Building a Bidding agent: Real Time Bidding with RL

Paul Pereira

Previous Work

- Two step approach to building a real-time bidder:
  - Learn to predict the probability that a user will click on the ad if we win the bid
  - Use this probability to determine when and how much to bid
- Much of the work is focused on the first step:
  - Possible to get high accuracy using simple linear classifiers
  - In practice, use the Lin method to decide when to bid and how much.
  - $a(x) = b_0 * \frac{CTR(x)}{AVG\_CTR}$

Issues and RL solution

- The main problems of the Lin bidder is that it doesn’t take into account the entire situation.
  - The bidder has a budget B to manage.
  - The number of auctions T is limited.
  - Truthful bidding doesn’t make sense if we have multiple auctions.

- [1] showed that it was possible to set up a dynamic programming solution that performed better in A|B testing compared to Lin.

$$V(t,b) \approx \max_{0 \le a \le b} \Big\{ \sum_{\delta=0}^{a} m(\delta)\theta_{\mathrm{avg}} + \sum_{\delta=0}^{a} m(\delta) V(t-1, b-\delta) + \sum_{\delta=a+1}^{\infty} m(\delta) V(t-1, b) \Big\}. \tag{8}$$

Algorithm 1 Reinforcement Learning to Bid

```
Input: p.d.f. of market price $m(\delta)$, average CTR $\theta_{avg}$, episode
    length $T$, budget $B$
Output: value function $V(t,b)$
 1: initialize $V(0,b) = 0$
 2: for $t = 1, 2, \cdots, T-1$ do
 3:     for $b = 0, 1, \cdots, B$ do
 4:         enumerate $a_{t,b}$ from 0 to $\min(\delta_{max}, b)$ and set $V(t,b)$ via
            Eq. (8)
 5:     end for
 6: end for
Input: CTR estimator $\theta(\boldsymbol{x})$, value function $V(t,b)$, current state
    $(t_c, b_c, \boldsymbol{x}_c)$
Output: optimal bid price $a_c$ in current state
 1: calculate the pCTR for the current bid request: $\theta_c = \theta(\boldsymbol{x}_c)$
 2: for $\delta = 0, 1, \cdots, \min(\delta_{max}, b_c)$ do
 3:     if $\theta_c + V(t_c - 1, b_c - \delta) - V(t_c - 1, b_c) \ge 0$ then
 4:         $a_c \leftarrow \delta$
 5:     end if
 6: end for
```

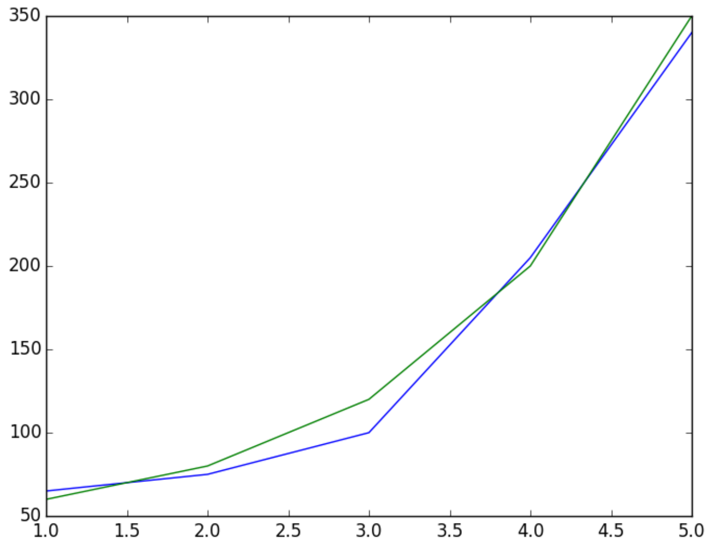
Q-Learning

- Motivation : Only requires us to know how to evaluate the CTR, no need to estimate the bidding landscape and make the approximation : $m(b, x) = m(b)$
- We can assume that if we know the CTR of every state, we can use the CTR as part of the state instead of the feature vector. In this case, the state is a tuple (t,b,CTR(x))
- This might not be enough to reduce the state space. We can choose to divide the series of auctions into multiple episodes.
- In order to speed up the convergence, we can choose to fill in values for states that are close to one another:
  - Run through an episode and keep track of which states were visited.
  - For every state in the neighborhood that still has a value of 0, assign the same value as the visited state.

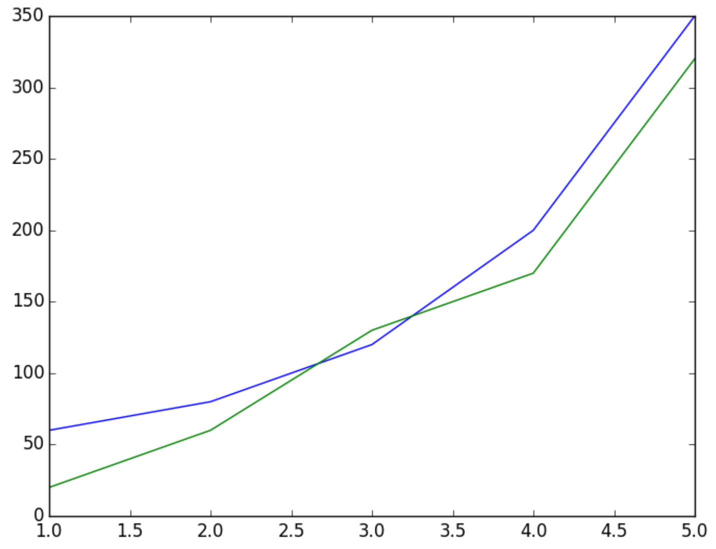
Results

- Methodology : Used 10,000 episodes of 100 auctions for training, scored using 50 episodes of 100 auctions.
- Feature vector made of 3 binary features. One bidder uses Q-learning to pick actions, the other two use LIN. Reserve price of 1.

Using CTR vs feature vector

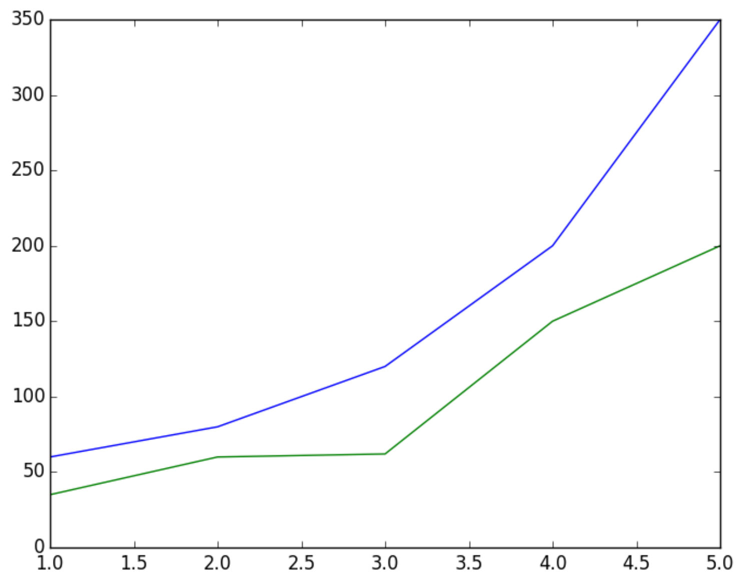


Using LIN vs Q-Learning



Friend-Q

- Second price auctions are vulnerable to collusion.
- Our previous model assumes that the other bidders strategy are fixed.
- Friend-Q allows us to move into a multi-agent setting and take advantage of the possible coordination equilibria.
- I expected to obtain better results in a situation where the budget ratio is small.



References

- [1] Real Time Bidding by Reinforcement Learning in Display Advertising, Weinan Zhang, Jun Wang
- [2] Friend-or-Foe Q-Learning in General Sum Games, Littman