A reinforcement learning method to select ad networks in waterfall strategy

Citation for published version (APA):

Document license:
CC BY

Document status and date:
Published: 01/01/2019

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 25. Apr. 2020
A Reinforcement Learning Method to Select Ad Networks in Waterfall Strategy *

Reza Refaei Afshar Yingqian Zhang, Murat Firat, and Uzay Kaymak
Eindhoven University of Technology, Eindhoven, Netherlands

1 Introduction

In [1], we develop a decision support system for ad publishers to select the most promising ad network for each impression. A high percentage of online advertising is currently performed through Real Time Bidding (RTB), which is the process of selling the ad impressions in real time auctions. In RTB, there exist many different ad networks who are intermediate constructs between publishers and advertisers. As different ad networks connect to different sets of advertisers, from a publisher’s point of view it is critical to send its impressions to the most promising ad networks that can increase their revenue. One approach to choose a particular ad network given the available ad slot is through the so called Waterfall Strategy. In the waterfall strategy, publishers send an ad request to ad networks sequentially based on a predefined and fixed ordering. This strategy is inefficient in terms of time and revenue because often the first selected ad networks may not provide advertisements successfully. We present a method for helping publishers to decide which ad networks to use for each available impression. Our proposed method uses reinforcement learning with initial state-action values obtained from a prediction model to find the best ordering of ad networks in the waterfall fashion. We show that this method increases the expected revenue of the publisher by experiments on ad auction data.

2 Proposed method

We use reinforcement learning to derive the best ordering, namely the one maximizing expected revenue. Since we do not have access to real RTB environment, it is not possible to explore the state-action space. Therefore, we use historical data and consider each sequence of ad requests to fill a certain ad slot as an episode. The complete list of features in an ad request are shown in [1]. We then use a Monte-Carlo algorithm to learn the state-action values. The definition of states, actions and rewards are as follows.

**States:** The combination of *ad tag id, floor price* and *request order* are selected to construct the states because they make a balance between the number of states and the number of observed ad networks for each state.

* Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0)
**Actions:** The actions are selected ad networks. In each state, the model decides which ad network makes the most revenue in the shortest time. There are $N$ possible ad networks and each ad request could be sent to any one of them.

**Reward function:** The floor price is the lower bound of revenue for an impression. We assign the value of floor price as the reward of successful ad requests. Conversely, unsuccessful attempts are penalized by the value -1.

**Initial state-action values:** In order to find initial state-action values, we first find the success probability of sending requests to a certain ad network. For this, we build a classifier that predicts the success probability for each ad request. The multiplication of this probability to the floor price of the current ad request yields the expected lower bound for the revenue of the ad request.

### 3 Experiments

The dataset we use contains the ad requests of one week (20-26 November 2017). We use some part of this dataset for the prediction model and the rest for the Monte Carlo algorithm. For each day, the ad requests are divided into train and test. Figure 1 shows the ROC curves for the success probability prediction. Figure 2 illustrates the cumulative revenue prediction for the test dataset (red curve) compared to the real revenue earned (blue curve). We used the data samples of five days in the Monte Carlo algorithm and the ad requests of November 26 were used for testing the method. The results show that our approach could help publishers not only to fill their ad slots in the shortest time, but also to increase their expected revenue.

### References