which demonstrated superior numerical performance.

3. DATA EXPLORATION

The data set used in this paper is a subsample of the iPinYou logs [20]. It only contains impressions and was split into train/validation/test sets of 80/10/10 proportions. Out of the 25 fields available for each record, only 22 of them can be utilized for deciding bid prices.

Purportedly, `slotprice` acts as the minimum bid price for impression. However, approximately 0.5% of the train and validation sets contains impressions where the `payprice` is less than the `slotprice`. This phenomenon is present in both the train and validation sets. Notably, these impressions have $avgCTR$s of 0.06% which is comparable to the global average of 0.07%. For Winning criterion #1, we cannot discard these impressions as winning them would yield more clicks.

Table 1 shows the relevant metrics for train and validation sets. CTR is slightly lower in the validation set and

* All authors contributed equally to this work.

[1] https://github.com/Francois-Aubet/Multi-Agent-AI-bidding

split	Impr.	Clicks	Cost	CTR	CPM	CPC
train	2,430,981	1,793	189,984	0.074%	78.15	105.96
validation	303,925	202	23,777	0.066%	78.23	117.71
advertiser	Impr.	Clicks	Cost	CTR	CPM	CPC
1458	492,353	385	33,969	0.078%	68.99	88.23
2259	133,673	43	12,428	0.032%	92.97	289.03
2261	110,122	36	9,874	0.033%	89.66	274.27
2821	211,366	131	18,828	0.062%	89.08	143.73
2997	49,829	217	3,129	0.435%	62.80	14.42
3358	264,956	202	22,447	0.076%	84.72	111.12
3386	455,041	320	34,932	0.070%	76.77	109.16
3427	402,806	272	30,459	0.067%	75.62	111.985
3476	310,835	187	23,919	0.060%	76.95	127.91

Table 1: Summary statistics. Bottom table is for training set only.

exhibits a higher Cost Per Click (CPC). The impressions can be further classified by the 9 advertisers. In the bottom table, we computed the same metrics for the training set but grouped them by advertisers 1. Evidently, some advertisers like '2997' do receive higher CTRs which could be attributed to the industries they belong to. Industries that are more aligned with online consumers' interests are likely to be better-received and therefore have more clicks.

3.1 Cardinality

Most of the 22 fields provided are categorical data with high cardinality. This can be problematic for downstream models to handle. Hence, we dropped fields that have a near 1-to-1 mapping with impressions as these are likely to be uninformative.

To enrich the remaining fields, we engineered `os` and `browser` from `useragent` by splitting along '_'. A "bag of tags" feature (inspired from bag of words) was constructed by splitting `usertag` along ',' yielding 68 unique user segments. `IP` was also split into 4 tokens along '.'. Figure 1 illustrates the cardinality of each feature.

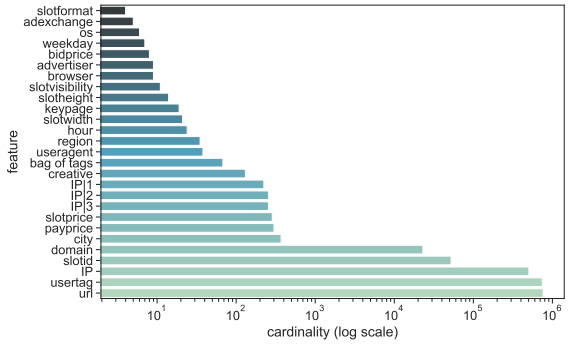


Figure 1: Cardinality of features considered.

3.2 User Feedback

Figure 2 indicates some relationship between CTR which could be exploited for prediction. Notably, mobile devices (i.e. iOS with safari (default browser on iOS) and android) have the highest CTR. Varying CTR rates for each tag from "bag of tags" suggests that they can be leveraged in building a bidding strategy.

4. BASIC BIDDING STRATEGIES

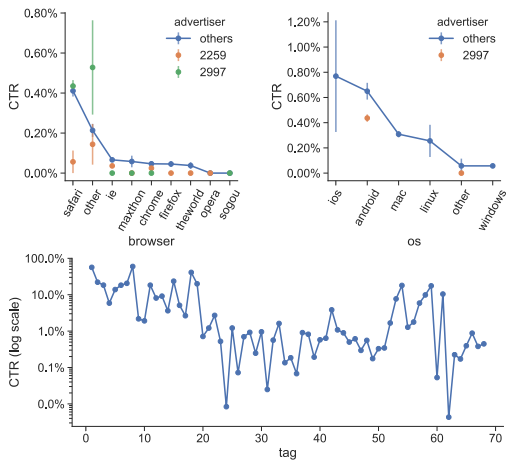


Figure 2: CTR for engineered features. Individual advertisers plotted do not have impressions across all categories in the training set whereas the remaining advertisers grouped under 'others' do.

First we consider basic bidding strategies that will be used as base-lines for more complex strategies in subsequent sections. We evaluate each bidding strategy on the validation set using the following metrics: number of clicks, CTR, total cost spent, average Cost Per Mille (CPM), and average CPC. In order to evaluate the bidding strategies with an appropriate budget on the validation se,t we multiply the budget per impression on the test set (6250 CNY fen divided by the number of impressions in the test set) by the number of impressions in the validation set. Since the validation set is only 0.18% bigger than the test set, the scaling is marginal. That said, this difference is important on bootstrapped sets used for training and tuning.

4.1 Constant bidding

The first bidding strategy that we investigate consists in bidding a constant price for every impression. We denote the value of the constant bid that we aim to learn as *base_bid*. In order to find it, we employed the Cross Entropy method (CE) for fast policy search [9].

4.1.1 Cross Entropy method (CE) for policy search

CE method is a black-box optimization algorithm for policy search. Intuitively, CE starts with a pre-defined distribution over policy parameter vectors. This distribution is parameterized by the initial parameters $\boldsymbol{\theta_0}$. It then samples K samples of policy parameter vectors from this distribution. Let $\mathbf{b}_1, \ldots \mathbf{b}_K$ denote these samples. It evaluates these K sampled policies and obtains the corresponding returns. It then picks the elite samples which are the E top performing policy parameter vectors (for some constant $E < K$) and fits an MLE estimator ($\boldsymbol{\theta}_{mle}$) to these elite parameter vectors. The previous $\boldsymbol{\theta_0}$ are then replaced by $\boldsymbol{\theta}_{mle}$ and this process is then repeated. For a constant bidding strategy, the policy parameter is just a constant bid (*base_bid*) and we assume the *base_bid* to be distributed as a Log-normal distribution. We present pseudocode for CE in Algorithm 1.

Using CE, we found that the *base_bid* achieving the highest number of click was $base\_bid = 82.083$.

Clicks	CTR	Cost	CPM	CPC	Impr.
66	0.000472	6261.27	44.808	94.867	139735

Table 2: Validation set results using the constant bid found.

Input:

1. Initial parameters for a *base_bid* distribution (Log-normal): μ, σ
2. Population size K
3. Elite size $E < K$

initialization;
while *true* **do**
  for $k = 1$ **to** K **do**
    $b_k \sim \text{Lognormal}(\mu, \sigma)$;
    $\hat{V}_k = evaluate(b_k)$;
  end
  $sort((b_1, \hat{V}_1), \ldots, (b_K, \hat{V}_K), descending)$;
  $\mu = \frac{1}{E} \sum_{k=1}^{E} \log b_k$;
  $\sigma = \frac{1}{E} \sum_{k=1}^{E} (\log b_k - \mu)^2$;
end

Algorithm 1: Cross-Entropy (CE) for Policy Search.

4.2 Random bidding

The second bidding strategy that we investigate consists in drawing the bid price of each impression from a uniform random distribution. We have to find the lower and upper bound of this distribution.

To do this, we keep the price that we had found for the constant bidding as the mean of our uniform distribution and learn λ the distance between the lower bound and the mean:

$$bid = \text{uniform\_rand}(base\_bid - \lambda, base\_bid + \lambda).$$

We use the CE method (4.1.1) with a log-normal distribution again to find the best λ. We find that $\lambda = 10.08$ performed the best.

Clicks	CTR	Cost	CPM	CPC	Impr.
66.65	0.000463	6261.28	43.621	94.12	143538.15

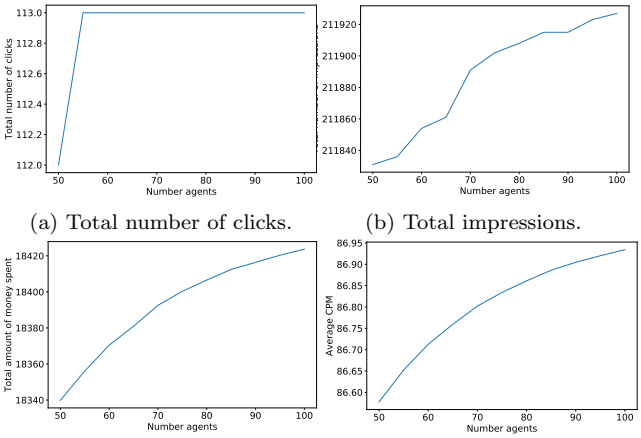
Table 3: The results on the validation set averaged over 20 runs using the λ we found.

4.3 Multi-agent random bidding

We now investigate the multi-agent case. We create between 50 and 100 random bidding agents each using the range obtained above for the uniform distribution.

Before commenting about the above number of agents, we observe that a single random agent spends all his budget during one run through the data set. However, from 3 or 4 random agents on, they finish the run through the data set without using all their budgets. This means that 50 agents do no use all their budgets either.

We observe from figure 3b that the total number of impressions won by the random agent steadily increases with the number of agents used. The reason for this is that the more agents there are, the higher the chance that the aggregated bids are greater than the payprice on certain bids.



(a) Total number of clicks. (b) Total impressions.

(c) Total spent. (d) average CPM

Figure 3: Evolution of different metrics for 50 to 100 competing random agents.

However, they cannot bid higher than the upper bound of the uniform distribution, hence there is a maximum number of clicks that can be obtained. And we think that this was reached at 55 agents as shown in figure 3a.

We observe that the total amount of money spent steadily increases as shown in figure 3c. One might suspect that this is due to the increasing number of impressions won. However, we also notice that the average CPM increases too, cf. figure 3d. This is caused by the fact that the more agents there are, the higher the chance that two bids are going to be very close to the upper bound. Hence, this pushes the average payprice towards the upper bound. This is an interesting phenomena that could show that with more agents following the same strategy, the less profitable it becomes for each agent.

We could change the lower and upper bounds of the uniform random distribution used by the agents in order to win more impressions and hence more clicks. The optimal way to do this would be to choose the bounds and the mean such that the budgets of all agents run out before all impressions are shown. We do not investigate this because this would require us to have different bounds for each number of agents considered.

5. LINEAR BIDDING STRATEGY

In this section, we consider linear bidding strategies where we assume a linear bidding function w.r.t. predicted CTR and a scaling parameter *base_bid*.

$$b(\mathbf{x}) = base\_bid \times \frac{pCTR(\mathbf{x})}{avgCTR}$$

where $\mathbf{x}$ is an impression feature vector, $pCTR(\mathbf{x})$ is predicted CTR and $avgCTR$ is the average CTR over all impressions. We consider both linear and non-linear CTR predictors in the following subsections. The scaling parameter *base_bid* was tuned using the CE method (4.1.1) on training data.

5.1 Linear predictor

CTR was modelled using regularized logistic regression from `scikit-learn`. The optimal regularization strength was found to be 0.001 using a random train/test split from the training set. Following that, features were pruned to assuage over-fitting ills. From the feature importance depicted in figure 4, we decided to prune the bottom two.

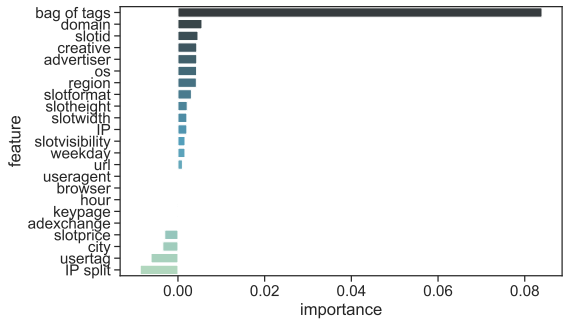


Figure 4: Drop-column importance [11].

For each selected feature, unique categories (including `NaN`s) are assigned integer indices. After training, any unseen categories are ignored. These features are then one-hot encoded for logistic regression. We found one-hot encoding to be the most effective as treating features like `slotprice` as numerical and calculating its z-score garnered fewer clicks on validation set.

As the data set is heavily imbalanced (only 0.07% of impressions have clicks in train set), impressions with no clicks are assigned lower class weights in the model fitting stage. Thus, mistakes on these are weighed less in the loss function and results in learned parameters that are more aligned with our main metric of achieving the most clicks.

After fitting, we still have to calibrate the probabilities as the model now predicts a higher CTR than empirically observed. The transformation is stated below with w as the negative downsampling rate [4]:

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

5.2 Non-linear predictors

The predictive power of Logistic regression is often limited due to its assumption of a linear relationship between predictor variables and the logit of the outcome variable. We overcome this limitation by considering non-linear predictors for CTR prediction. We adopted Gradient Boosting Trees (GBTs) using `XGBoost` library as our non-linear CTR model. The features were selected and pre-processed with the same procedure described in section 5.1. The hyper-parameters were tuned using train/test split from the training data.

To handle imbalanced data, we down-sampled the negative class (impression without click) to 10% of the original size. The down-sampling not only sped up the training process but also significantly improved performance of the model.

Despite its good result in accuracy, precision and AUC, GBTs typically yield poorly calibrated probability predictions because they do not directly optimize cross-entropy (log-loss) [10]. To address this matter, we followed [17] in applying Isotonic Regression, a non-parametric calibration method, to obtain calibrated probabilities.

Predictor	Clicks	CTR	Cost	CPM	CPC	Impr.	AUC
Logistic	166	0.00148	6091.8	54.38	36.69	112016	0.865
GBTs	170	0.00143	6252.3	52.77	36.77	118478	0.888

Table 4: Linear bidding strategy results on validation set.

6. NON-LINEAR BIDDING STRATEGY

6.1 Closed-form prediction

An easy extension from a linear bidding strategy to a non-linear one is to replace the linear bidding function with a non-linear one. We followed the solution proposed by [19] where the authors derived a closed-form solution for non-linear bidding strategy. The derived bidding function (a function that maps CTR prediction to the final bid price) is a concave function instead of a linear function. In particular, we implemented the following non-linear bidding function.

$$b(\mathbf{x}) = \sqrt{\frac{c}{\lambda} pCTR(\mathbf{x}) + c^2} - c$$

where $\mathbf{x}$ is an impression feature vector, $pCTR(\mathbf{x})$ is predicted CTR, c and λ are parameters for the algorithm. Similar to the scaling parameter *base_bid* in the linear strategy, we tuned c and λ using the CE method (4.1.1) on training data.

Since the formula directly depends on the CTR estimator's performance, we omit the weaker logistic CTR and only considered GBTs CTR in table 5.

Clicks	CTR	Cost	CPM	CPC	Impr.
167	0.00138	5752.15	47.867	34.444	120167

Table 5: Closed-form non-linear strategy results on validation set.

6.2 Using prediction uncertainty (CDF)

The aforementioned methods only utilize point estimates of $pCTR$. Here, we devise a novel approach that accounts for uncertainties in bid price determination. Uncertainties can be obtained through Bayesian inference but due to the sheer size and dimensionality of the data, it is too slow and memory intensive. Instead, we approximate by fitting n models on bootstrapped samples of the training data. Standard deviations of $pCTR$ on the validation set serve as an approximation of the true uncertainties.

As $pCTR(\mathbf{x}) \in [0, 1]$, beta distributions are fitted using method of moments. The bid price is then computed by relying on the fitted Cumulative Distribution Function (CDF):

$$\begin{aligned} b(\mathbf{x}) &= base\_bid \times \mathbb{P}\left[pCTR(\mathbf{x}) > avgCTR\right] + min\_bid \\ &= base\_bid \times \left[1 - F_{pCTR(\mathbf{x})}(avgCTR)\right] + min\_bid. \end{aligned}$$

A minimum bid parameter was added as we found that the probabilities are typically underestimated and consequently, the budget tends to be under-spent. Both *base_bid* and *min_bid* were tuned using CE.

10 logistic regression and 5 GBTs CTR models were trained to obtain uncertainty estimates. However, evidence of any improvement over their linear bidding counterparts remain

CTR Model	Clicks	CTR	Cost	CPM	CPC	Impr.
Logistic	157	0.00144	5888.9	54.19	37.50	108661
GBTs	171	0.00222	5955.3	77.43	34.82	76912

Table 6: CDF results on validation set.

unclear (table 6). CDF with logistic CTR performed worse while GBTs CTR engendered an additional click, higher CTR and lower CPC.

One possible explanation is that bootstrapping produced more dissimilar and less correlated GBTs than logistic regression did by training on a smaller fraction of the bootstrapped data. Thus, prediction uncertainties from logistic regression were grossly underestimated, resulting in higher CDF probabilities and over-fitting of hyper-parameters. This is supported by the observation that *min_bid* is 24.3 for logistic CTR whereas it was only 1.66 for GBTs.

6.3 Using winning price prediction (PRUD)

We consider another non-linear bidding strategy called PRUD, proposed in [8]. Unlike the two non-linear strategies mentioned earlier, PRUD incorporates additionally a winning price (WP) predictor in the bidding strategy. Specifically, the algorithm first defines a *bid efficiency* ρ of an impression $\mathbf{x}$ as $\rho(\mathbf{x}) = pCTR(\mathbf{x})/pWP(\mathbf{x})$, where $pCTR(\mathbf{x})$, and $pWP(\mathbf{x})$ are the predicted CTR and predicted WP for the impression $\mathbf{x}$. The bid efficiency essentially reflects *return on investment* of the given impression. After defining the bid efficiency for an impression, PRUD then employs the following algorithm to produce the final bid value.

Parameters :

- ρ_{cut} - a bid efficiency cutoff
- L - a lift value

Input: $\mathbf{x}$ - an impression feature vector
Output: $b(\mathbf{x})$ - bid value for $\mathbf{x}$
State variable: B_{cur} - Current budget available, initialized as a total budget

```
if ρ(x) ≥ ρ_cut then
    if pWP(x) ≤ B_cur then
        b(x) := pWP(x) + L
        B_cur := B_cur − P(x), where P(x) is an exact
          winning price.
    else
        b(x) := 0
    end
else
    b(x) := 0
end
```

Algorithm 2: PRUD

The parameters ρ_{cut} and L were tuned using train/test split on training data. Both CTR and WP predictors were Gradient Boosted Trees trained using `Xgboost` library. Table 7 outlines the performance of PRUD on validation data. Although the performance of the algorithm is on par with the best linear bidding strategy in terms of number of clicks obtained, the idea of this approach could be beneficial in the multi-agent setup where the market price information is modeled from agent interactions. We will explore this idea of using *bid efficiency* again in section 7.3.3.

Clicks	CTR	Cost	CPM	CPC	Impr.
170	0.00212	6210.461	77.736	36.532	79891

Table 7: PRUD algorithm results on validation set.

6.4 Ensemble

Having trained a number of linear and non-linear bidding strategies in the previous sections, a natural extension is to build an ensemble of these models. Basically, the ensemble model prediction is a weighted-average of the sub-models' bid prices. The weights are optimized using the CE method (4.1.1) by sampling each model's weight from a different beta distribution (initialized as uniform). Note that we do not constrain the weights to sum to 1 as we wanted additional flexibility in ensembling. Furthermore, the budget constraint is sufficient in preventing the weights from blowing up during optimization.

We found that optimizing over fewer individually strong models produced the most robust results. In the end, the four models included in the ensemble are linear bidding (logistic and GBT CTR), closed-form and CDF predictions with GBTs CTR. Results on validation set surpassed individual models (table 8).

Clicks	CTR	Cost	CPM	CPC	Impr.
174	0.00180	6260.933	64.644	35.982	96852

Table 8: Ensemble model results on validation set.

7. MULTI-AGENT BIDDING STRATEGY

We now explore the multi-agent setting where agents compete against each other and hence the best strategy has to take into account opposing strategies.

7.1 Framework

We decide to use a real-time bidding setting where bidding takes the form of a repeated game allowing a more game theoretical approach. The impressions are treated in a sequential manner where they are bid on one after, permitting feedback to the agents. More specifically, the feedback informs the agents whether they won the auction or not, and in the case that they did win, the pay price and if a click happened.

We design a framework to allow this real time bidding setting. We use a list of different algorithms and have each one output a bid for each impression. Then, we compute the winning bid and pay price and send the feedback to the different algorithms.

7.2 Analysis of multi-agent setting

In this setting, each algorithm has more information about its own state since it knows if it wins a bid immediately after bidding. This allows the algorithm to know about the number of clicks it has achieved as well as other metrics and most importantly, how much budget it has left.

Furthermore, the algorithm has information about the behavior of its opponents through the price it has to pay when it wins an impression, i.e. the highest bid of competitors. It can also know that the winning price of the auctions it loses is higher than its bid price. This allows the agent to evaluate the market price in real time.

Furthermore, by using these information, the algorithm can possibly take adversarial actions against the opponents' strategies.

7.3 BidStream

We design a game theory based approach taking advantage of the points mentioned above.

7.3.1 Estimation of the winning price

The first module of this algorithm is an estimator of the winning price of each impression. We assume that the winning price is a function of the CTR, since in general all strategies we have seen and developed so far take it into account. From there, we estimate the current highest *base_bid* in the competition in an online fashion. Put in other words, we estimate the current market value of a click.

To perform this estimation, we follow an approach inspired by the BIRCH [18] and CluStream [1] algorithms. But in order to adapt more easily, we use a moving average instead of a true cluster. We keep track of the current average highest *base_bid* and of its squared value, allowing us to have an expected *base_bid* of the current market and the variance of this estimation.

Since the algorithm is only informed of the pay price when it wins the auctions, we perform updates as shown in algorithm 3. Note that if it loses intentionally, we do not update our estimate and if it loses when it thought that it would win, we consider the bid as a lower bound on the true pay price.

Input:

1. bool: *won_bid*
2. float: *pay_price*

if *won_bid* **then**
  $base\_bid = pay\_price / last\_pCTR$
  $\mu_{base\_bid} = \mu_{base\_bid} \times recall + (1 - recall) \times base\_bid$
  $\mu_{bid^2} = \mu_{bid^2} \times recall + (1 - recall) \times (base\_bid)^2$
else
  if *last_predicted_opp_bid* $<$ *last_bid* **then**
    $base\_bid = last\_bid / last\_pCTR$
    $\mu_{base\_bid} = \mu_{base\_bid} \times recall + (1 - recall) \times base\_bid$
    $\mu_{bid^2} = \mu_{bid^2} \times recall + (1 - recall) \times (base\_bid)^2$
  end
end
$\sigma^2_{base\_bid} = \mu_{bid^2} - \mu^2_{base\_bid}$

Algorithm 3: Online market price estimation.

We empirically observe that this algorithm finds very good estimates of the *base_bid* of agents with linear strategies when competing against them.

7.3.2 Budget planning

The second point of the feedback we want to take advantage of is the budget the algorithm as left. With this information, it can try to allocate its budget approximately evenly over the data set. The motivation behind this is that the algorithm cannot know in advance where the best opportunities of bets are. Hence, it makes sense not to spend everything on the first few impressions trying to out-bid other agents when the opportunities might be better on later impressions. In a similar way, the agent cannot assume that later impressions are going to be better.

To balance these two objectives, we compute the average budget per impression and a *budget_factor*:

$$budget\_factor = \frac{current\_budget}{impressions\_left} / average\_budget.$$

We use this factor to encourage higher bids when the budget was under-spent and reduce bids when it has over-spent.

7.3.3 Measuring efficiency

Inspired by the PRUD approach [8], we propose to enhance the BidStream algorithm by including a *bid efficiency* measurement. We argue that this allows the algorithm to counter non-linear strategies.

In a very similar way to how we estimate the *base_bid* of our opponents' strategies, we propose to estimate its *bid efficiency*. We then compute a *bid_eff_factor* by dividing the bid efficiency on the next bid with the average bid efficiency of the opponent. This factor encourages higher bids on impression that would have a higher bid efficiency.

We note that it can be mathematically shown that the $bid\_eff\_factor = 1$ if the opponent uses a linear strategy.

7.3.4 Final bid function

The aim of the BidStream algorithm is to adapt to the bid prices of the opponent algorithms, hence it uses the following function to determine the bid:

$$bid = \mu_{base\_bid} \times \frac{pCTR}{avgCTR} \times budget\_factor \times bid\_eff\_factor.$$

7.3.5 Discussion

We observe that this function allows the algorithm to perform well and adapt to the opponent's strategy. Before comparing it to the other strategies in a competitive setting, we measure its performance on the validation set without any active opponents. Note that it performs almost as good as the linear bidding without having to tune a *base_bid* before hand but by just discovering it online.

Clicks	CTR	Cost	CPM	CPC	Impr.
168	0.001528	6261.227	56.981	37.2692	109881

Table 9: BidStream alone on the validation set.

In a competitive setting, this algorithm has two additional advantages. One, it bids around the bid price of its opponents which forces them to pay a higher price when they win the auction, causing them to run out of budget quicker. Two, the use of the $budget\_factor$ pushes the algorithm to perform the equivalent of exploration actions in reinforcement learning if it did not win a bid in a long time. This allows the market price estimation algorithm to be more accurate as well.

However, we identify two drawbacks to this algorithm due to the two assumptions that we make. We first assumed that the good bid opportunities are evenly spread through the data set. This is not necessarily true, but sadly when processing the data set in a sequential way, it seems hard to predict where the best opportunities lie. Hence we do not think we can avoid this assumption.

The second assumption is that in general, the bid price of the opponents can be approximated by a linear function of $pCTR$. This appears to be a reasonable assumption given that the algorithm performs well against the original pay price of the validation set and against the closed-form solution agent. Moreover, we think that the *bid_eff_factor* improves upon this against non-linear agents.

8. EVALUATION AND DISCUSSION

We now evaluate all the agents that we have developed in a competition against one another. The results are shown in the leader-board in table 10 and the evolution of the number of clicks and budget of each algorithm are shown in figures 5 and 6. The algorithms that use the CTR prediction were ran with two versions as identified by their number: ones using the `XGBoost` predictor (*Linear1*, *Close1*)and ones using the logistic predictor (*Linear2*, *Close2*). We also show a recap of the performance of the algorithms on winning criteria #1 in table 11.

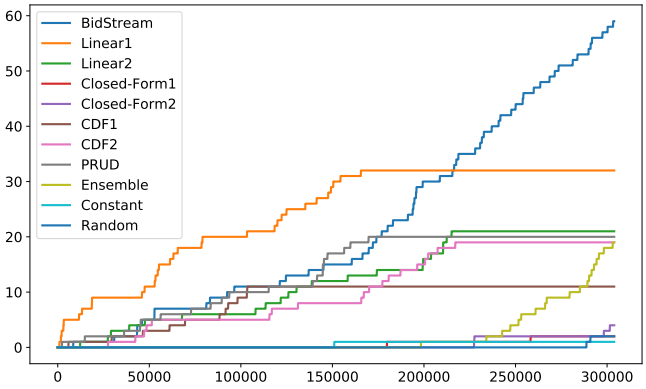


Figure 5: Evolution of the number of clicks of each algorithm of the run.

We observe that in this competitive setting, the BidStream algorithm largely outperforms the others that were not designed for the competitive setting. We see that it manages to spread its budget over the entire data set, whereas the linear bidding strategy is preferment in the beginning but runs out of budget in the middle of the run.

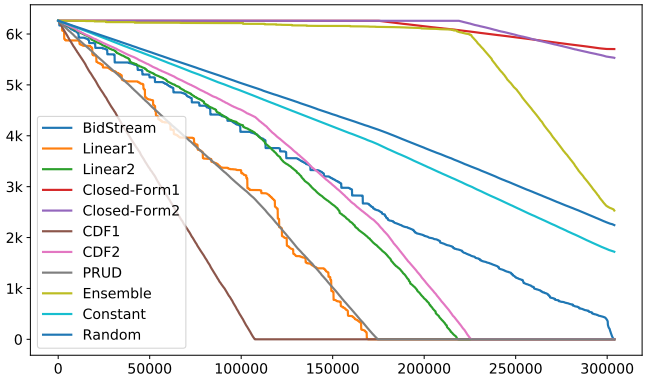


Figure 6: Evolution of the budget amounts of each algorithm of the run.

Comparing the performance on the two winning criteria, we observe that the closed-form solutions and the ensemble method perform very well on criteria #1 but have difficulties in the competitive setting. We believe that this is because their equivalent of *base_bid* is lower than one of the other algorithms and since they cannot adapt to the competition, they lose many auctions. The behaviour of the ensemble in the last 50k impressions appears to confirm this observation. It performed very well and achieves a high number of clicks in a few iterations as the other agents run out of budget and the ensemble bids the highest on high CTR impressions.

Algo	Clicks	CTR	Cost	CPM	CPC	Impr.
BidStream	59	0.0125	6261.1	1331.0	106.1	4704
Linear1	32	0.0227	6261.2	4453.2	195.6	1406
Linear2	21	0.00144	6261.3	429.5	298.1	14578
PRUD	20	0.00046	6261.0	144.33	313.0	43380
Ensemble	19	0.00128	3731.0	253.1	196.3	14738
CDF1	19	0.00096	6261.2	318.2	329.5	19675
CDF2	11	0.00072	6261.3	413.9	569.2	15125
Close2	4	0.00064	729.4	116.8	182.3	6242
Close1	2	0.00030	557.2	84.9	278.6	6559
Random	2	3.85e-05	4017.1	77.3	2008.5	51922
Constant	1	1.57e-05	4540.6	71.7	4540.6	63322

Table 10: Leader-board of the different algorithms designed in this paper when competing against each other.

Algo	Clicks	CTR	Cost	CPM	CPC	Impr.
Ensemble	174	0.00180	6260.9	64.6	36.0	96852
CDF1	171	0.00222	5955.3	77.4	34.8	76912
Linear1	170	0.00143	6252.3	52.8	36.8	118478
PRUD	170	0.00212	6210.4	77.7	36.5	79891
BidStream	168	0.001528	6261.2	57.0	37.3	109881
Close1	167	0.00138	5752.15	47.9	34.4	120167
Linear2	166	0.00148	6091.8	54.4	36.7	112016
CDF2	157	0.00144	5888.9	54.2	37.5	108661
Random	66.65	0.000463	6261.2	43.6	94.1	143538
Constant	66	0.000472	6261.2	44.8	94.9	139735

Table 11: Leader-board of the different algorithms designed in this paper using winning criteria #1.

9. CONCLUSION

Computational advertising has evolved dramatically over the years with the introduction of RTB and machine learning algorithms. Using the iPinYou data set, we first explored strategies in a simple environment where pay prices are independent of our bids. For linear bidding strategies, we found that using non-linear CTR estimators when properly calibrated, performed better than linear ones as they had higher AUCs. Next, we considered more general non-linear strategies that modelled the bidding landscape such as winning price and CTR uncertainties. These more complex strategies engendered some improvements in secondary metrics such as CPM and CPC. Through a model ensemble, we could obtain more clicks than any individual strategy did. Finally, in the multi-agent setting, we considered the realistic case of real-time bidding. We designed a game theory based algorithm – BidStream – to specifically exploit the feedback it receives by. Even with online estimations of parameters, we demonstrate in the previous single-agent setting that BidStream could still adequately estimate market price and perform comparably with linear bidding strategies. In the simulated multi-agent competition, BidStream indubitably proved to be the best by achieving 84% more clicks than the next best bdding strategy.

10. REFERENCES

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 81–92. VLDB Endowment, 2003.

[2] Google. The arrival of real-time bidding and what it means for media buyers. In *Google White Paper*, 2012.

[3] X. Guo, A. Hu, R. Xu, and J. Zhang. Learning mean-field games. *arXiv preprint arXiv:1901.09585*, 2019.

[4] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–9. ACM, 2014.

[5] J. Jin, C. Song, H. Li, K. Gai, J. Wang, and W. Zhang. Real-time bidding with multi-agent reinforcement learning in display advertising. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2193–2201. ACM, 2018.

[6] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 43–50. ACM, 2016.

[7] K.-c. Lee, B. Orten, A. Dasdan, and W. Li. Estimating conversion rate in display advertising from past erformance data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 768–776. ACM, 2012.

[8] C.-C. Lin, K.-T. Chuang, W. C.-H. Wu, and M.-S. Chen. Combining powers of two predictors in optimizing real-time bidding strategy under constrained budget. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2143–2148. ACM, 2016.

[9] S. Mannor, R. Y. Rubinstein, and Y. Gat. The cross entropy method for fast policy search. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 512–519, 2003.

[10] A. Niculescu-Mizil and R. Caruana. Obtaining calibrated probabilities from boosting.

[11] T. Parr, K. Turgutlu, C. Csiszar, and J. Howard. Beware default random forest importances, 2018.

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

[13] 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):645–659, 2018.

[14] I. Trofimov, A. Kornetova, and V. Topinskiy. Using boosted trees for click-through rate prediction for sponsored search. In *Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy*, page 2. ACM, 2012.

[15] W. C.-H. Wu, M.-Y. Yeh, and M.-S. Chen. Predicting winning price in real time bidding with censored data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1305–1314. ACM, 2015.

[16] J. Xu, X. Shao, J. Ma, K.-c. Lee, H. Qi, and Q. Lu. Lift-based bidding in ad selection. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[17] B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. Citeseer.

[18] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, pages 103–114. ACM, 1996.

[19] 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*, pages 1077–1086. ACM, 2014.

[20] W. Zhang, S. Yuan, J. Wang, and X. Shen. Real-time bidding benchmarking with ipinyou dataset. *arXiv preprint arXiv:1407.7073*, 2014.