Clicks-Maximizing Bidding Algorithm for a Programmatic Advertising Company

Martin Wiegand

April 29, 2022

1 Introduction

In this document, I outline the idea and implementation of a clicks-maximizing ad-bidding procedure for a UK-based programmatic advertising company. The code, together with some statistics and figures summarizing the resulting optimal bids, can be found in the file `clicks_maximizer.ipynb`.

The implementation is based on a dataset of a completed digital marketing campaign containing all the auctions the advertising company won. The dataset contains 3.46 million observations, each of which represents the set of auctions won by the ad-buying platform within one hour for users with certain characteristics. This amounts to a total number of 238.7 million impressions shown during the campaign. The dataset furthermore contains the average bid for each set of auctions, device type, line item, location, hour of day, and day of week. Notably, it does not contain dates, which means the problem could not be framed as a *dynamic* optimization problem.

The names of the advertising company, the advertiser, and the product, as well as the place where the campaign was run are kept anonymous.

2 Setting up a clicks-maximization problem

For this problem, we want to maximize the total number of clicks within a specified time frame, given a fixed budget. This is reasonable as long as each click is equally valuable to the advertiser.[1]

Let's say we have N observations, where each observation contains the number of bought impressions, clicks, and bidding price, for a given set of characteristics (e.g., device type, line item id, location, hour, day). We denote the bids given for each observation as $b = (b_1, \dots, b_N)$, and the corresponding characteristics $X = (X_1, \dots, X_N)$. We can assume that expected impressions m_i for any observation i in our dataset are a function of bids b_i and features X_i,

[1] We can, of course, think of reasons this may not be the case—e.g. if a click is more likely to result in a conversion for a retargeted customer than a first-targeted one.

$$m_i = f\left(b_i, X_i\right).$$

Impressions are increasing in bids,

$$\frac{\partial f}{\partial b_i} > 0,$$

but with decreasing speed,

$$\frac{\partial^2 f}{\partial b_i^2} < 0.$$

Next, we have expected clicks k_i, a function of impressions m_i and features X_i

$$k_i = g\left(m_i, X_i\right).$$

Clicks are increasing in impressions,

$$\frac{\partial g}{\partial m_i} > 0,$$

and we have no reason to believe this would happen at increasing speed, so

$$\frac{\partial^2 g}{\partial b_i^2} \leq 0.$$

Lastly, we can also consider the indirect clicks function that depends on bids instead of impressions,

$$k_i = h\left(b_i, X_i\right) = g\left(f\left(b_i, X_i\right), X_i\right).$$

We have

$$\frac{\partial f}{\partial b_i} > \frac{\partial h}{\partial b_i} > 0,$$

and

$$\frac{\partial^2 f}{\partial b_i^2} < \frac{\partial^2 h}{\partial b_i^2} < 0.$$

Total clicks, Π, is just the sum over all observations,

$$\Pi\left(b, X\right) = \sum_{i=1}^{N} h\left(b_i, X_i\right).$$

Next, we have a cost function C and a budget B, leading to the budget constraint

$$C(b, X) = \sum_{i=1}^{N} b_i f(b_i, X_i) \leq B.$$

Last of all, we cannot have negative bids, so a further constraint is

$$b_i \geq 0 \ \ \forall i.$$

The maximization problem now becomes

$$\max \Pi(b, X) \quad \text{s.t.} \ \ C(b, X) \leq B, \ \ b \geq \mathbf{0}.$$

3 Interpretation

The above problem can be solved within the Karush-Kuhn-Tucker framework (see the appendix for details). This leads to one particular insight I would like to highlight here. For two observations i and j—let's say they represent auctions in different locations—where we find an inner solution (i.e., $b_i > 0$ and $b_j > 0$), we have

$$\frac{\partial \Pi(b, X)}{\partial b_i} \Big/ \frac{\partial \Pi(b, X)}{\partial b_j} = \frac{\partial C(b, X)}{\partial b_i} \Big/ \frac{\partial C(b, X)}{\partial b_j}.$$

The left-hand side is the *marginal rate of substitution*, which says: if I reduce my bid for location j by one cent, by how many cents do I need to increase my bid in location i to have the same number of clicks in the end? The right-hand side is the price ratio, where "price" refers to the cost raised by marginally increasing the bid in the respective location. So what this equation is telling us is that the marginal click per unit of money spent is proportional to the marginal cost for each location.

Let's take an example where the above equation does not hold. Assume there are only two locations, 1 and 2, and as of now we bid amounts b_1 and b_2 per impression. For these bids, we find that in location 1, a 1 cent increase in the bids leads to 1 more dollar spent, and in location 2, a 1 cent increase in the bids leads to 2 more dollars spent. At the same time, we see that a 1-cent increase in bidding for location 1 leads to an expected number of 0.1 more clicks, whereas it leads to an expected number of 0.3 more clicks in location 2. Then we can improve our situation by reducing our bid in location 1 by 2 cents and increasing the bid in location 2 by one cent. We have to pay 2 dollars less on location 1 but 2 dollars more on location 2. We also lose 0.2 clicks in location 1 but gain 0.3 clicks in location 2. Clearly, we should keep increasing our bid like this until the above equation holds.

4 Estimating the functions f and g

We approximate f and g from the data. Our approach is to find clusters of covariates for which the relationships between bids, impressions, and clicks can be assumed to be homogeneous, while still having sufficient variation to identify the function. The main reason for taking this approach is to reduce the size of the optimization problem: instead of maximizing over a 3 million-dimensional bid vector, it is enough to maximize over a less than 5,000-dimensional bid vector, which represents clusters of observations that can be treated the same way.

For the function f, the characteristics whose unique combinations constitute clusters are the following: line item ID, device type, day of the week, and hour of the day. This choice is informed by the observed variation of bids, total revenue, and the number of impressions over these variables. For the estimation, we use a spline estimator with monotonicity and concavity constraints, using the `PyGAM` package (see here).

For the function g, the characteristics whose unique combinations constitute clusters are the following: retargeted, device type, weekend, and indicators for various time periods within a day (early morning: 7:00-9:00, late morning: 9:00-11:00, noon: 11:00-13:00, afternoon: 13:00-18:00, evening: 18:00-23:00, early night: 23:00-3:00, late night: 3:00-7:00). This choice is informed by the observed variation of the number of impressions and click-through rate over these variables. Note that by not clustering by every hour of every day of the week, and not by each line item ID, we create fewer and larger clusters than for f. This is due to the fact that clicks are harder to predict, and that relatively more observations are needed to obtain robust estimates of click-through rate. For the functional form of g, we assume a linear relationship. The expected number of clicks is then simply the sample click-through rate times number of impressions.

Below are two example clusters with their respective functions. The upper figure shows the relationship between bids and impressions. The function is forced to be monotonic and concave, but when estimating unconstrained spline approximations instead of constrained ones, the result tends to look almost the same. The lower figure shows the relationship between impressions and clicks. Even though a first glance at the point cloud does not suggest a linear relationship, the unconstrained spline approximation (in blue) is almost perfectly coinciding with the regression line (in orange) until shortly before 3000 impressions. It is noteworthy that the large variance in the relationships between bids, impressions, and clicks is most likely inherent to the problem, given the hard-to-predict behavior of market actors and users. At the same time, it seems quite likely that conditioning on more covariates or using different approaches to obtain f and g, or their respective derivatives, may lead to more precise conditional expectations.

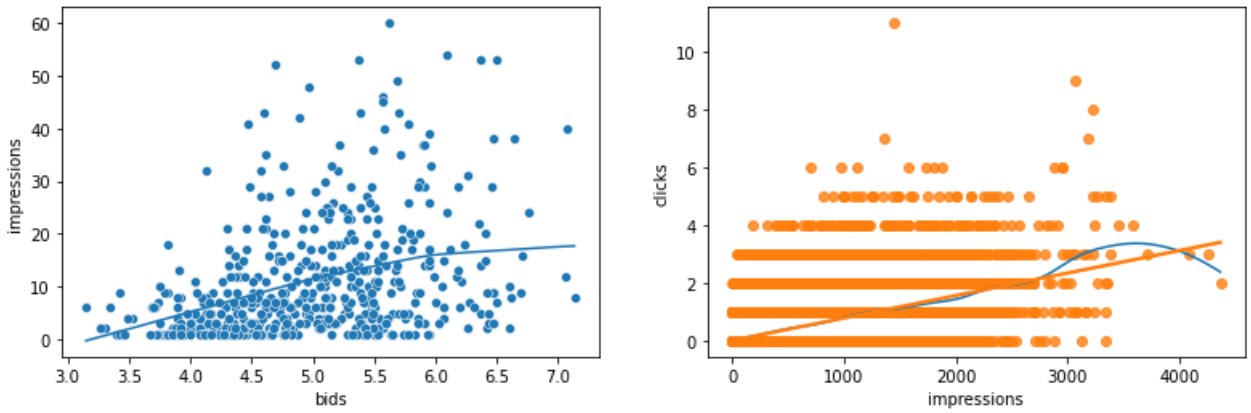

Figure 1: Example clusters with function estimates of f (upper) and g (lower)

5 Optimization

Having estimated the functions f and g, we go on to the optimization problem. We use Python's `scipy.optimize` package (we found this link helpful to get started).

5.1 Sample Splitting

The optimization is over a dataset with one row per unique combination of features, amounting to 4,636 rows. Optimization over a 4,636-dimensional vector is still computationally very costly. We therefore split the sample into k random splits of roughly equal size and optimize over these splits, assuming a respective budget equal to the total bids made within them. This means that we do not "truly" optimize over the whole set of clusters. It could be that, by accident, a high-potential cluster that is currently under-budgeted ends up in a split with other low budget clusters, resulting in an underestimate of the true optimal bid for that cluster. However, for a large enough split size, this is relatively unlikely to occur, as such differences in budget should equal out. We choose a value of $k = 40$, but also consider $k = 30$, which takes significantly longer but yields an almost identical optimal bid vector.

We also look at the shadow prices, i.e. the Lagrange multipliers of the budget constraints, resulting from each optimization. This term can be interpreted as the marginal increase of clicks resulting from a marginal increase in the budget. These terms are fairly similar over the various folds, ranging from about 0.07 to about 0.1, with a few outliers going up to 0.16. The fact that they are close to each other suggests that the procedure comes fairly close to the overall optimum, while still possibly leaving some room for improvement. This could perhaps be achieved by a sample splitting procedure that attempts a roughly

equal distribution of cluster-wise impressions and spending per split. There was no time left to try this out.

5.2 Additional constraints

Instead of the mere condition that bids have to be non-negative, we require them to be within the bounds of the bids we observe for them in the respective clusters. The reason is that we were cautious to extrapolate outside of the intervals for which we have estimated the functions f.

In addition to these bounds, we constrain the range of possible bids to values for which the estimated function f has non-negative values. This is necessary because for a very few instances, due to the functional form we chose for f, the lowest bids are predicted to yield negative clicks.

Appendix

Lagrange optimization

The Lagrangian is

$$\mathcal{L}(b, X, \lambda) = \Pi(b, X) + \lambda_0 \cdot (B - C(b, X)) + \sum_{i=1}^{N} \lambda_i \cdot \left(a_i^2 - b_i\right).$$

$\lambda_0, \ldots, \lambda_N$ are the Lagrange multipliers, and $a_1, \ldots, a_N$ are slack variables. The Kuhn-Tucker conditions are:

$$\nabla \mathcal{L}(b, X, \lambda) = \mathbf{0},$$

or, for all i:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial b_i} &= \frac{\partial \Pi}{\partial b_i} - \lambda \frac{\partial C}{\partial b_i} \\
&= \frac{\partial g(b_i, X_i)}{\partial b_i} - \lambda_0 \left(f(b_i, X_i) + b_i \frac{\partial f(b_i, X_i)}{\partial b_i} \right) - \lambda_i \\
&= 0,
\end{aligned}$$

for $i = 1, \ldots, I$:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \lambda_i} &= a_i^2 - b_i \\
&= 0,
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial a_i} &= 2\lambda_i a_i \\
&= 0,
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \lambda_0} &= B - C(b, X) \\
&= 0.
\end{aligned}$$