CS229: Apply Reinforcement Learning on Ads Pacing Optimization

Ying Chen (SCPD: ychen107)

Abstract

Online display advertising is a marketing paradigm utilizing the internet to show advertisements to the targeted audience and drive user engagement. Billions of display ad impressions are purchased on a daily basis through public auctions hosted by real-time bidding (RTB) exchanges. While the digital Ads business plays an increasingly important role in the market, how to spend the budget from the advertiser in an effective way becomes one of the essential challenges. Pacing is a strategy to ensure that the budget spends evenly over the schedule of advertiser's ad set. Here, we present two reinforcement learning approaches, DQN and DDPG to smooth the daily budget spending.

1 Introduction

Since 2009, Real-time bidding(RTB) has become popular in online display advertising [1]. RTB allows the advertiser to use computer algorithms to bid in real-time for each individual ads placement to show ads. With its fine-grained user targeting and auction mechanism, RTB has significantly improved the ad's return-on-investment (ROI).

Advertisers usually manage their advertising campaigns through Demand Side Platforms (DSP). The interaction between users, DSPs, and the ad exchange is summarized as follows:

(1) When a user visits an ad-supported site, each ad placement triggers an ad request to the ad exchange.

(2) The ad exchange sends the bid requests for this ad opportunity to several DSPs, along with other available information such as the user cookie.

(3) Given the bid request and attached information, a DSP finds the best matching ad and calculates a bid price. It sends the price back to ad exchange to join the auction.

(4) After receiving the bid responses from DSPs, the ad exchange hosts an auction and picks the ad with the highest bid as the winner. Then ad exchange notifies the winner DSP, and the price will be charged on the winning advertiser.

(5) Finally, the winner's ad content will be shown to the specific user. It considers as one impression counted on the ad. And the user's feedback (e.g., click and conversion) would be tracked by the DSP.

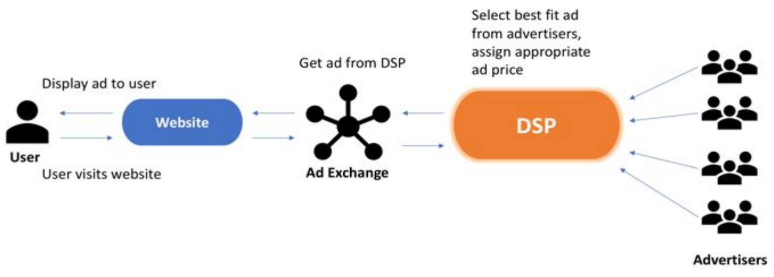

Figure 1. Real-time Bidding in Advertising

This whole procedure above takes within milliseconds. That's why we call it Real-time bidding (RTB).

In RTB, an advertiser prefers the ads to be viewed by users evenly throughout a day to increase the chances of reaching the possible potential customers, but budgeting alone does not account for this concept. Therefore, we need the pacing in ads, which is very similar in concept to pacing in running. Say an advertise set up a budget for one day, and let the DSP run the ad. It happens to start spending at a time when opportunities are more expensive due to increased auction competition. If DSP didn't use pacing, it could spend the entire budget in a few hours on expensive opportunities. Instead, DSP "pace" spending so that the ad has the budget available later at the end of a day when there are likely to be lower-cost opportunities available. Most of the IT companies employ probabilistic filtering to control pacing, which is called pacing rate. Pacing rate is a bidding probability from 0 to 1. If an ad's pacing rate is set to 0.34, it means the probability for this ad to join the next auction is 34%. We consider a pacing strategy is good if it satisfies: (1) It can spend most of the daily budget but not overspending. (2) It has smooth delivery throughout the day. (3) The pacing rate does not oscillate too much.

2 Related Work

Reinforcement learning as a framework for sequential decision making has attracted the attention of researchers since many years ago [9]. In recent years after deep neural networks were introduced to solve reinforcement learning problems, a series of new algorithms were proposed, and progress was made on different applications [10,11,12,13]. Some researchers reported success stories applying deep reinforcement learning to online advertising problem, but they focus on bidding optimization [4,5,14] not pacing.

Despite its importance in ads-serving systems, budget pacing for ads campaigns is relatively less discussed in the literature. In [6] the authors described budget pacing problem in online advertising and then addressed it by proposing an algorithm that dynamically changes the bid price. A later work [7] used throttle rate instead of bid adjustment to control the speed at which a campaign delivers. In a more recent work [8] the authors tried to solve both smooth delivery and campaign performance optimization problems with a multiplicative feedback pacing algorithm. To the best of our knowledge, there was no reinforcement learning based approach for budget pacing in existing literature.

3 Dataset and Features

Since the company data is confidential and there is no public dataset available for ads auction simulation (the link of our previous choice iPinyou dataset [3] has expired), we generated data through simulation.

3.1 Simulator

We implemented a simulator that simulates the ads auction environment. To be more similar to the real world, we use time-of-day pattern to simulate the traffic. With traffic ups and downs, it makes the environment varies in different time steps. It also brings difficulty in algorithm training, which is good. The simulator is implemented approximately following the convention in Gym package [2], with a *step()* function and *reset()* function interface that can be called by the pacing algorithm.

The function *step()* is called with pacing signal as input. The pacing algorithm plays the role of an agent, at each time step it issues a pacing signal to the simulator, which runs the auction with provided pacing signal, computes and returns the results.

When initialization and reset, we generate simulated data for the length of one day. We partition the time of one day into time slots of 1 minute each, and for each time slot, we generate a random amount of impressions according to some predefined traffic density. Each impression has a cost associated with it, a certain percentage of these impressions will have clicks, these are also generated by sampling a pre-defined distribution.

4 Methods

4.1 Problem Formulation

It is well known that optimal bidding can be formulated as an MDP problem [4]. If we assume the bid for a campaign is fixed, then the bidding behavior relies totally on the pacing rate. We largely follow the setup in these works and describe the model structure for campaign pacing problem as follows:

State space The state of the campaign is characterized by its remaining budget and remaining delivery time. To simplify the discussion, we only consider budget delivery for a single day here. We divide the time of one day into finite time steps of length dt. At each time step, we define state S_t = (b_t, t), where $b_t = B_t/B_{daily}$ is the budget consumption ratio at time t, B_t is the remaining budget at time t and B_{daily} is the daily budget. Note that b_t is not an absolute value but normalized to $[0, 1]$ range.

Action space The action signal is the pacing rate $p_t \in [0, 1]$, which represents the probability of joining an auction.

State transition Let's say at time t a campaign receives n_t bid request. It will ignore $(1-p_t)n_t$ of them and join the remaining auctions. Assuming it bids with a fixed price bid_t , the campaign will win all auctions with a cost less than bid_t. Therefore the budget spent at time t is decided by the pacing rate p_t along with the cost distribution in the auctions (for further details about online ads auction see [4,6,8]).
At each time t our pacing agent issues a p_t, the campaign adopts p_t and join auctions. Suppose it spent c_t dollars, the state can be updated as:

$$S_{t+1} = (b_t - c_t/B_{daily},\ t + dt)$$

We assume the traffic available to a campaign has a relatively stable distribution. Note that the state transition is affected also by other external factors

Cost function We define an ideal state trajectory $B_t^* = f(t)$ as our objective (e.g. if we want to spend the budget evenly throughout the day, then this curve will be a straight line). The cost at each time step t is

$$Cost_t = a(B_t - B_t^*)^2 + b(p_t - p_{t-1})^2$$

it's composed of two parts, the first term penalizes the deviation from idea state trajectory and the second term meant to encourage smoothness of action sequence.

4.2 Proposed Solution

Deep Q-learning Q-learning is a straightforward off-policy learning algorithm, it basically builds a Q-table which gives the reward function for (state, action) pairs, and update it while exploring the environment. The optimal policy, on the other hand, can be generated by taking greedy actions at each state according to Q-table. Deep Q-learning uses neural-network to approximate the Q-function. The algorithm of DQN with experience replay can be found in [1] so it's omitted here in the interest of space.

DDPG Deep deterministic policy gradient is a model-free off-policy actor-critic algorithm. It can be regarded as a combination of DPG [12] and DQN. The algorithm of DDPG with experience replay can be found in [13] so it's omitted here in the interest of space.

5. Experiment Results

Since we generate a random amount of impressions in each time slot to mimic the actual traffic, the environment input is entirely different every time. For pacing, it's hard to qualitatively compare the performance between two different algorithms.
Evaluation metric: We look at two quantitative criterions: a) the overspending or underspending should be small at the end of a day, b) the pacing signal should be as smooth as possible. We could alternatively use the cost function in Section 4.1 to compare RL based algorithm and baseline, which is defined to quantify the above heuristic.

Baseline

We use a multiplicative feedback control algorithm as our baseline. It's a closed-loop control system, similar to that of [7] with simplification. The idea is to adjust the pacing signal by multiplying it with the ratio of the desired speed of spending and the actual speed of spending in the last time interval, i.e.,

$$p_{t+1} = \frac{S_d}{S_a} p_t$$

where S_a is the actual spending speed, S_d is the desired speed. When p_{t+1} is too large we clip it to 1.0.
In our simulation, the baseline algorithm gives a reasonable result regarding budget delivery(c.f. Fig 2), however its pacing signal changes drastically throughout the simulation. It is because the environment is highly noisy so that the multiplicative factor can be vast and jumpy.
Adding further smoothing to the pacing signal gives another baseline, as shown in Fig 2. The smoothness of the pacing signal is improved but still jumpy towards the end of the day.

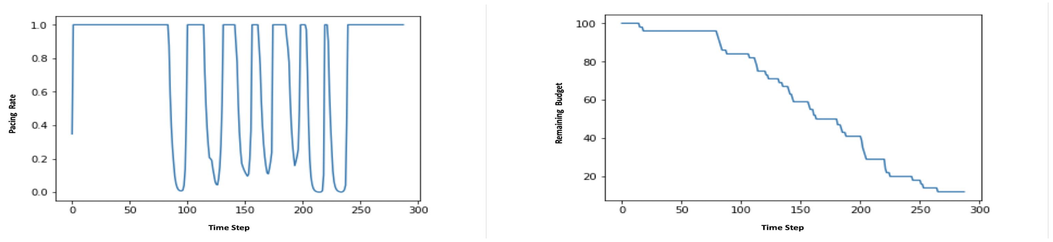

Figure 2. Baseline algorithm generates Pacing Rate and Remaining Budget

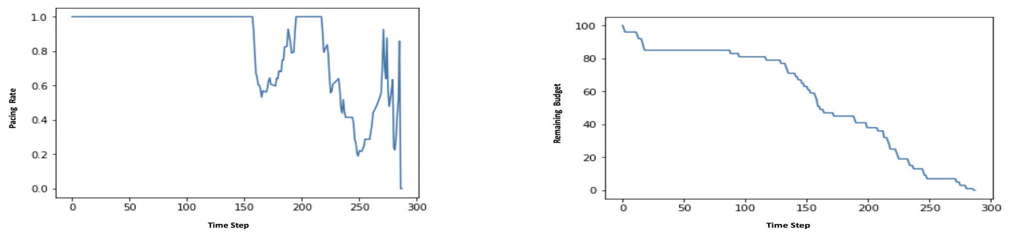

Figure 3. Smoothed Baseline algorithm generates Pacing Rate and Remaining Budget

DQN

The action space is discretized into a finite number of bins (we experimented with 10,12, 15 and 20 then found 10 performs best). As to state space, we tried both discretized and continuous and found their results are comparable, so we used continuous version for simplicity. The DQN is implemented with experience replay, for the discounted cost we set discount factor to 0.99, and we used TD(0) to compute the loss, for optimization we used Adam. In the beginning, we attempted to use a series of neural network layers and its result turned out very unstable. To avoid overfitting, we simplified the architecture of the neural network in code, using only one hidden layer with 16 neurons.
Results shown in Fig 4 and Fig 5 are generated from two trained DQN models. We observe that the pacing signal has been smoothened in Fig 5 but occasionally oscillates as Fig 4. It shows the training is not sufficient, and we could improve it by training with more data and more iterations.

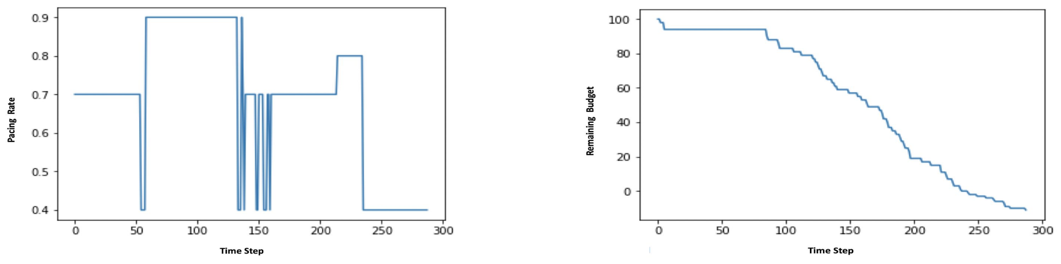

Figure 4. DQN algorithm generates Pacing Rate and Remaining Budget

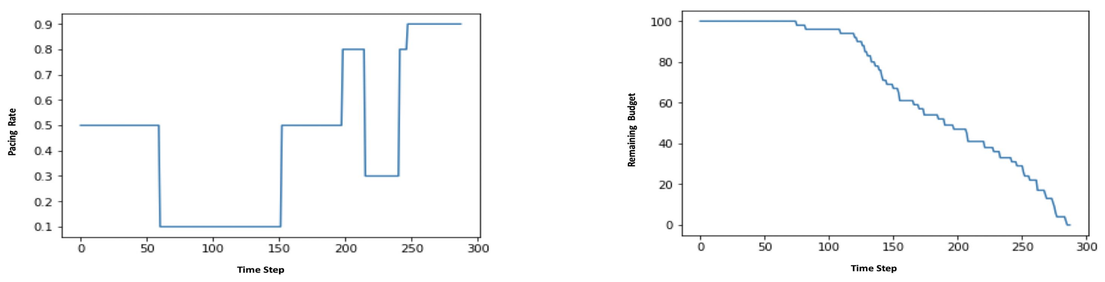

Figure 5. DQN algorithm generates Pacing Rate and Remaining Budget

Remark: For all these algorithms in the morning of a day the speed we spend the budget is usually not high. This is because when generating the simulated data, we considered time-of-day pattern and intentionally reduce the amount of traffic in morning hours.

DDPG

For DDPG the action space is continuous so that we won't see the stairs-like pacing signal curve as in DQN. However our algorithm still needs improvement because the training is not stable: sometimes the network can generate results like in Fig 6, sometimes it cannot learn anything, generating constant pacing rate for test cases. Even in Fig 6, the algorithm is not doing very well: the pacing rate at the end of day ramps up when it should go down, leading to overspending.

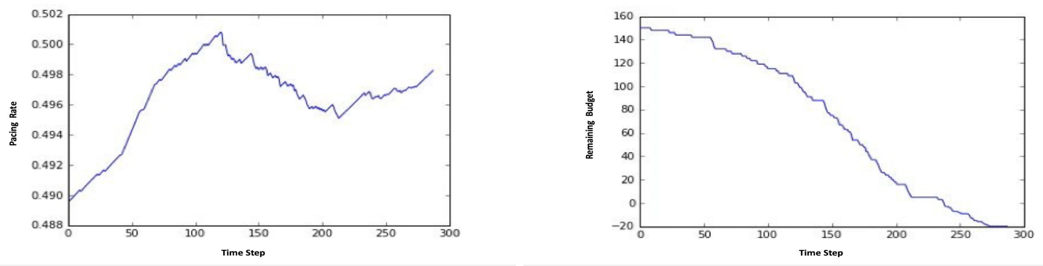

Figure 6. DDPG algorithm generates Pacing Rate and Remaining Budget

We suspect the instability in training is related to insufficient exploration of state-action space. Simply increasing the number of training iterations does not seem to be effective, we'll examine the O-U process and the way we inject noise into action to see if improvements can be made.

6 Conclusion & Future Work

Overall, we explored the possibility of applying reinforcement learning to ads budget pacing problem. On simulated ads auction data, our DQN algorithm shows promising results comparing with the baseline algorithm. It not only achieves the goal of budget delivery, but also maintains the pacing rate relatively smooth. We also implemented DDPG, but it needs further performance tuning to improve its stability.

In the future, we'll also consider the following directions for performance improvement: (1) Adding more features (e.g., features extracted from time series) (2) Explore other network structures. (3) Try alternative cost function and regularization. Regarding the project, we can also have the training on some company data if the data-security team grants permission. And intra-day budget spending could be another interesting topic for us to explore.

Code path: https://github.com/YingChen/cs229

Reference:

[1]. David S Evans. 2009. The online advertising industry: Economics, evolution, and privacy. Journal of Economic Perspectives 23, 3 (2009), 37–60.

[2]. Claudia Perlich, Brian Dalessandro, Rod Hook, Ori Stitelman, Troy Raeder, and Foster Provost. 2012. Bid optimizing and inventory scoring in targeted online advertising. In 18th SIGKDD. ACM, 804–812.

[3]. Weinan Zhang, Shuai Yuan, Jun Wang, Xuehua Shen, Real-Time Bidding Benchmarking with iPinYou Dataset. https://arxiv.org/abs/1407.7073

[4]. Han Cai, Kan Ren, Weinan Zhang, Kleanthis Malialis, Jun Wang, Yong Yu, and Defeng Guo. 2017. Real-Time Bidding by Reinforcement Learning in Display Advertising. In 10th WSDM. ACM, 661–670.

[5]. Yu Wang, Jiayi Liu, Yuxiang Liu, Jun Hao, Yang He, Jinghe Hu, Weipeng Yan, and Mantian Li. 2017. LADDER: A Human-Level Bidding Agent for Large-Scale Real-Time Online Auctions. arXiv preprint arXiv:1708.05565 (2017).

[6]. Kuang-Chih Lee, Ali Jalali, Ali Dasdan, Real Time Bid Optimization with Smooth Budget Delivery in Online Advertising. ADKDD 2013

[7]. Deepak Agarwal, Souvik Ghosh, Kai Wei, and Siyu You, Budget pacing for targeted online advertisements at LinkedIn. KDD 2014

[8]. Jian Xu, Kuang-chih Lee, Wentong Li, Hang Qi, Quan Lu, Smart Pacing for Effective Online Ad Campaign Optimization. KDD 2015

[9]. Leslie Pack Kaelbling, Michael L. Littman, Andrew W. Moore. Reinforcement Learning: A Survey. Journal of Arti cial Intelligence Research 4 (1996)

[10]. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. NIPS Deep Learning Workshop 2013

[11]. Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. ICML 2016

[12]. David Silver, Guy Lever, *et.al.* Deterministic Policy Gradient Algorithms. ICML 2014

[13].Lillicrap, et al. Continuous control with Deep Reinforcement Learning.
https://arxiv.org/abs/1509.02971

[14]. Junqi Jin, Chengru Song, Han Li, Kun Gai, *et. al.* Real-Time Bidding with Multi-Agent Reinforcement Learning in Display Advertising. CIKM 2018

[15]. Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, Wojciech Zaremba. OpenAI Gym https://arxiv.org/abs/1606.01540