UNIVERSITÉ DU LUXEMBOURG

PhD-FSTC-2019-65
The Faculty of Sciences, Technology and Communication

DISSERTATION

Defence held on 21/10/2019 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Manxing DU
Born on 16 August 1987 in Beijing (China)

TOWARDS OPTIMAL REAL-TIME BIDDING STRATEGIES FOR DISPLAY ADVERTISING

Dissertation defence committee

Dr. Radu STATE, Dissertation Supervisor
Associate Professor, Université du Luxembourg

Dr. Björn OTTERSTEN, Chairman
Professor, Université du Luxembourg

Dr. Vijay K.GURBANI, Vice Chairman
Chief Data Scientist, Vail Systems, Inc.

Dr. Mats BRORSSON
Research Scientist, Université du Luxembourg

Dr. Petko VALTCHEV
Assistant Professor, Université du Québec à Montréal

To my beloved family

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Dr. Radu State for offering me the opportunity of the Ph.D study and for the encouragements, guidance, and advice he has provided throughout my study.

Secondly, I would like to express my appreciation to my CET members and jury members: Prof. Dr. Björn Ottersten and Prof. Dr. Mats Brorsson for providing their insights and constructive comments. In the same measure I express my appreciation to Dr. Vijay Gurbani and Prof. Dr. Petko Valtchev for agreeing to join my defense jury and for taking time reviewing my dissertation.

Thirdly, I am grateful to the industrial partner OLAmobile, the Luxembourg National Research Fund, and the Interdisciplinary Centre for Security, Reliability and Trust (SnT) for funding my Ph.D project. Further thanks go to Mr. Antoine Moreau, Dr. Tigran Avanesov and the team at OLAmobile for their support.

My sincere thanks go to Prof. Dr. Jun Wang and Mr. Rael Cline for hosting a research visit to MediaGamma and UCL in London and for the cooperation and discussions with their team.

I would like to thank all the present and past members in SEDAN lab for the fruitful discussions and for making the working environment full of joy. I am deeply grateful to all of my co-authors for the proof readings, discussions, and collaborations.

I would also like to thank my friends Tunhe Zhou, Xuan Sun, Le Zhang, Tian Gan, and Tuo Jiang for their spiritual support; Linlin Wang, Mo Tian, and Xi Chen for the amazing trips together; Siwen Guo, Cui Su, and Menglin Zheng for sharing many unforgettable moments and best of luck to your Ph.D studies!

A very special gratitude goes to my boyfriend Mathis Baden and the Baden family. Words can not express how grateful I am for their love, support, and company during this journey.

Last but not least, I would like to thank my beloved family, especially my parents, my grandmother, and my late grandfather for their love, support and motivations throughout my study and my life in general. I could not have gone so far without them.

Abstract

Real-time bidding (RTB) is one of the more popular mechanisms for trading online ad slots. RTB enables advertisers to directly match the user's profile and the publisher's profile with their ads for every ad display opportunity through real-time auctions. Properly designed and allocated ads enhance users' browsing experiences by providing relevant sources of information without bringing too much interruptions. Positive user experiences encourage more ad engagements, which directly increases high profits for both publishers and advertisers.

In this dissertation, we address three major challenges in the RTB environment for display advertising, namely *user response prediction*, *bidding market prediction*, and *bidding strategies design*. First, user response prediction models facilitate the evaluation of the value of each ad display opportunity. For this purpose, we leverage self-exciting effects in users' purchase behaviors to complete user profiles with temporal features. It improves the performance of forecasting users' future purchase behaviors. Second, bidding market prediction models estimate the cost of winning an auction, which directly impacts a bidder's budget consumption. We propose two market prediction models providing price level and distribution level estimations. The price level model captures and correlates the temporal patterns in the bid request features with the market prices at an aggregated level. It significantly decreases the prediction error compared to uni-variate baseline models. The distribution level model provides fine-grained market price distribution predictions for each bid request. It facilitates estimating the winning probability for any bid price and outperforms state-of-the-art models applied in the RTB domain. Third, we propose two bidding strategies in the single agent and multi-agent scenarios respectively. The former model demonstrates high-efficiency in terms of winning the auctions with clicks under low budget constraint in a stationary bidding environment. The latter model allows each bidder to infer its opponents' behaviors in the partially observable stochastic bidding environment. It shows faster convergence of the strategy training when all bidders adopt and learn their strategies simultaneously in the non-stationary environment.

The proposed solutions have been tested on large scale real-world bidding datasets and also private datasets from the industrial partner, OLAmobile. The results demonstrate significant improvements over state-of-the-art studies.

Table of contents

List of figures

List of tables

LIST OF ABBREVIATIONS

AIC Akaike information criterion

ANLP Average Negative Log Probability

ARIMA Autoregressive Integrated Moving Average

CDF Cumulative Distribution Function

CMDP Constraint Markov Decision Process

CPA Cost per Acquisition

CPC Cost per Click

CPM Cost per Mille

CTR Click Through Rate

CVR Conversion Rate

DASA Deep Attentive Survival Analysis

DDPG Deep Deterministic Policy Gradients

DLF Deep Landscape Forecasting

DMP Data Management Platform

DQN Deep Q-Network

DSP Demand-Side Platform

eCPC Effective Cost per Click

FM Factorization Machine

GSP Generalized Second Price

KG Knowledge Graph

LSTM Long Short Term Memory

MAB Multi-Armed Bandits

MDP Markov Decision Process

MI Mutual Information

NE Nash Equilibrium

MFE Mean Field Nash Equilibrium

MSE Mean Squared Error

KPI Key Performance Indicator

pCTR Predicted Click Through Rate

PDF Probability Density Function

RL Reinforcement Learning

RNN Recurrent Neural Network

ROI Return on Investment

RTB Real-Time Bidding

SSP Supply-Side Platform

Chapter 1

Introduction

1.1 Online Advertising

Online advertising has become a multi-billion dollar industry in recent years. According to the Interactive Advertising Bureau annual report[1], in 2018, online advertising for the first time generated over 100 billion dollar in the US market. It surpassed the second category on the list, TV advertising, by over 30 billions. Fueled by the global proliferation of mobile devices usage, mobile ads have brought significant revenue shift over ads on the desktop from only 9% in 2012 to over 65% in 2018. With more user engagement in the online activities, among all media types such as on paper or radio, online advertising is the only type with rapid growth in the market share since 2014.

Common types of online advertising are *sponsored search* [64], *contextual advertising* [126], and *display advertising* [153]. Instead of blindly serving the ads regardless of the place where they are shown, the relevance, or the audience's preferences, online advertising allows the ad slots to be sold in a programmatic way to realize more efficient ad delivery. *Sponsored search* uses a query-based matching criteria, where the advertisers target the key words in the user's search queries and submit a bid price that they are willing to pay if a user clicks on the ad. As is shown in Figure 1.1, on the Google search's side, on the top of the search results, a few ad slots are available. Google AdWords [38] for example, as the platform used by

[1]https://www.iab.com/insights/iab-internet-advertising-revenue-report-2018-full-year-results/

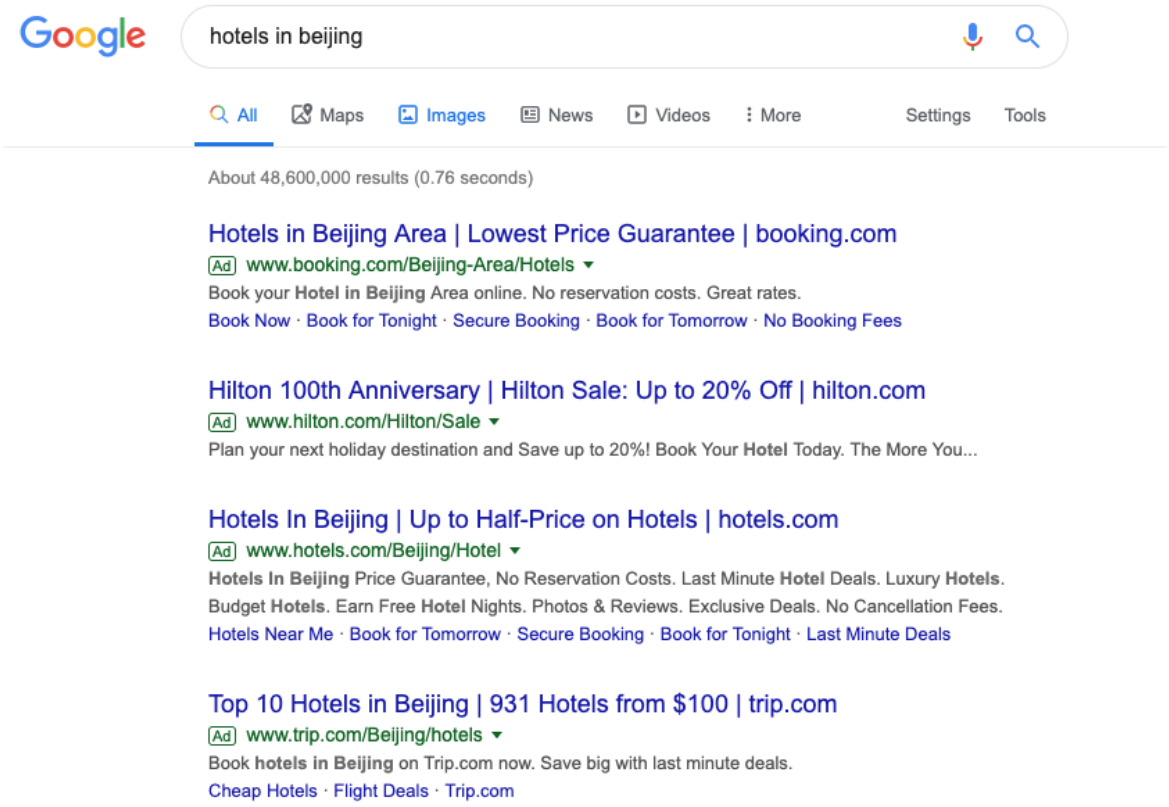
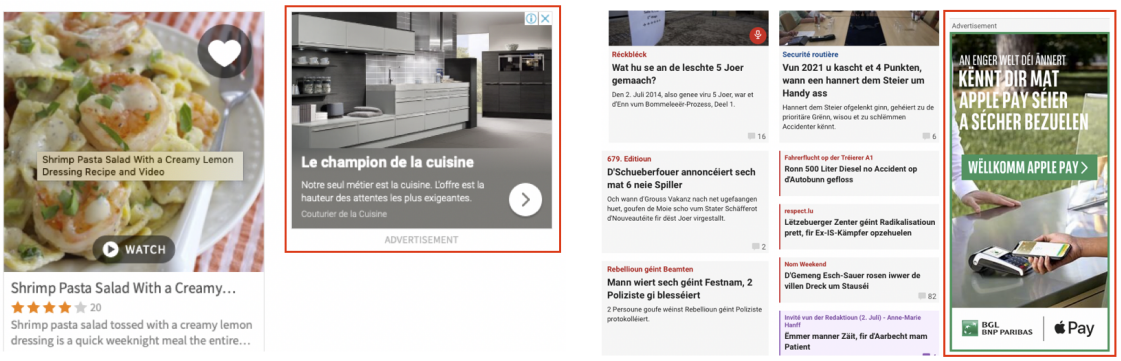


Figure 1.1 An example of sponsored search ad on Google, powered by Google AdWords.

Google, takes a global view and allocates the submitted ads by calculating a rank score [49] in order to optimize its revenue. The rank score is decided by the submitted bids and the evaluation of the quality of the ad in terms of the probability of a click.

Instead of targeting the search query, *contextual advertising* matches the ads with the content of the publisher's site (e.g. in Figure 1.2a) or the use's geographical location (e.g. in Figure 1.2b). Google AdSense [38] is such a platform which serves contextual advertisements. It firstly determines a set of keywords of each webpage and upon each user's visit, same as on AdWords, the platform then ranks the ads based on the keywords-based bid price and allocates the ads on different locations.

Display advertising sells ad slots through direct contracts [65] or by means of real-time auctions. Direct contract means that the impressions are pre-sold to advertisers on a predetermined contract at a fixed price. The publisher guarantees to show the advertiser's ads and meets some pre-defined Key Performance Indicator (KPI), for instance, the publisher can



(a) Content matching

(b) Geographical location matching

Figure 1.2 Examples of contextual advertisements.

agree to deliver a minimum amount of impressions or to guarantee a certain number of clicks or acquisitions.

In contrast, RTB allows the advertisers to programmatically design their strategies for buying ads in a more cost effective way. It allows more fine-grained targeting by directly matching their potential customers for every ad display opportunity. In Figure 1.3, the hotel booking ads are based on the user's earlier browsing behavior on Booking.com which is irrelevant to the content on the news site. In addition to the publisher's site information, the advertisers also receive each end user's profile, such as the device type, the time he/she visits the site, or the geo-location. If the user profile matches the target audience, the advertiser will set a bid price in order to win the real-time auction.

1.2 Real-Time Bidding

An RTB system in general consists of three major parts: Demand Side Platforms (DSPs), an Ad Exchange, and Supply Side Platforms (SSPs). Figure 1.4 shows a simplified overview of an RTB system. Although only one SSP and one DSP are shown in Figure 1.4 for simplicity, in practice, an ad exchange connects multiple SSPs and DSPs. When an end user visits the publisher's site or opens a mobile app, the publisher's end sends a bid request containing its own profile, attributes of the user, and the description of the available ad slots (ad format or size) to the ad exchange requesting for the best matching ads. The publishers usually register their ad inventories at an SSP. It provides better control for determining the minimum inventory price to ensure the publisher's profits and automates the inventory selling

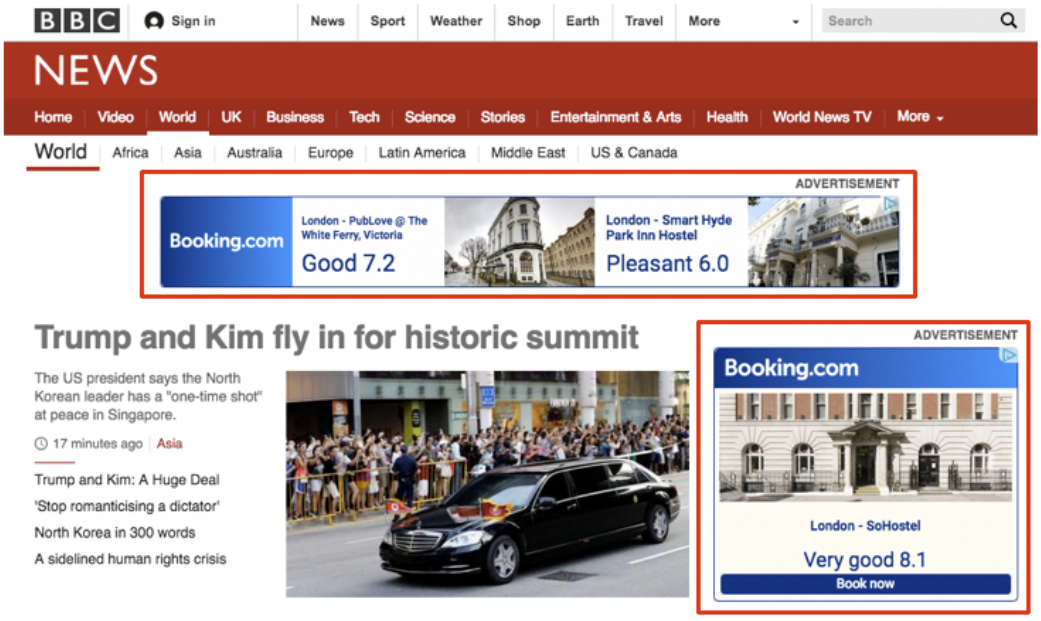


Figure 1.3 An example of display advertising on BBC.com.

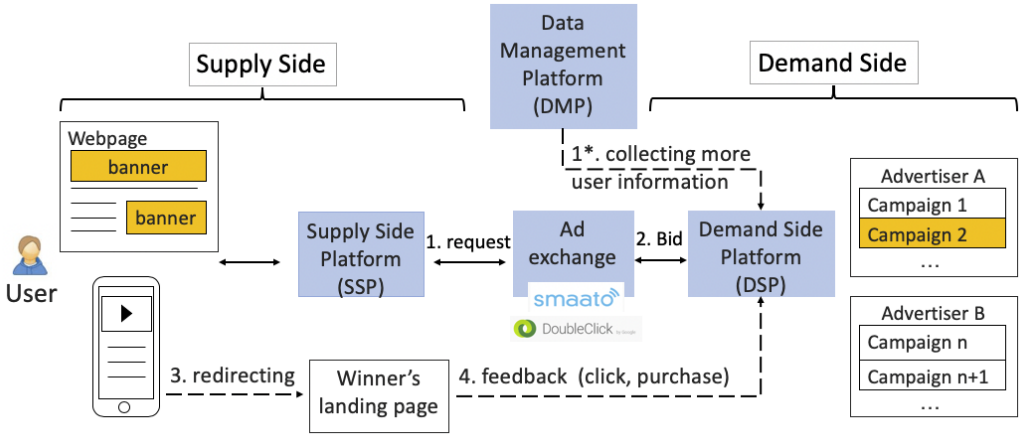


Figure 1.4 An overview of a simplified RTB system.

process with as many advertisers as possible. The ad exchange starts a real-time auction and distributes the request to the advertisers' side.

Advertisers have different forms of ads such as video ads, banner ads, and pop-up window ads. A specific ad that an advertiser wants to show is called an *ad creative*. Advertisers define the budget, targeting user segments, and maybe a restricted list of publishers, together with one or more ad creatives, to create an *ad campaign*. Similar to SSPs, DSPs facilitate the advertisers to purchase ad slots from a variety of publishers by optimally deciding a bid price

for their ad campaigns. Upon receiving the bid request, the DSP selects the campaigns with the relevant targeting parameters. In addition to the information in the bid request, the DSP can also collect more online behavior data of the user from a third-party Data Management Platform (DMP). Based on all the gathered data, the DSP needs to set a bid price in order to win the opportunity of showing the ad, also called an *ad impression*.

The ad exchange receives the submitted bid price from multiple DSPs and notifies the one with the highest bid price as the winner. The winner pays the market price, also called the winning price according to the type of the auction. First price auctions (also known as English auctions) and second price auctions (also known as Vickrey auctions) [76] are the two most commonly used auction types in RTB. As the name suggested, in the first price auctions, the winner pays its own bid while in the second price auctions, the winner pays the second highest bid. The ad exchange sends the winner's ad creative back to the SSP and the ad is shown to the user.

The process of the online auction typically takes place within 100ms during the page loading time. A successful ad impression will attract users to click on it. After being redirected to the winner's landing page, the user may also complete a service subscription or a purchase. The click and purchase information will be sent to the DSP as the feedback to better estimate the value of a similar impression. The DSP evaluates the campaign performance by calculating the Click Through Rate (CTR) or the Conversion Rate (CVR) as the reference for making future bidding decisions. The CTR of a campaign is defined as the ratio of the number of clicks to the number of impressions shown to the users. The CVR of a campaign is the percentage of the pre-specified conversion actions over the total number of clicks.

Ideally, advertisers aim to pay less for a high valued impression to achieve higher Return on Investment (ROI) and in the meantime publishers work on selling their inventories at a better price to maximize their revenue. The advertisers usually have three pricing models to pay for the DSPs, Cost per Mille (CPM), Cost per Click (CPC), and Cost per Acquisition (CPA). The CPM model defines the cost for winning 1000 impression. It is usually used for branding ad campaigns which aims for the wide exposure to users. CPC and CPA are the performance-driven models which charge the advertisers for every click or post-click action correspondingly. Thus, the DSP earns nothing by winning an impression alone. From the advertisers to the publishers, the cash flow in the RTB system goes though many entities and due to the non-transpatrent setting of the ad market, publishers usually do not get as much as the advertisers pay [159]. For instance, a DSP may pay the market price to the publisher by

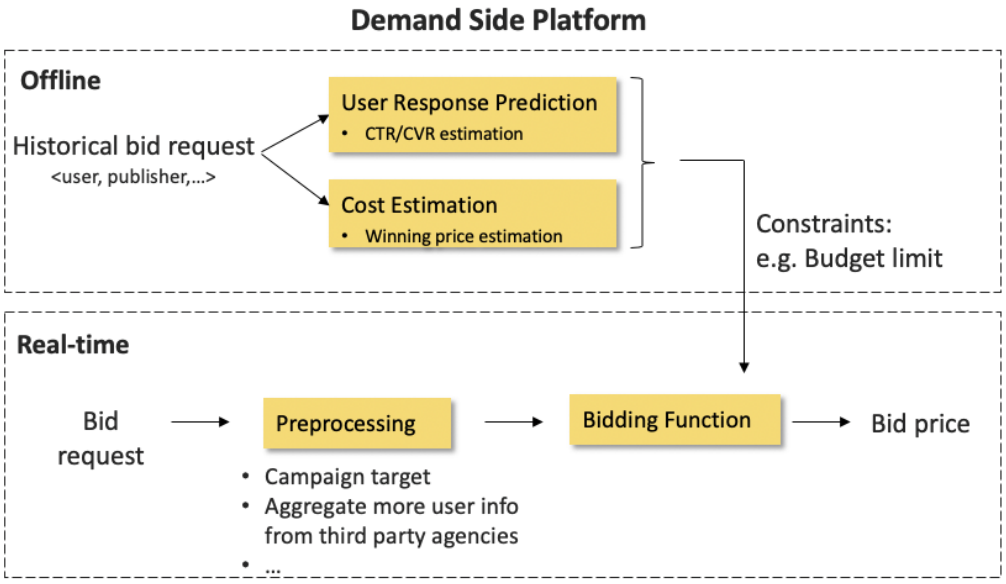


Figure 1.5 Major components on a Demand Side Platform (DSP).

the CPM scheme and could get paid by the CPA scheme of the advertisers. Consequently, the DSP exploits the price discrepancies between these two schemes and earns the difference between the CPM cost and the CPA payoff. Intermediary agencies may exist between the publishers and the advertisers, which could buy an ad slot with low price and sell it with high price [159]. In this case, each entity in the chain takes a percentage of the payment as profit.

The second-price auctions have so far been the dominent model in RTB where the winner pays the second highest price in the market which may be lower than the price at which the publisher is willing to sell. Therefore, SSPs uplift their revenue by setting a hard floor price and a soft floor price [152, 153]. If the highest bid is lower than the hard floor price, the ad slot remains unsold which defines the minimum price for a inventory. If the highest bid price surpasses the hard floor price but is lower than the soft floor price, the auction is turned into a first price auction. Only if the winner bids higher than the soft floor price, the second price auction takes place.

This dissertation focuses on the design of bidding strategies on the DSP's side. Figure 1.5 shows the major components of a DSP. The goal is to design a bidding function which maps each bid request into a competitive bid price and maximizes the profits. The bidding function should evaluate the value of the ad impression through the user response prediction, estimate the competition in the bidding market, and consider the budget limit of each campaign. The user response prediction and cost estimation models are usually trained

offline. Each bid request firstly goes through preprocessing steps on the DSP. The DSP filters out some campaigns with the targeting criteria mismatching the bid request. For instance, the ad campaign may target only users from a specific geo-location or a certain device type. The DSP can also aggregate more user behavioral data from a third party DMP and more customized feature selections. The bidding function integrates the above components and determines a bid price.

1.3 Research Problems

With the research presented in this dissertation, the goal is to design an optimal bidding algorithm which allocates the budget sequentially in order to win impressions with high potential of a click or purchase from the DSP's perspective. The advertisers usually define specific KPIs for each campaign, for example, the total number of clicks, the total number of purchases or the Effective Cost per Click (eCPC). Here eCPC means the average cost per click, which can also be called the expected CPC.

One concern raised by using the number of clicks as the performance measure is click frauds [130, 156, 106]. Some dishonest publishers may create fake websites with ad slots to attract advertisers. Malwares are used as bots to automatically simulate clicking the ads. In this way, the publishers can maintain high CTR which further encourages the advertisers to bid higher for the fake high valued ad slots. It is more difficult to mimic a post-click behavior like purchasing which requires instant spending of money. Therefore, it is effective to focus on predicting the conversion rate.

To observe a conversion, the first step is to win the auction so as to show the ads to the users. A busy DSP may receive millions of requests per hour [77]. The market price of each auction fluctuates heavily over time due to the stochastic competitions in the bidding market. In this dissertation, we investigate solutions for addressing the winning price prediction in the bidding market at an aggregated level and on the per impression level.

An optimal bidding algorithm integrates the winning probability, the potential gain, and the budget limit to calculate a bid price for each bid request. We start from optimizing a strategy for a single bidder where the bidder takes all the external parts in the bidding market as the environment. We further extend the bidding problem into a multi-agent scenario where all

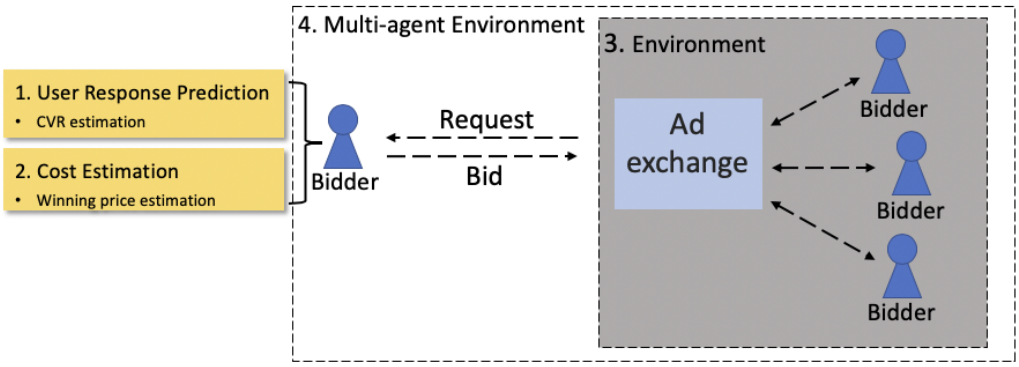


Figure 1.6 An overview of the research problems and their connections. (1). Conversion Rate Prediction. (2). Cost Estimation. (3). Single agent bidding optimization. (4). Multi-agent bidding optimization

bidders in the environment learn and optimize their strategies simultaneously. Figure 1.6 shows the relations between the research problems discussed next.

1.3.1 User Response Prediction

User response prediction is one of the more popular topics in the online advertising literature. Most attention has been given to CTR prediction, which is vulnerable to click fraud. In addition, on the mobile devices, due to the limited screen size and the ad design, users may easily misclick on the ad when they try to tap on the small close button on the ad, known as the fat-finger effect [70]. Comparing with clicks, the conversion event, i.e. the action of a purchase, subscription or similar, directly reflects the user's interest. However, conversion events are even more sparse than the clicks and consequently even harder to predict. In previous studies, the purchase events are commonly treated as independent [124, 84]. As opposed to that, this dissertation deals with the data sparsity of individual users by aggregating them into groups and models the cascading positive effects of the previous purchase events within the group. We introduce new features of showing self-exciting effects of historical purchases and the campaign level CVR changing signals into the CVR prediction model and improves the CVR prediction performance.

1.3.2 Cost Estimation

In RTB, ad impressions are sold upon every user's visit to a web site or an app. A group of unknown bidders with different strategies compete in every auction. In addition, the supply side may introduce a soft floor price to avoid selling its inventory lower than its expectation. The decisions of all the participants in the market determine the winning price distribution. It is essential to estimate the market price which provides a reference for deriving the winning probability of each bid price. The winning probability is the Cumulative Distribution Function (CDF) of the market price distribution with respect to a certain bid price. Previous studies have focused on either predicting the exact market price of each impression [146, 145] or estimating the price distribution [142]. Similar to stock market price [139, 123], the bidding market price can be formed as a time series. The dependence of the market price on time have been neglected by previous works. Therefore, in Chapter 4, we propose to formulate a time series of the average market price and adopt a Recurrent Neural Network (RNN) to model such dependencies and provide a more accurate prediction.

However, predicting the market price on an aggregated level may not be adequate since it does not reflects the price fluctuations on the per impression level. Another challenge is that from each bidder's point of view, in the second price auctions, the market price is only revealed to the auction winner. The other bidders only know that the market price should be equal to or higher than their bids. In this case, the true market price is called right censored. With only partially observable information, the winning probability can be overestimated. Thus, it requires the prediction model to handle the bias in the training data. In Chapter 6, we adopt an attention-based deep neural network model to learn the correlation between the user features and the impression level winning price distribution.

1.3.3 Single Agent Bidding Optimization

The goal of a DSP is to apply a bidding algorithm which makes sequential decisions maximizing the profits within a budget limit. The ad campaign usually defines a daily budget and when the budget is used up, the campaign stays out of the market for the rest of the day. Many efforts have been made to smooth the budget spending process to avoid a fast and early drop out and to reach a wider range of users [148, 83, 4]. These methods are usually called *pacing algorithms* in which the spending speed relies on the traffic volume or performance driven

metrics. The descriptive user profile information in the bid requests and the competition in the market are usually neglected. Other attempts to design bidding algorithms integrate the cost model and the user features but restrict the form of the bidding function to be concave [160, 119]. We treat the bidding market, and all the bidders, as the *environment* and model the interaction between one bidder and the environment as a Markov Decision Process (MDP). In this way, we avoid pre-defining the form of the bidding algorithm and the optimal bid price can be derived by maximizing the value function under constraints in a constraint Markov decision process framework. Another limitation in previous studies is that the market price is usually modeled independently [23, 160]. However, the bidding decisions of all bidders depend on the same bid requests. Without information regarding other bidders, the bid request is considered to be an importance source of information to infer the market. Therefore, instead of an independent model, we introduce a conditional market price distribution with respect to the user features. The proposed method outperforms the state-of-art bidding algorithms in terms of the total number of clicks.

1.3.4 Multi-Agent Bidding Optimization

In RTB, a large amount of unknown bidders with unknown strategies participate in every auction. A traditional reinforcement learning framework, from one bidder's perspective, assumes that the environment is stationary. A single agent interacts with the environment and learns to adapt its behavior. However, in a competing market like RTB, other bidders may learn and adapt their strategies as well, which consequently change the environment to be non-stationary. Therefore, in a multi-agent scenario, for each agent, it is necessary to model the joint behavior of all agents. Most research in the multi-agent learning problems assume that the other agents' actions or states are fully observable [151, 68] or their strategies are known a priori [34]. Dynamic online bidding can be viewed as a stochastic game which requires equilibrium-solving solutions. The actions of other bidders are only shown to the winner of the auctions. In this specific application, since only the second highest bid decides the market price, all the lower bids bring zero gain to the bidders. Although the participants in the market can be numerous, it is not necessary to model them all except the market price. This dissertation assumes a virtual opponent who represents the unknown number of bidders and always bids at the market price and proposes an opponent model to solve the partially observable problem.

Bidding in the online auctions can be seen as *games* defined by Game theory [102] in which two or more rational *players* make decisions on how to interact with each other. In such *games*, if every player follows its strategy and no one can improve its utility by deviating from its own strategy when others fix their strategies, the game reaches to the Nash equilibrium and the strategy is the optimal solution [102]. The optimal strategy in the second price auctions is known as the truthful bidding where each bidder submits its true valuation for the item as the bid price. However, in practice, the bidding process is a sequential decision making process and has a budget constraint which makes the bid price deviate from the true value of each bidder. This dissertation assumes that all the bidders adjust their strategies towards the Nash equilibrium. It proposes a multi-agent framework which integrates an opponent model into the policy learning process as modeling the joint actions. To the best of our knowledge, such a solution addressing the partially observable opponent in the multi-agent RTB scenario has not been proposed in existing literature.

1.4 Summarized Contributions

The dissertation addresses the challenges on the demand side in the real time bidding system. The contributions are summarized as follows:

First, a self-exciting user behavior prediction model is proposed. The focus is on predicting the probability of a purchase event given the user profiles and their purchase history. We focus on the top active user groups with long purchase histories. Two new sets of features are introduced to the prediction model which provide the globally short term campaign performance indication and the intra-group purchase intensity measure. The new feature sets boost the prediction performance with respect to the baseline model significantly for the top user group.

Second, to model the dynamic bidding market and provide reference to estimate the winning probability and the budget spending process, two market price models are proposed. The first model formalizes the market price as time series data and focuses on modeling the average price changing patterns over time. It captures the temporal dependencies of the market price and integrates the aggregated features into the Long Short Term Memory (LSTM) model. The prediction error decreases significantly over traditional uni-variate time series models. In addition, instead of predicting an exact price, a more fine-grained and generalized solution

is proposed to estimate the impression level market price distribution. To differentiate the contributions of different user features to the market price, an attention model is proposed to map the input from the feature space to the market price space. The proposed model outperforms many state-of-the-art bidding market models and serves as the opponent model in the later work to support the bidding strategy learning.

Third, from a single bidder's perspective, we formulate the bidding problem as a constraint Markov decision process. The proposed model considers the bidding risk in the reward function and demonstrates better performances in terms of total clicks against other state-of-the-art bidding algorithms. The market model used in this work is derived from all the winning bid requests and provides coarse-grained estimation on the impression level and neglects the losing auctions with censored winning price. In addition, the proposed model assumes the controlled bidder is the only learning bidder in the environment.

Fourth, to overcome the shortcomings in the above single agent solution, a novel multi-agent bidding optimization framework is provided. In the multi-agent scenario, each bidder models its opponent's behavior by an unbiased opponent model proposed earlier and integrates its estimation into a model-free deep neural network based policy learning process. We demonstrate faster convergence of the learning algorithm with the support of the opponent model comparing with the baseline model.

1.5 Thesis Structure

The rest of this dissertation is organized as follows. Chapter 2 provides a literature review of the related research topics. Chapter 3 presents a user behavior prediction model with the integration of self-exciting effects in the purchasing histories. Chapter 4 investigates the bidding market modeling both on the aggregated level and on the impression level which provides insights into the highly dynamic bidding competitions. Chapter 5 discusses an optimal bidding solution for one single bidding agent combining a user behavior model and a market model. Chapter 6 takes one step further, by allowing each bidding agent to be aware of other opponents in the same bidding environment and study the optimal bidding strategy for the multi-agent scenario. In the end, the conclusions of the dissertation and the future work are discussed in Chapter 7.

1.6 Supporting Publications

This section lists the publications related to each chapter.

1. Manxing Du, Alexander I. Cowen-Rivers, Ying Wen, Phu Sakulwongtana, Jun Wang, Mats Brorsson, and Radu State. Know Your Enemies and Know Yourself in the Real-Time Bidding Function Optimisation. In the Proceedings of the 19th IEEE International Conference on Data Mining Workshops (ICDMW) 2019. **Main text of Chapter 6, 4**.

2. Manxing Du, Christian Hammerschmidt, Georgios Varisteas, Radu State, Mats Brorsson, and Zhu Zhang. Time Series Modeling of Market Price in Real-Time Bidding. In the Proceedings of the 27th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), 2019. **Main text of Chapter 4**.

3. Manxing Du, Redouane Sassioui, Georgios Varisteas, Radu State, Mats Brorsson, and Omar Cherkaoui. Improving Real-Time Bidding Using a Constrained Markov Decision Process. In the Proceedings of the 13th International Conference on Advanced Data Mining and Applications (ADMA), 2017. **Best paper runner up award**. **Main text of Chapter 5**.

4. Manxing Du, Radu State, Mats Brorsson and Tigran Avanesov. Behavior Profiling for Mobile Advertising. In the Proceedings of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT), 2016. **Main text of Chapter 3**.

Chapter 2

Related Work

This chapter provides the summary of the state-of-the art research work in RTB. It presents an overview of the display advertising both on the supply side and on the demand side. As this dissertation focuses only on the demand side, it further breaks down the research topics based on different components in the bidding decision making process. In Section 2.2, as the basis of understanding the user behavior, the techniques of predicting the CTR and CVR are discussed. The section supports the reward modeling parts of Chapter 5 and 6. Section 2.3 discusses the approaches for market modeling which serves as the technical background of the solutions in Chapter 4 and the opponent model in Chapter 6. In Section 2.4, it focuses on the optimal bidding strategy design which provides the basis for Chapter 5 and 6.

2.1 Programmatic Advertising

Programmatic advertising allows the advertisers to effectively set their targets in different forms. In sponsored search [132], advertisers match their ads with the keywords in the search queries. In contextual advertising [3], the advertisers select ads relevant to the content on the publisher's site. However, in these two forms of ads, the advertisers are unable to distinguish the different individuals with the same search query or viewing the same content. In display advertising, the advertisers directly target their potential customers on the per ad impression basis. More relevant ad impressions may stimulate users click on the ads or to complete the conversion which also helps the publisher's side to attract and keep their customers. In this

way, both the advertisers and the publishers benefit from more effective ad deliveries. As mentioned in the previous chapter, in RTB, DSPs and SSPs usually work on behalf of the advertisers and publishers respectively to manage the online advertisement trading. In the ad trading game, the DSPs aim to set a competitive bid price for the ads in their inventory to win more impressions with higher CTR or CVR with relatively low cost, a.k.a, higher ROI. On the publisher's side, SSPs control and optimize the density and the selection of the ads on their web page to enhance the user experience. Meanwhile, the SSPs are allowed to set a reserve price, which is also called the floor price to ensure the minimum price paid by the advertisers [152]. When the reserve price is higher than the winning price in the bidding auction, the winner has to pay the reserve price instead. By adjusting the reserve price, publishers could potentially obtain more revenue.

In this dissertation, the scope is limited to the decision making on the DSP's side. The following sections present the related work for different components in the DSP system in particular.

2.2 User Response Prediction

User response prediction is important to evaluate the value of an ad impression. By winning an auction, the ad of the winner is shown to the users. However, the advertisers are not satisfied with only the ad browsing, they focus more on the follow-up actions such as, clicks, purchases, or service subscriptions. Therefore, modeling the CTR or CVR directly estimates the effectiveness of the advertising and it has been extensively studied in domains of sponsored search [32, 28], contextual advertising [25, 67] and display advertising [96, 71].

The research challenge mainly lies in dealing with the model input. The prediction model is a mapping from the feature space to the probability of a click or conversion. The model input is usually a mix of categorical features, such as the publisher and user attributes (e.g. location, device type, webpage context etc.) and numerical features such as user behaviour summaries (e.g. user's revisiting counts or the time between two visits).

Linear models are widely used for CTR prediction which assume the input features are linearly correlated, ranging from logistic regression [162], Poisson regression [31], and Bayesian probit regression [49]. To learn the nonlinear dependencies between features, in [133], the authors choose gradient boosting trees to cope with the interactions between

individual features. However, the individual features may be insufficient and have low correlations with user click/convert intentions. Adopting conjunction features into linear models is introduced in [2, 124]. The conjunction features are constructed from Cartesian product of each pair of single features. It is equivalent to a degree-2 polynomial mapping which learns a weight for each feature pair. To reduce the computational complexity of this method, instead of having one weight per cross-feature, Factorization Machines (FMs) proposed in [121] map each feature to a latent vector and use the inner product of the corresponding latent vectors to model the feature conjunction effects [97, 107]. In [72, 71], the authors take one step further from FMs and introduce field-aware FMs which map several latent vectors for each feature depending on the values of other features it is combined with. So far, those methods are limited to address only order-2 feature interactions. However, in practice, higher order feature interactions may also contribute to user's click or purchase intentions, which are usually hidden and not given as a prior.

Deep neural networks are able to address the sophisticated feature interactions directly without any manual feature engineering. However, in the online advertising, the mixed typed features are usually represented as one-hot-encoded vectors with the number of dimensionality as same as the number of feature values. This results in tuning over millions of parameters in the neural network which is prone to overfitting and results in slow convergence. In [158, 52], the authors combine the advantage of FMs and neural networks to reduce the dimensionality of the input features first and model the latent feature structure to provide more accurate CTR prediction. More CTR models empowered by neural networks have been focusing on processing ad images [29], integrating the features of products from users' purchasing history [168], or applying feature crossing at every layer in the deep neural network to address the features interactions in the hidden space [141].

However, the sequential nature of purchase and click events are neglected in the above studies. In the online advertising area, user's browsing and click history are intuitively modeled as sequences and RNN based networks are applied to map the user features to the click through rate [165, 154]. These models implicitly encode the previous users' clicks or purchase interests sequentially into the input structure without considering the self-exciting effect in such events.

Hawkes processes [54], as a particular type of point processes, naturally accommodate modeling discrete and correlated events over continuous time. The events are sorted by their arrival time and each preceding event is assumed to increase the probability of a new

event. It has been applied in diverse areas, such as earthquake prediction [108], stock market prediction [7], and civilian deaths prediction [85]. In recent years, it has been widely applied to study the information cascade in online social media services such as Twitter [24, 167] and Microblogs [85].

In online advertising, the authors of [150] are among the first to model and predict conversion rate using Hawkes processes. Their contribution consists in a model that leverages mutually exciting point processes such that advertisement clicks and purchases are modeled as dependent random events in continuous time. This model is furthermore used to predict future purchases and clicks without fine tuning to the user profile level granularity. More similar work extend Hawkes process to predict multiple types of events, such as commenting or posting in an online forum [86] or distributing different categories of coupons for online shops [95].

Instead of focusing solely either on the high dimensional bid request features or on the time of the targeted events, this dissertation constructs the time series based features from the ad purchase events and combines them with user features to provide prediction results.

2.3 Cost Estimation

In RTB, the online auction is usually the second price auction where the winner is the one with the highest bid and pays the second highest price. Therefore, the cost of winning an auction is decided by the competitors in the market, which is called the winning price or the market price. The market price prediction is important for bidding strategies design since it determines the budget consumption and has an impact on the profits as well. In [90], by having a winning price prediction model together with a CTR predictor, the proposed bidding function outperforms the conventional CTR based bidding function [111] in terms of clicks and the eCPC.

Many research works have been focused on designing more accurate market price prediction models in RTB [117, 142, 146]. The existing literature can be summarized into two categories: *parametric* and *non-parametric* models. In the first category, the models directly predict the market price for each impression [90, 146, 87, 145]. A very straightforward solution is linear regression which takes the features extracted from the impression as the input and predicts a single price [146, 87, 116]. Nonetheless, it fails to capture the true distribution of the market

price. Another type of solutions assume the market price distribution follows a specific form, for example, a log normal distribution [160, 37, 145] or a Gamma distribution [169]. In [145], the authors proposed a general log likelihood function which provides an easy adaption for three basic distributions, namely normal, log normal, and Gumbel distributions. In addition, it presents a deep learning structure to better capture the non-linear correlations between the impression features and the market price.

However, the temporal dynamics of the market price have been largely neglected. One of the most popular time series models is the Autoregressive Integrated Moving Average (ARIMA) model [26], which assumes the linear dependencies among the time series values. In [1], the authors considered using ARIMA models to predict the averaged winning price at every 15 minutes interval. The authors in [116] demonstrate the use of a generalized linear model to infer the mean shift of the market price as a function of the floor price set by the publisher. A strong autocorrelation of the market price is shown to exist on an hourly scale.

In practice, time series data are often generated by heterogeneous variables which introduce latent nonlinear patterns into the time series. RNNs have shown their capability of modeling nonlinear dependencies and predicting sequential data in the areas of natural language processing [98], music generation [22], and extreme event forecasting [79]. One of the most popular structures of RNN is the LSTM network, which effectively avoids the vanishing gradient problem in the vanilla RNN model when dealing with long sequences.

Inspired by [116] and based on the observation of the price dispersion on a per bid request basis, this dissertation aggregates bid requests by time and formulates a time series of the averaged market price per aggregated segment. It adopts an univariate LSTM-based RNN to predict the dynamic market price, and then demonstrates the prediction improvements by including encoded user features as exogenous variables.

So far, the discussed parametric solutions are not flexible in practice and are vulnerable to the changes in the feature space. Instead, the *non-parametric* models predict the bid price distribution, named as the bid landscape, without any prior assumption of the distribution form. In addition, it is also straightforward to calculate the winning probability of a certain bid price by taking the CDF of the market price distribution. A common approach is to construct the Probability Density Function (PDF) of the historical winning price [23]. However, one single model fails to capture the divergence in different data segments and competitions in the market which leads to biased predictions. For instance, the same bid price may have

different winning probabilities at different times of the day or with requests from different location. In [37, 142], decision tree based models are proposed to split bid requests into subgroups based on the KL-Divergence between the distributions of child nodes.

Another challenge in the market price prediction is the right-censored problem. Since only the winner can observe the true market price and in case of losing, the bidder only knows the market price should be equal to or greater than its bid price. To address the data censorship, the survival model is commonly applied to estimate the time until the occurrence of a particular event, for instance, the survival time of patients in the medical domain [100], the time of unemployment in the social science study [45], or the time taken for an ad click in online advertising [18]. The survival model has been adopted into the market price prediction solutions either on an aggregated level [142, 164] or on the impression basis [117, 118]. In [117, 118], the authors proposed to adopt the deep recurrent network to model the sequential pattern in the feature space of the individual user and the recurrent dependencies between the bid price, and estimate the market price distribution for each bid request. However, the features may not only be limited to the sequential dependencies. The transformer model [136] is the first model relying entirely on self-attention to compute a representations of its input without using convolutions or sequence aligned recurrent neural networks [50]. It has fueled much of the latest developments, such as pre-trained contextualized word embeddings in the natual language processing domain [112, 39, 113].

This dissertation adopts the transformer model to generalize the sequential dependency in the feature space combined with the survival loss function to provide an impression level market price distribution prediction.

2.4 Bid Optimization

A bidding strategy is one of the key components of online advertising [10, 47, 160]. The goal is to optimally spend the budget to win more effective ad display opportunities, which have higher chances of getting positive user feedback, e.g. clicks or conversions and to maximize the profits. The advertisers usually set up ad campaigns with a certain budget and a lifetime. If the budget is exhausted during the campaign's lifetime, the campaign has to be dropped from the bidding market and loses the monetizing chance for the rest of the time. There is rich literature focusing on designing budget pacing algorithms to smoothly

win the impressions during a budget day such that the advertisers cover broader audience and achieve sustainable impact [4, 83, 147, 148, 163]. In sponsored search, the budget pacing is implemented on the keyword basis in the Generalized Second Price (GSP) auction [42]. The sponsored links are ranked by the submitted bids. The ad with the highest price is shown at the top and pays the second highest bid. Following the same logic, the ad with the second highest bid is shown at the second place and pays the next highest bid. In this type of auction, the dominant strategy is not the truth-telling strategy as in the second price auction. Therefore, more works have focused on defining new methods for computing unique equilibrium in the GSP auctions [13, 42].

In RTB, the second price auction is commonly used which is unlike the GSP auction, only the winner pays the second highest price and the winner's ad is shown to the user. Budget pacing solutions are also discussed in this context [44, 83, 148, 20, 69]. The proposed algorithms model the budget spending schedule on an aggregated level with respect to discrete time slots. They either assume the bid requests are from a fixed user segmentation [44, 69] or calibrate the budget spending rate based on the historical spending schedule [83, 148]. These solutions neglect that the budget spending rate depends on the competition in the market. In addition, they are not fine-grained on the per impression level.

More efforts have been made to design an impression level bidding strategy which maps the impression features to a bid value which maximizes the campaign performance. To find such mappings is usually formulated as a functional optimization problem. An optimal bidding strategy should take the user response prediction, the auction winning probabilities, and the given budget as three major components. The previous research work can be categorized as *performance-driven models*, *risk-aware models*, and *comprehensive models* based on the different components they focused on during the bidding optimization process. In each group, the algorithms may not consider all the components at once in the same framework.

Performance-driven models emphasize the importance of the advertiser's interest (e.g. the cost per click, the total impressions, the number of clicks or conversions). In [111, 149, 30], the bid price is linearly correlated to the predicted CTR without considering the other two components. In [93], the optimization goal is increasing the ad exposure to the audience, a.k.a brand advertising without considering the future user response. It assumes the market price is already known as prior knowledge.

Risk-aware models address various uncertainties in the RTB environment. For example, the prediction of the CTR/CVR model is usually directly used as the optimization goal without considering the prediction error. In [155], the authors adopt a Bayesian logistic regression model to predict the CTR distribution instead of a single CTR. The proposed model introduces the standard deviation of the predicted distribution as the prediction risk into the bidding function. In [164], the authors address the problem of data distribution discrepancy in the training and test data and provide an unbiased bidding model. In addition, many works have also focused on integrating the winning probability into the bidding function. The winning probability can be interpreted as the risk of losing an auction or the probabiliy of paying the cost [160, 90, 135, 87, 47].

So far, the above solutions train the CTR model and the market model independently neglecting the dependencies between them. For instance, it requires the CTR prediction model to be more fine-tuned when the market model has low confidence and vice versa [119]. *Comprehensive models* unify the CTR prediction and cost estimation models and jointly optimize them towards the profit goal as an end-to-end framework [120, 119].

However, the functional based methods have strong assumptions of the model form, fail to incorporate the dynamics in the bidding environment and neglect the interaction nature of the bidding process between the bidders and the bidding market. Reinforcement Learning (RL) is a research topic known for enabling agents to learn the best actions to take by interacting with the unknown environment. Combined with deep learning, the deep RL algorithms started gaining successes since Deep Q-Network (DQN) proposed by Mnih et al. [101]. Without any prior knowledge of the environment, the DQN based algorithm outperforms human players in 49 Atari games. Similarly, the AlphaGo algorithm proposed by DeepMind beats the world-level top Go players [129].

Deep RL has been widely applied on solving decision making problems in online advertising applications. The models fall into two major frameworks, namely the Multi-Armed Bandits (MAB) [131] and the MDP. In both models, the key components are the *states*, *actions*, and *rewards*. In [46, 66, 127, 15], the authors fit the banner delivery and the ad allocation problems into MAB based models. The rewards are the number of ad clicks or the profits. Most of the works assume no costs for showing the ad impressions and thus considers no budget constraints. In [40, 11], the budget constraint controls the number of a specific action to take. As a single state model, the limitation of MAB is that in case of considering multiple states, multiple MAB models are required for each state [15].

As a more scalable solution for multi-state problems, MDP based models handle the state transition based on the Markovian property. In [10], the authors proposed a bidding function for sponsored search. The keyword bidding process is formalized as an MDP, where the number of auctions and the budget limit are the states, the discretized bid prices are the actions, and the total number of clicks are the rewards. By extending the concept in [10], the authors in [23] formulated a RL based bidding function into the RTB system by extending the impression level features into the state. They implicitly correlate the user features with the auction winning rate approximation by multiplying the average CTR to the density function of the market price.

This dissertation extends the work in [23] by proposing a Constraint Markov Decision Process (CMDP) based model to directly account for budget constraints and implicitly take the impression level information in the predicted CTR as the state to find the optimal bid price. In addition, in [23], the bid price is set in two steps: state value lookup and action calculation. In contrast, the model in this dissertation solves the bidding optimization problem with linear programming which derives the optimal bid price for each state. In this way, the bid price can be set after a single lookup per bid request. The previous solutions are all model based which requires the agent to model the state transition dynamics. It is computationally expensive when the dataset gets large. Model-free approaches are proposed to break the limitation of directly modeling the dynamics in the bidding environment [144, 166].

The RL based solutions are all from one agent's perspective and assume the other agents are part of the environment. However, the real-world applications are usually more complex and have multiple agents involved which makes the single RL solutions not applicable. In multi-agent environments, from each agent's view, when its opponents adapt their behaviour, the changes affect every agent and the environment becomes *non-stationary*. To address the non-stationary environment is the main challenge in the multi-agent learning [58, 105]. It is essential for each agent to account for how other agents react and take the joint behaviour to learn its own strategy. Such scenarios are called *stochastic games*, which require equilibrium-solving approaches for the policy learning. These approaches can be categorized by how the non-stationary behavior are handled [57]. In most of these studies, the learning agent assumes its opponents are acting in a specific way, either using a fixed strategy to minimize each other's reward [92] or using a mixed strategy drawn from a set of known strategies [34]. In case the strategy is not explicitly defined, Nash-Q learning is introduced with the assumption that all the agents are adapting their strategies to converge to the Nash Equilibrium (NE) [61],

which is called the NE strategy. In practice, the Mean Field Nash Equilibrium (MFE) is used to approximate the Nash equilibrium when the number of agents is large [151].

In RTB, the existence of MFE under budget constraints has been theoretically proved in [63, 51]. Both studies showed that to reach the MFE, each agent takes the known fixed market price, budget, and the observed reward distribution (e.g. the estimated click through rate) to estimate the value function. In [68], Jin et al. aggregated bidders into clusters and applied the Deep Deterministic Policy Gradients (DDPG) algorithm on the cluster level (as one agent) to simulate the multi-agent environment. It demonstrates the profit gain per bidding cluster under the competing and the collaborating settings and assumes that each agent knows each other's state, action, and reward. In practice, the states (e.g. the budget, the current obtained reward) of the other bidders are usually unknown. The actions of other bidders are partially observable through the market price only to the winner of each auction. In this dissertation, we provide a multi-agent bidding strategy by extending the work in [68] with the generalization of playing with partially observable opponents. Given that the number of bidders in the bidding market is large and their strategies remain unknown, inspired by the MFE theory, the solution in this dissertation aggregates the set of bidders who bid the market price in each auction as a virtual bidder and introduces a market price prediction model as the opponent model. From each bidder's view, it assumes all the other bidders adapt their strategies towards the mean field equilibrium without any irrational drastic changes.

Chapter 3

Behavior Profiling for Mobile Advertising

3.1 Background and Motivations

The mobile advertising industry is estimated at 100 billion worldwide for 2016[1], being driven by the steady increase in tablet and smart phone usage. Targeted profiling of users consists of identifying how users behave in this environment such that relevant advertisement banners can be served. This profiling becomes even more important in the context of RTB platforms, where online bidding and auctions are performed at millisecond time scales and accurate predictions are essential for computing the likelihood that a given user profile is a potential purchaser. The design of bidding strategies immensely relies on the CTR and CVR estimation [160]. In this chapter, we focus on describing our experiences and results in building a profiling engine to improve the CVR prediction performance for a mobile advertising performance company.

Basic user profiling data in the bid requests are mostly related to the browser, time of the day, type of device and short term (24 hours) tracking history. Given such a user profile, the mobile advertising platform has less than 100 ms to decide which campaign might be the most profitable one. The CTR or CVR prediction is important for the advertiser to

[1]https://www.emarketer.com/Article/Mobile-Ad-Spend-Top-100-Billion-Worldwide-2016-51-of-Digital-Market/1012299/

evaluate its bidding performance. Many research efforts have been made on CTR and CVR predictions [122, 107, 84]. Feature engineering is one of the most important mechanisms to enhance the performance of prediction models. For instance, visual features can be extracted either from a set of pre-defined image attributes [33] or by using neural networks directly processing the ad images [29]. More efficient feature conjunctions modeling for the CTR prediction is provided by factorization machines based methods [72, 109]. Instead of treating all ads independtly, the ads from the same advertiser or are shown on the same page may share more latent information. Such correlations are modeled by constructing hierarchical structures of pages and ads [97] or by jointly training user's browsing and response prediction models [157].

However, in previous works, either the CTR or CVR of each bid request is predicted independently over time neglecting that users' future behaviors may depend on their historical behaviors. In addition, comparing to ad clicks, conversions usually require more engagements such as subscribing to a service or completing a payment. Therefore, some advertisers adopt the CPA model which means that they pay to the DSPs for every completed conversion [94]. For this purpose, we focus on the CVR prediction for a mobile advertising platform.

To address the problem of lacking historical information in the CVR prediction models, in this chapter, we formulate the user's purchase events as a stochastic point process. Two temporal features are constructed by modeling the self-exciting effects in the purchase behaviors and by tracking the global campaign level performance signals.

Before detailing these contributions in the remainder of the chapter, we briefly summarize them below:

- We provide insight into different approaches and their corresponding performances for learning and predicting user behaviors using static features such as location, software, and time of purchase.

- We integrate global campaign level signals into the prediction approach. The rationale behind this approach consists in adding exogenous information corresponding to a campaign (and thus summarizing multiple user profiles) in a similar way to which trading signals are used in trading platforms.

- We propose an extended user profile which includes temporal modeling using Hawkes process. The intensity of a Hawkes process at a given time depends on all previous

observed events and the influence of past events decays exponentially. The rationale behind such a model is that mobile users interact not only with the mobile advertising platform, but also among themselves such that social influence within a group can be a factor for more purchases. However, we have observed that in some cases, the behavior can be self-correcting, and recent user purchases tend to impact negatively on future purchases. This is probably due to purchases which were regretted by users and where the negative feedback influenced other potential users.

- We also discuss the adequate amount of training data that can be used and the frequency of model updates.

3.2 Feature Engineering

Feature engineering is the fundamental step for building a prediction model. We consider three sets of features in this dissertation, defined as static features, temporal features, and feedback features.

3.2.1 Static Features

The basic feature set can be divided into three groups: user side features, advertiser side features, and publisher side features. User side features include the time stamp of each event, location, operator, device related details such as browser and device type. campaign ID and the vertical type belong to the advertiser side features, while publisher side attributes are represented by publisher ID and type. More statistical summary about the features can be found in Section 3.3.1.

3.2.2 Temporal Features

Each user profile's purchase history can be represented as a stochastic point process $N(t)$, which represents the number of events accumulated until time t. The homogeneous Poisson process [74] is a special class of point process, which assumes the intensity of the event λ is independent of the past and the mean of the number of events during a certain time period t

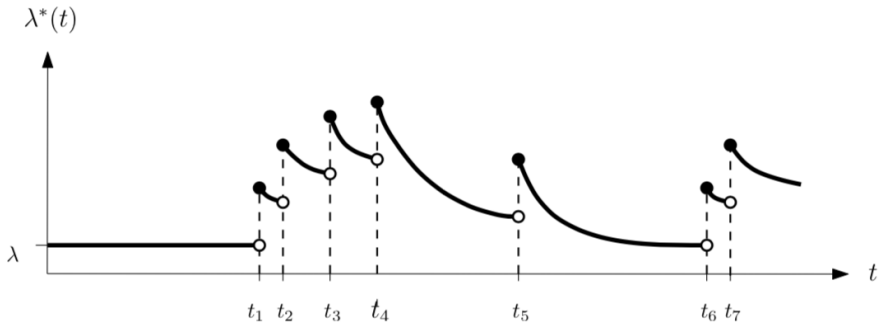


Figure 3.1 Example of a self-exciting point process.

can be calculated as $\lambda(t)$. Equation 3.1 shows the intensity function of a Poisson process, where $N(t+\tau)-N(t)$ is the total number of events in the time interval $(t,t+\tau]$.

$$\lambda(t) = \frac{N(t+\tau)-N(t)}{\tau} \tag{3.1}$$

A Hawkes process [54] models the occurrence of an event that depends on previous events. Each occurrence of the event stimulates the subsequent events which is called the *self-exciting* effect. The exciting influences decay over time. Figure 3.1 [80] shows an example of the conditional intensity function of a Hawkes process. The conditional intensity function is defined by Equation 3.2 [115], where $\mu > 0, \alpha > 0, \beta > 0$, t denotes the time since the start of the process, and μ is the baseline intensity, which is equal to the intensity of a homogeneous Poisson process. The historical events prior to the current time t are represented as H_t and the time of the i^{th} past event is t_i. In this model, the parameter α measures the intensity increase stimulated by the previous events and β controls how fast the effect decays over time. In other words, the further back the event in the process, the less impact it has on future events. Overall, the intensity of a Hawkes process contains the accumulated effects of all the past events prior to the current event, thus it keeps changing over time. We use the conditional intensities of the Hawkes process as the new temporal features.

$$\begin{aligned} \lambda(t|H_t) &= \mu + \sum_{t_i<t} g(t-t_i) \\ &= \mu + \sum_{t_i<t} \alpha \cdot e^{-\beta(t-t_i)} \end{aligned} \tag{3.2}$$

3.2.3 Feedback Features

In addition to the temporal features described in the previous section, we also consider the change of conversion rate in the past as an indicator for the future purchase intent. Considering that advertisers notify of the success of conversions as feedback with uncertain delay and that user may not complete the purchase right after clicking the ad, we first examine the distribution of the time between two purchases. Let t be the current hour and CVR is computed for hour $(t-1)$ and hour $(t-2)$. The CVR change during these two hours is denoted as increase, decrease, or constant.

3.3 Experiments

3.3.1 Data Overview

We obtained a real-world dataset from OLAmobile, a global mobile advertising performance company in Luxembourg. The dataset contains one week of ad clicks and purchases logs of mobile display advertisements in July 2015. Table 3.1 shows the cardinality of each feature in the dataset. The hour and weekday are extracted from the timestamp of each event. Instead of tracking each individual user's purchase history, we use the unique combinations of country, browser, operating system, and operator to construct user profiles. By using the aggregated user profile, more purchase events can be collected which facilitates the model training with the temporal features proposed in this study.

During one week, there are $\sim$14M clicks events generated by $\sim$17K user profiles. After each ad click, the notification of conversion may be sent back by the advertisers with delays which can be up to days. The label of conversion will be added to each click event accordingly. The delay of the conversion has been discussed in a few studies. In [124], the statistics from Yahoo ad exchange shows that 86.7% of the conversions happen within 10 minutes after the ad click, while according to another study with the Criteo dataset [27], within one hour of the clicks, only 35% of the conversions can be observed. However, in our dataset the time of each conversion is unknown. We assume the purchase delay and the delay of the purchase notification from advertisers to the mobile advertising platform are constant for each ad campaign. Based on this assumption, we estimate the purchase time to be the same

Table 3.1 Log Data Statistics.

Feature	Cardinality
hour	24
weekday	7
country	225
operator	397
device	21018
hardware	4
browser	12
operating system	14
campaign	520
vertical type	5
publisher	1503
publisher type	5

as the click time, which keeps the value of time interval between two purchases to be closer to the reality.

The training set contains 5 days of data, and the last 2 days of data are used as test set. A well-known public dataset provided by iPinYou [88], one of the biggest DSP in China, is also available. However, in their dataset, 5 out of 9 campaigns do not contain any conversion events and the amount of conversion data is insufficient to construct time series. Therefore, in the following sections, the results are only based on the dataset from OLAmobile.

All the features in our dataset are categorical features. For example, the feature *country* contains the index of each country, which cannot be treated as numerical features which are ordered by their values. The one-hot-encoding method is used to transform the discrete features to binary vectors for prediction models to process. For instance, if a feature with cardinality of 3: [Firefox, Chrome, Safari], three columns will be needed for the new feature vector. Each categorical variable could be expressed as [0, 0, 1], [0, 1, 0], and [1, 0, 0]. Consequently, after one-hot-encoding, the feature space will tremendously expand to thousands or even millions of dimensions.

3.3.2 CVR Change over Time

The dynamics of the CVR throughout the day and a week is shown in this section. For simplicity, only the top 5 campaigns ranked by their generated revenues are selected as an

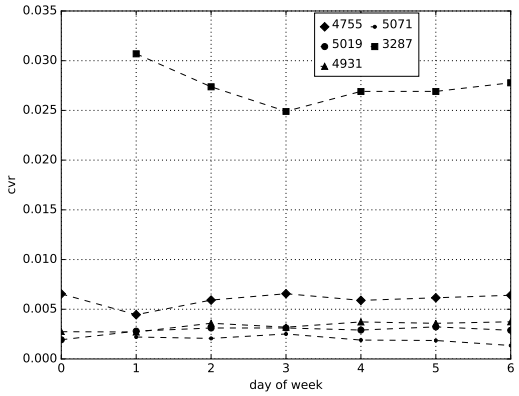


Figure 3.2 CVR change over a week.

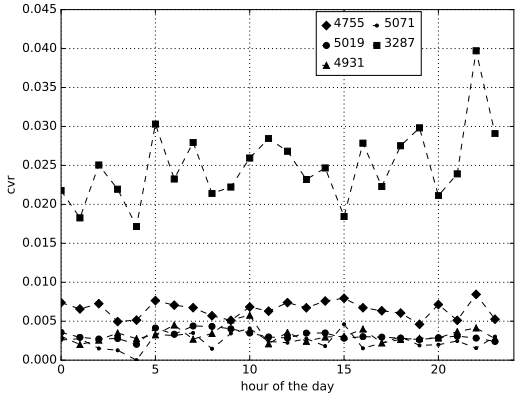


Figure 3.3 CVR change over a day.

example. In Figure 3.2, the x axis represents the consecutive days from Monday to Sunday, respectively. It shows that the CVR fluctuates over multiple days for campaign number 3287 while the CVRs of other campaigns maintain a certain value. One explanation is, campaign 3287 is a local campaign which is only available in one country, the other campaigns at the bottom are global campaigns which are launched in over 200 countries, their daily CVR is averaged over all the users. Similar observations are shown in Figure 3.3, during a day, the CVR of the local campaign varies over each hour while global campaigns have relatively steady conversion rates. It leaves the CVR prediction to be very challenging and requires more fine-grained solutions for each campaign. Given the current timestamp of the ad click, we propose to monitor the trend of CVR changes during the past two hours for each campaign as an additional feature for the prediction model. It introduces the short-term memory of the historical user's behavior for predicting the future behaviors.

3.3.3 Self Exciting Point Process

In this study, the purchase history of each user profile for each campaign is considered as a series of random point processes. The time of each event is obtained from the timestamp in the log. Fitting the time series of user's purchase histories into both Hawkes process and Poisson process helps us to estimate the user's purchase intent at any given time. Our hypothesis is that the current historical purchase from the same user group increases the intra-group purchase probabilities with the effect decaying over time. This is known as the self exciting point process.

Considering that over 500 campaigns are present in our dataset with over 17K user profiles, we first test the Pareto Principle [104], also known as 80-20 rule, to target top campaigns in terms of revenue and eCPM and top user profiles with more click and purchase activities. Figure 3.4 shows that the top 10% of campaigns contribute to 80% of revenue. Campaigns are also ranked by eCPM, which measures the effectiveness of each campaign. The union of the two sets of top 10% campaigns is defined as top campaigns. Correspondingly, the top active user profiles are introduced by ranking the number of clicks, the number of purchases, and the total revenue. As is shown in Figure 3.5, the distribution is highly skewed. For example, only 4.9% of user profiles generate 80% of the clicks and 2.2% user profiles contribute to 80% purchase. Regarding to revenue, 80% of the revenue is produced by 1.1% of user profiles. Meanwhile, we found that 7% of the user profiles with most clicks have no purchase records. The behavior of the click-only (without conversions) user profiles is not analysed in this dissertation.

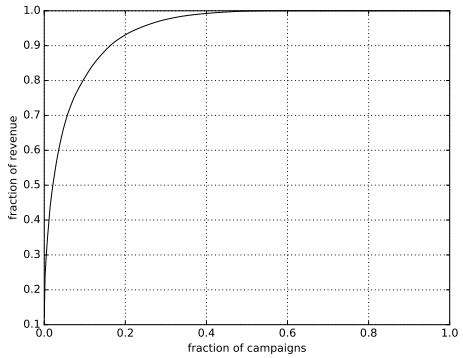


Figure 3.4 CDF of revenue per campaign.

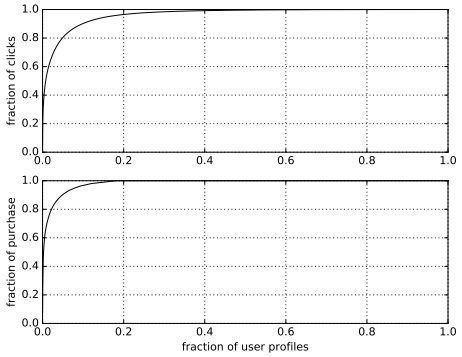


Figure 3.5 CDF of clicks/purchase per user profile.

Since the point process analysis requires a chain of purchase history, we focus on the campaigns with more purchases and revenues in the following analysis. We selected the data from top campaigns and set a threshold for the number of purchases of each user profile to be at least 100. There are 64 unique combinations of user profiles and campaigns. Figure 3.6 shows the density distribution of the time between two purchase events for each unique combination of user profile and campaign ID. The first peak in Figure 3.6 indicates two consecutive purchases from the same user profile happening within 160 minutes and the probability of having longer intervals decreases. The second peak is at about 30 hours which could be explained as the maintenance time from the administrator when some campaigns are temporarily disabled. Given the fact that the burst of purchase from the same user group

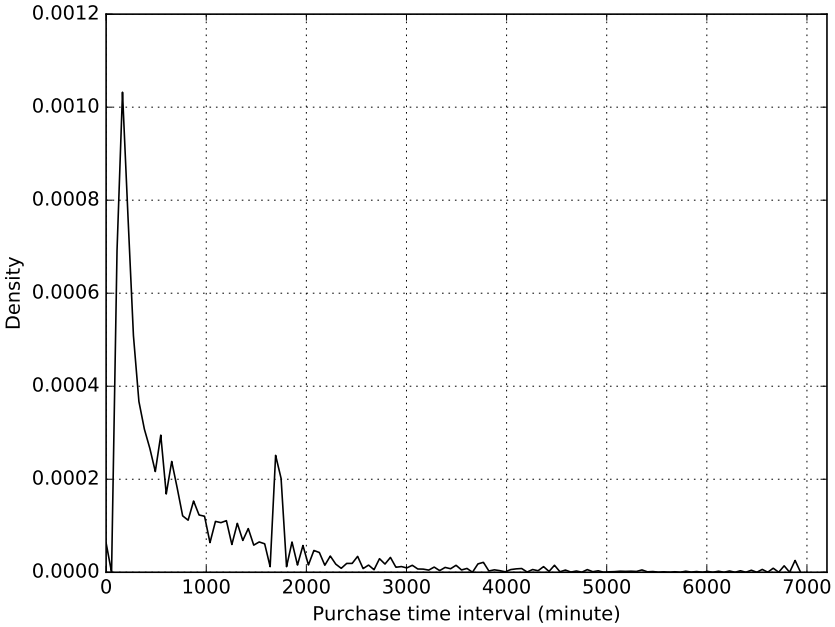


Figure 3.6 Density distribution of purchase time interval.

has high probability within 2 to 3 hours, the next step is to model the purchase process to check if it matches a self-exciting process.

The purchase events for each user profile and campaign is evaluated separately. The Akaike information criterion (AIC) is calculated for both Hawkes model (AIC_H) and homogeneous Poisson model (AIC_P), which is used for comparing the fitness of different models [8]. The AIC score is defined as $2K - 2log(L)$ where K is the number of parameters in the model and L is the maximum likelihood. The model that has the lower AIC score is fitted better. Figure 3.7 depicts an example of the conditional intensity for a particular user profile purchase history that fits the Hawkes process better. The x axis shows the minutes since the beginning of the measurement. Each new purchase triggers an increase of purchases for a short time then decreases back to the background intensity. Another example is shown in Figure 3.8 where the purchase in the past has negative effect on subsequent purchases. The intensity drops immediately after a purchase event.

The AIC score is used to compare two models which cannot individually show how good of each model is fitted to the data. We further conduct a residual analysis [16] for the data which Hawkes process fits better than a Poisson process and for the data which Poisson process fits better. For the one-dimensional temporal point process, the residual is defined as: $R(t) = N_t - \int_0^t \lambda(s)ds, s < t$, where N_t is the number of events accumulated from time 0 to t,

λ is the estimated intensity of the point process model, in our case, it is the Hawkes process, and s is a time point prior to t. The residual is the difference between the real number of events during a certain time and the approximated number of events calculated by the fitted model. Ideally, when the estimated intensity is closer to the real intensity, the residual process should be close to a homogeneous Poisson process with the rate μ estimated by the Hawkes model in Equation 3.2. Correspondingly, the inter-event times of the residual process should be exponential with mean $1/(\mu)$ [110]. Thus, the log-plot of the number of events during each inter-event time for a Poisson process should be a straight line as is shown in Figure 3.9 and Figure 3.10. The better the model fits the data, the closer the residual log-plot to be linear. In Figure 3.9, it proves that the Hawkes process fits the data better.

Based on the observation, we selected the user profiles with purchase history which can be fitted into a Hawkes process and compute the intensity for every minute since the first purchase event. If the current time is denoted as t minutes since the first purchase, the intensity of $t-1$ minutes calculated by the fitted Hawkes model is considered as an additional feature for logistic regression. The performance result is summarized in Section 3.4.

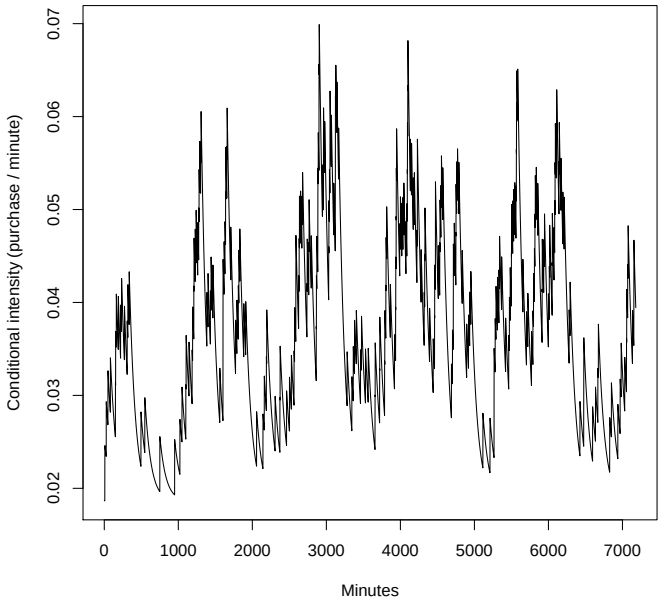


Figure 3.7 Conditional intensity of a purchase process having $AIC_H < AIC_P$.

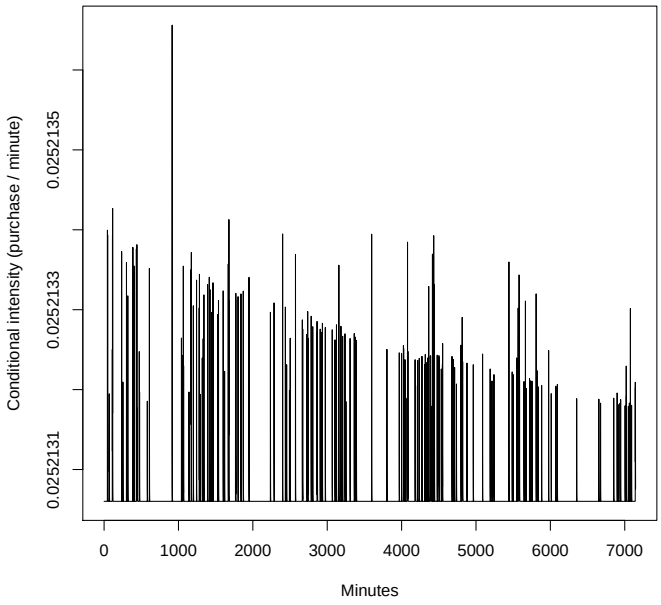


Figure 3.8 Conditional intensity of a purchase process having $AIC_H > AIC_P$.

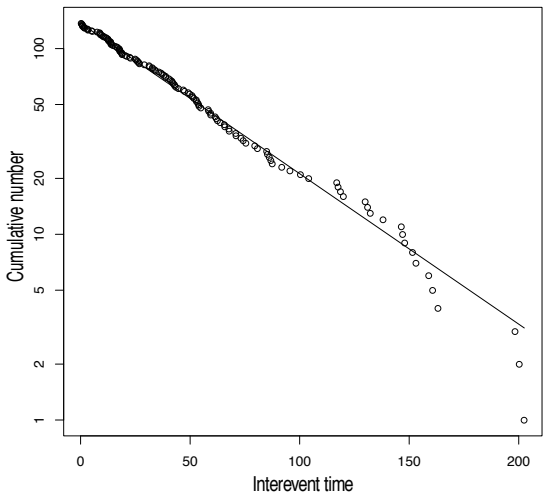


Figure 3.9 Residual analysis of a purchase process having $AIC_H < AIC_P$.

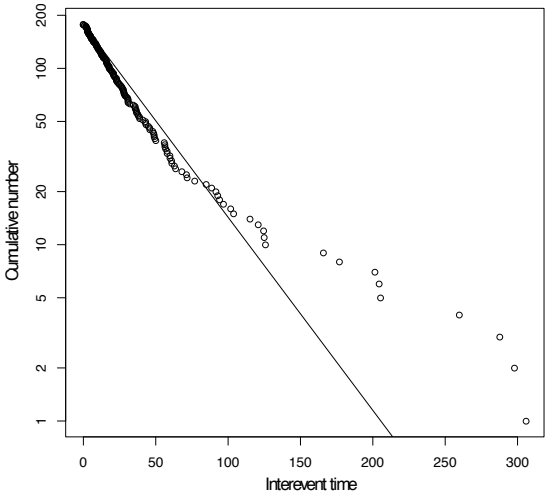


Figure 3.10 Residual analysis of a purchase process having $AIC_H > AIC_P$.

3.3.4 Models

In this section, three baseline models are tested for CVR prediction: Logistic Regression, Naive-Bayes, and Random Forest. We use X, y to represent the feature vector and true label, respectively. For each feature vector X_i, it contains k different features: $X_i =< x_{i1}, x_{i2}, ...x_{ik} >$.

- **Logistic Regression** Logistic regression has been widely used as a linear model for CTR/CVR prediction [122, 84]. It assumes linear dependencies between features: $f = \alpha_0 + \alpha_1 x_1 + ... + \alpha_k x_k$. The probability of having label y_j, given feature vector X_i is $P(y_j|X_i) = \frac{1}{1+\exp(-f)}$ and its loss function is defined as the cross entropy between the predicted probability and the true label. In addition, the L_2-regularization is added to avoid over-fitting.

- **Naive-Bayes** Different from logistic regression which directly models the conditional probability $P(y|X)$, the Naive-Bayes model [5] calculates the probability of $P(y|X)$ by estimating the joint probability $P(X,y)$ and applying the Bayes theorem. For example, given a feature vector X_i, the probability of having label Y_j is represented by $P(y_j|X_i) = \frac{P(X_i|y_j)P(y_j)}{P(X_i)} = \frac{P(X_i|y_j)P(y_j)}{\sum_j P(X_i|y_j)P(y_j)}$. Naive-Bayes is based on the assumption that features are independent which simplifies the estimation of $P(X_i|y_j)$ to be the product of marginal probabilities: $\prod_{m=1}^{k} P(X_{im}|y_j)$. However, the simple assumption adds high bias to the model.

- **Random Forest** Random forest [5] is an ensemble classifier which trains multiple decision trees in parallel and the classifiers can be combined by averaging their probabilistic predictions. Each tree is built by bootstrap sampling a subset of the features and data points. It eliminates the bias of assuming features being either linear dependent or independent as in the previous two models. Furthermore, through bootstrapping, it also reduces the risk of overfitting. The result can be aggregated by voting or averaging over the trees.

3.4 Evaluation

3.4.1 Metrics

To evaluate the performance of predictive models, the area under the ROC curve (AUC) is widely used [43]. Given the values from the confusion matrix, True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN), the sensitivity is defined as $\frac{TP}{TP+FN}$ measuring the percentage of correctly labeled positive entries over the positive class. The specificity is defined as $\frac{TN}{TN+FP}$ and the False Positive Rate (FPR) equals $1-$ specificity. For each data entry, the prediction model produces a probability that the example belongs to a class, which needs a threshold to classify it. In our dataset, the positive class represents purchases and the negative class is the clicks without any following purchase (non-purchases). For instance, the sensitivity in our case is interpreted as the percentage of correctly predicted purchase events over all the purchase events.

The ROC curve plots the sensitivity against the FPR under all possible thresholds. AUC is calculated as the area under the ROC curve which ranges from 0 to 1. In our study, since the overall conversion rate is only 0.5%, by classifying all the clicks to non-purchases, we can obtain accuracy of 99.6%, while the prediction for a purchase is 100% incorrect. However, the AUC will be 0.5 which indicates that the predictor is as bad as random guessing. Therefore, the insensitivity to imbalanced dataset is the advantage of AUC over other evaluation metrics, such as accuracy.

In Table 3.2, the AUC is computed from training on five days of data and testing on the next two days as described in Section 3.3.1. The static features in Table 3.1 serve as base line features. Naive-Bayes with the simple assumption of independent features performs the worst. In random forest, each tree selects the square root of the total number of features and we construct 20 trees for training. In this case, random forest with 20 trees performs better than the logistic regression model. However, the training time of random forest is 10 times longer than training logistic regression by using a server with 100GB RAM.

To keep the baseline prediction model to be more efficient with high dimensional features, we select logistic regression as the base line model to compare the performance of adding additional features as proposed in Section 3.2. The purchase behaviors of user profiles which can be fitted into a Hawkes process are limited. The total clicks generated from

Table 3.2 AUC Comparison between Baseline Models.

Models	AUC
Logistic Regression	0.7109
Naive Bayes	0.6362
Random Forest	0.7222

Table 3.3 AUC Improvement from Feedback Features and Temporal Features.

Features	AUC
Static	0.7397
Static + CVR flag	0.7406
Static + CVR flag + Intensity	0.7421

these user profiles are about 2 millions over 7 days. These data are chosen to first compute AUC but using only the base set of features, followed by adding the feedback feature (CVR change flag) and the temporal feature (intensity) to the logistic regression model. The result (see Table 3.3) shows the model performing better when the additional features are included. It demonstrates the importance of considering the temporal nature of the click and purchase events. At a first glance the results shown in tables 3.3 and respectively 3.2 seem contradictory, since the AUC for logistic regression without temporal features is higher than the the AUC where temporal features are leveraged. Note however that Hawkes processes will fit only a subset of the dataset used in Table 3.2, but on this subset the AUC is increased. In practice we use logistic regression with temporal features on user profiles which can be fitted to a Hawkes process, while using simple logistic regression without temporal features on the other user profiles.

3.4.2 Dealing with Different Sizes of Training and Testing Data

In the previous experiments, the training set was fixed to be 5 days of data. However in a dynamic and evolving system, it is natural to investigate which is the optimal history over which training needs to be done. Using historical data that ranges over a too distant time frame might not allow to be reactive and to rapidly learn new trends in the data, while training over shorter time periods comes with a price expressed both in required computational resources and in forgetting useful models learned in the past. We have therefore investigated in this section, the appropriate volume of training data for future use in the online system. In order to predict CVR for one hour, the model is trained using historical data with window

sizes k (1, 4, 6, 8, 12 hours) and tested on new data from the next hour. The sliding window is run over data covering 6 days. In this section, the whole dataset with 17M entries is used. The feature set consists of the static features and the feedback feature.

In Figure 3.11, from the top to the bottom, we report the AUC for each hourly interval when using window sizes of 1, 4, 6, 8, and 12 hours respectively. The predicting model is updated every hour beginning with the start of the measurement. The AUC is relatively low in the top plot comparing to others.

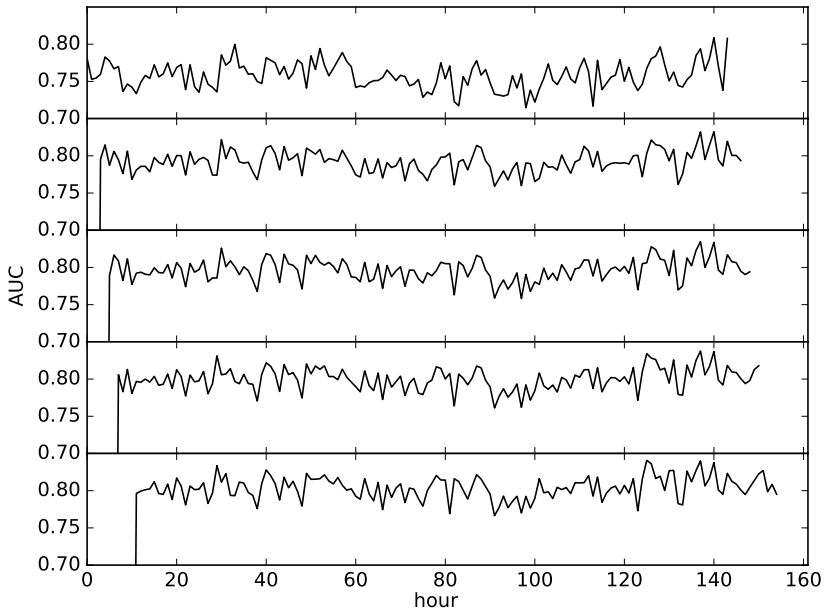


Figure 3.11 AUC changes over each hour with window size of [1, 4, 6, 8, 12] hours.

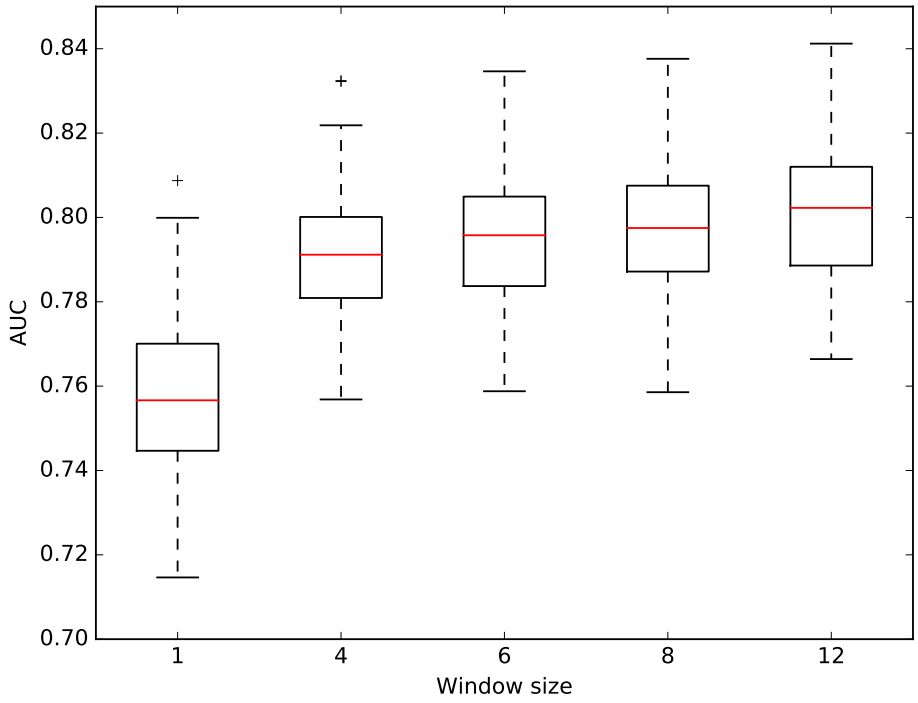


Figure 3.12 Boxplot of AUC for different training window size.

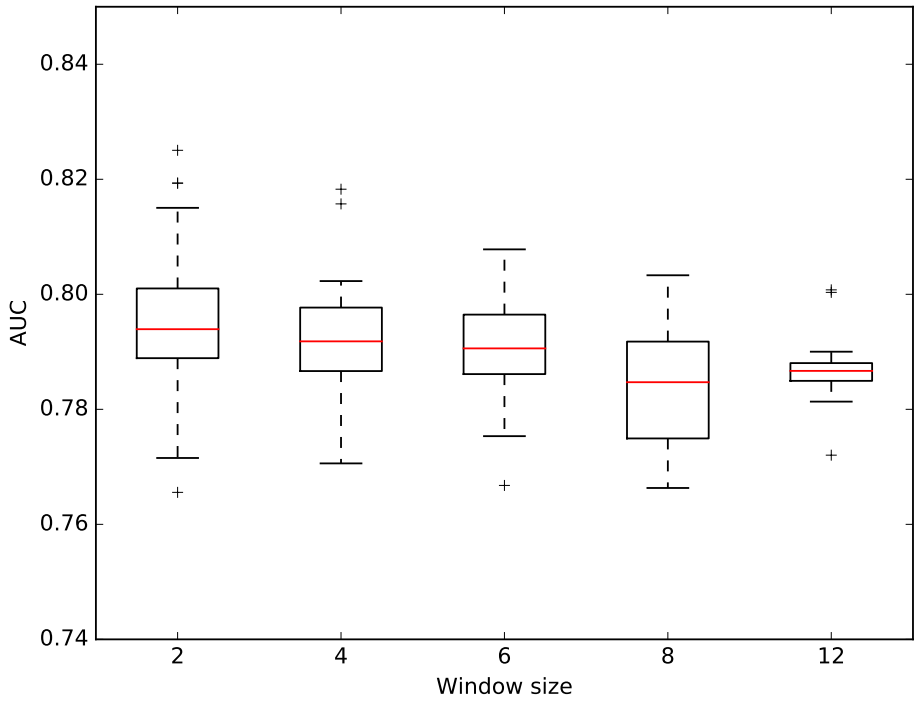


Figure 3.13 Boxplot of AUC for different test window size.

In Figure 3.12, the boxplots for the AUC over 144 hours (6 days) are shown for different window sizes. The central line marks the median and the lower and upper hinges define the 25th ($Q1$) and 75th ($Q3$) percentile. The length of the box ($Q3 - Q1$) is defined as interquartile range (IQR). The whiskers range from $Q1 - 1.5IQR$ to $Q3 + 1.5IQR$. The crosses outside the boxes are outliers. By having 4 hour training data, the median of AUC can be improved by more than 3% (from less than 0.76 to 0.79). This is mainly due to the sparse purchase data. The overall CVR in our dataset is only 0.5% and as is shown in Figure 3.3, the hourly CVR of the relatively active campaigns is still less than 1%. One hour is too short to collect a sufficient volume of purchase data for the model to learn from. The AUC increases become marginal if the window size is larger than 4. Using 4 to 8 hours data for training is reasonable in this case.

We further investigate how often the training model needs to be updated before the performance drops. The training window size is chosen to be 6 hours and the model is tested on window sizes ranging from 2, 4, 6, 8 and up to 12 hours. As can be seen in Figure 3.13, a noticeable AUC drop starts when the test window size is larger than the training window size. Even if it is only a drop of 1%, it can cause significant revenue loss if the model underestimates the conversion rate for more profitable campaigns or for the users with higher potential to purchase.

3.5 Summary and Future Work

In this chapter, we have addressed the prediction of user purchases in a mobile advertising context. For this purpose, we propose a new approach that leverages a mix of static and temporal features relating to a user profile and a campaign. Our model groups individual users sharing common features within one user profile and provides predictive solutions for specific campaign related purchases. We have shown how additional and global performance metrics can be used to generate signals that capture the short term trends and identified how these signals can be used for predictive tasks. We have validated our approach on large real world data obtained from a major actor in this industry, covering more than 200 countries and few hundred campaigns. We have evaluated several supervised classification methods and identified their relative strengths and limits for this purpose. We have also investigated the time granularity at which retraining is required in order to capture the inherent dynamics and behavioral shifts occurring in the advertising markets. Our solution requires pairwise

modeling of user-profile and campaign level temporal modeling. For the users without any purchase history, the proposed model cannot be applied directly. In this case, it requires to collect other relevant information of the users, for instance their browsing behavior and mobile app usage. With more informative profile, similar users can be classified which provides a more generalized estimation of their purchase intentions.

Chapter 4

Market Price Modelling in Real-Time Bidding

4.1 Introduction

In the RTB environment, from the advertiser's perspective, the DSPs aim to bid optimally so that within a limited budget constraint, they can win more high value impressions and maximize the total revenue. Many research works have addressed the bidding strategies optimization problem in RTB [160, 111, 41].

In the bidding optimization process, the competition in the market decides the cost of winning an impression and therefore has an impact on the budget consumption and the overall ROI metrics such as the CPC, CPM, and CPA. The underlying dynamics of the bidding market can be mainly attributed to various bidding strategies and the budget status of other bidders. In addition, in every auction, the identity and the number of participating bidders are unknown and varies from one auction to another. Furthermore, the competitions are not just among advertisers. On the publishers' side, to ensure the ad slots are sold over the minimum price and to maximize their revenue, the publishers adjust their pricing models for setting a dynamic floor price which may raise the paying price for the advertisers [116]. As is shown in Figure 4.1, if the highest bid is lower than the hard floor price, the slot remains unsold. If the highest bid surpasses the hard floor price but is lower than the soft floor price, the auction becomes a first price auction since the winner is obliged to pay its own bid price. Once the

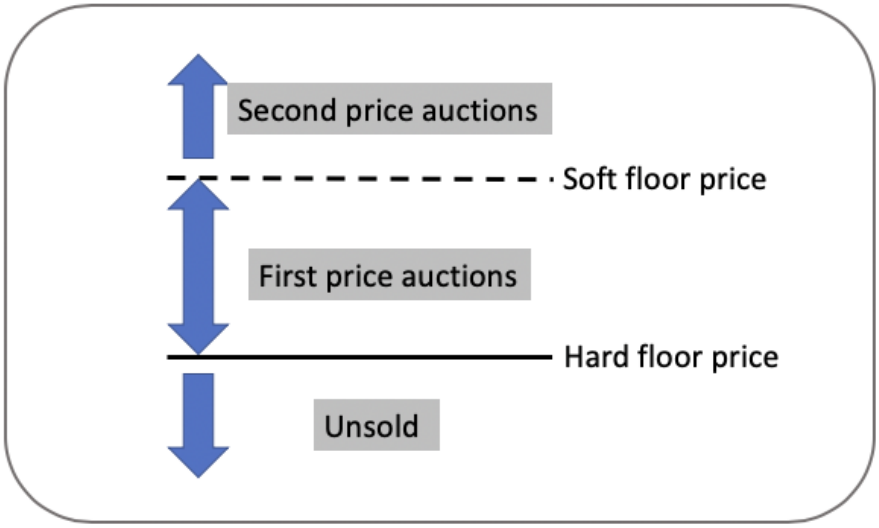


Figure 4.1 Floor prices set by publishers in the RTB market.

highest bid is beyond the soft floor price, the auction is a second price auction. Therefore, the publisher side also introduces more uncertainty into the bidding environment. In this chapter, we focus on predicting the market price in the second price auctions.

In the RTB system, each ad campaign can receive over hundreds or thousands of bid requests per second. The market price for these auctions can fluctuate significantly even within milliseconds, because of the competition changes in the bidding environment. Due to the high fluctuations in the market price, predicting the winning price for each individual auction is a very challenging task.

In this dissertation, we provide two approaches for estimating the market price. The first model provides a *price level prediction* and the second model takes the market price as a stochastic variable and predicts its distribution which is also called the *bid landscape prediction*. The *price level prediction* model formulates the market price as a time series and captures the temporal patterns in the price domain. However, the price level prediction lacks the generality of estimating the winning probability for any given bid prices. In the bidding optimization process, the winning probability affects both the budget consumption and the utility acquisition (such as CTR or CVR estimation). Since the click or the conversion is only possible to be observed after winning the auction, *bid landscape prediction* model maps the bid request features to the market price distribution and facilitates the winning probability estimation at an ad impression level.

4.2 Price Level Prediction

The market price for a single auction fluctuates heavily over time. We propose to aggregate the bid requests by time in order to easily formulate a time series and to observe the temporal patterns. Figure 4.2 shows an example of the number of winning impressions per second of an ad campaign in the iPinYou dataset[1]. Within one second, the winning auctions are over hundreds which is roughly 20% of the total participating auctions. Given the high fluctuation in the traffic volume per second, to keep the number of auctions in each aggregation segment constant, we aggregate winning auctions by a fixed window size such as 100, 500, and 1000 auctions. The average market price ($\overline{m_t}$) of each window is modeled as a time series. We forecast the average market price 1-step ahead with a LSTM model which captures the non-linear dependencies in the price domain over time. This is based on the assumption that, in RTB, each bidder calculates a bid price based on each received bid request. Although for each bidder, the winning probability and the utilities (CTR or CVR) are usually different and remains unknown to each other, one can still take the request features as the external source for referring to its impact on the other bidder's decisions. Therefore, we demonstrate that the fluctuation of the market price can be implicitly captured by exogenous variables such as the request features.

4.2.1 Check Stationarity of the Time Series Data

Choosing a proper prediction model depends on the underlying structure of the time series. For instance, it is important to know whether the time series is stationary or time dependent. The Augmented Dickey-Fuller (ADF) test is widely used for trend detection in a time series [53]. The more negative the value is, the more strongly that the time series is stationary. In addition, we also show the p-value from the ADF test. When p-value equals to or is greater than 0.05, it rejects the null hypothesis that the time series is stationary. Otherwise, the time series is stationary. In Table 4.1, the p-value and the ADF score are listed for different window sizes. The ADF test results suggest that for all campaigns except campaign 2259, the time series of the average market price per window is stationary.

[1] https://contest.ipinyou.com/

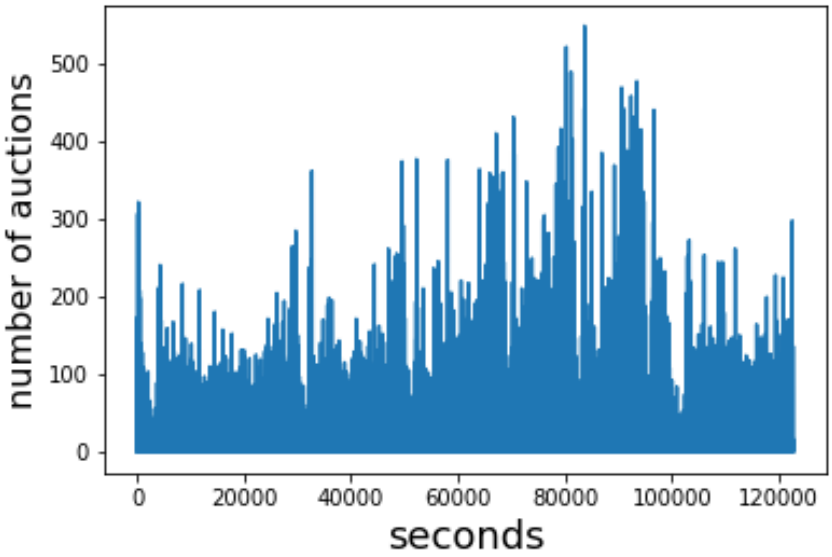


Figure 4.2 Example of the number of winning auctions per second from an ad campaign with ID 3476 in the iPinYou dataset.

4.2.2 Baseline: Autoregressive Model

Given the stationarity of the time series, the Autoregressive model serves as a reasonable baseline to forecast the mean market price per window. An $AR(p)$ model is used to model the linear correlation between the previous p values and the current value y_t. The model is denoted as:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + w_t \tag{4.1}$$

where c is a constant, ϕ_i is the model covariant, p denotes the autoregressive term, and w_t is the white noise, which can be ignored for simplicity.

For non-stationary campaigns, we adopt a more generalized ARIMA model. As the name suggested, an ARIMA model consists of three parts and is specified by the autoregressive order p, the non-seasonal difference order d, and the order of prediction error q.

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} - \theta_1 \varepsilon_{t-1} - ... - \theta_q \varepsilon_{t-q} \tag{4.2}$$

ε is the prediction error at each time step. For example, if parameter $d = 1$, the ARIMA(1,1,0) model is represented as:

$$y_t = c + y_{t-1} + \phi_1 (y_{t-1} - y_{t-2}) \tag{4.3}$$

Table 4.1 Stationarity Test.

camp.	window size					
	100		500		1000	
	p value	ADF	p value	ADF	p value	ADF
3358	6.50e-11	-7.42	5.79e-5	-4.78	0.0007	-4.17
3386	0	-22.17	3.79e-18	-10.28	2.972e-16	-9.52
3427	0	-22.25	8.42e-29	-15.89	4.188e-29	-16.18
3476	3.51e-30	-17.70	2.68e-13	-8.36	1.49e-11	-7.68
1458	0	-23.62	3.13e-15	-9.12	1.31e-07	-6.04
2259	**0.04**	**-2.90**	**0.06**	**-2.78**	**0.09**	**-2.58**
2261	0.001	-3.97	0.006	-3.57	0.002	-3.82
2821	1.25e-6	-5.60	0.001	-4.01	0.005	-3.59
2997	0.004	-3.65	0.004	-3.66	0.009	-3.43

To determine the (p, d, q) value, we fit the ARIMA model with the time series data and different settings of (p, d, q). The AIC score is chosen as the metric to select the best (p, d, q) set. Lower AIC scores suggest better fitting of the model.

4.2.3 Recurrent Neural Networks

The recurrent neural network [48] is designed for learning the non-linear dependencies in sequential data. As a variant of RNN, the LSTM model [59] extends the ability of selective recalling useful information in a long sequence. Figure 4.3[2] depicts one LSTM cell at time step t which contains two hidden states c and h and three gates, namely forget gate f_t, input gate i_t, and output gate o_t, to control the information which should be memorized and passed on in a sequence. In each LSTM cell, the cell state c_t consists of the state from the previous cell and the information from the current cell. The cell state is calculated as follows:

$$\tau_* = \sigma(W_*[h_{t-1}, x_t] + b_\star) \tag{4.4}$$

$$\hat{c}_t = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{4.5}$$

$$c_t = \tau_i \cdot \hat{c}_t + \tau_f \cdot c_{t-1} \tag{4.6}$$

[2]https://colah.github.io/posts/2015-08-Understanding-LSTMs/

where $[h_{t-1}, x_t]$ is the concatenation of the hidden variable h and the input vector x and τ_* represents the shared sigmoid function form between the three gates. Therefore, in Figure 4.3, at time step t, $f_t = \tau_f$ and $i_t = \tau_i$. The gate f_t determines how much historical information (c_{t-1}) from the previous cells are memorized and i_t represents how much new information $(\hat{c}_t)$ from the current cell should be passed along. The output $\hat{y}_t$ is calculated as:

$$h_t = \tau_o \cdot tanh(c_t) \tag{4.7}$$

$$\hat{y}_t = g_t(W_y h_t + b_y) \tag{4.8}$$

where $\tau_o = o_t$. It is the result of the output gate and decides how much information from the current cell is used for the prediction task. The final prediction is the output of an activation function g_t. In our case, since the output is a numerical value as the average market price, the output layer is a linear dense layer.

The structure of the LSTM models are shown in Figure 4.4. The inputs $y_1, ..., y_{t-1}$ are the historical market price, h_0 is the initialization vectors with zeros.

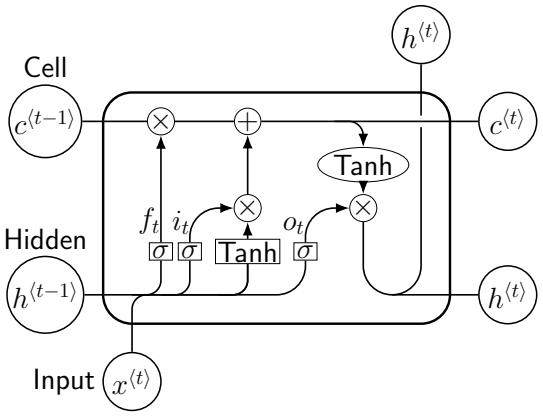
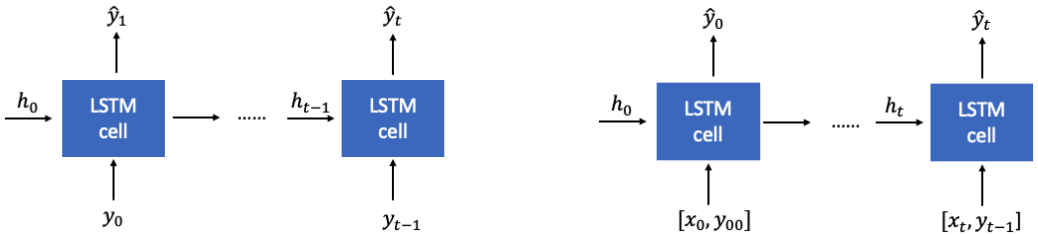


Figure 4.3 One LSTM cell.

4.2.4 Integration with Exogenous Features

In the dynamic RTB environment, the winning price is determined by many exogenous factors, such as the user and publisher's profiles in the bid requests and the strategies used by unknown sets of bidders participating in the auction. In theory, bidders bid at their optimal price according to the Bayesian-Nash equilibrium in the auction theory [76]. When



(a) LSTM Baseline.

(b) LSTM(X) model.

Figure 4.4 The proposed LSTM structure.

making bidding decisions, each bidder combines the information in the bid request, the ad details, together with their budget and sets their best bid price. Therefore, in addition to formalizing the market price as an univariate time series, we further investigate how the additional information in the bid requests contribute to the price prediction.

As mentioned at the beginning of this section, we aggregate different numbers of winning auctions into a window and take the averaged market price. Here we summarize the features from each request into one vector per window to describe the heterogeneous environment. For each feature, we calculate a window-based histogram of the feature values. When some values are not observed, they are considered to be zero. The histograms of each feature are concatenated into one vector. In this case, we ensure that the dimension of the input vector is the total number of unique feature values and remains the same:

$$x_t = [\underbrace{0.1, ..., 0.2, ...0}_{city, \sum=1}, \underbrace{0.2, .., 0.4}_{browser, \sum=1}, ...]$$

Besides the features from the bid request, we concatenate the market price prior to the prediction time step as input feature. At the beginning of the time series, y_{00} is the padded zeros to keep the input length identical to all other input vectors. The combined input is fed into a LSTM layer and the target value is the averaged market price y_t. We denote the model as LSTM(X) which is shown in Figure 4.4b.

$$x'_t = [\underbrace{0.1, ..., 0.2, ...0}_{city, \sum=1}, \underbrace{0.2, .., 0.4}_{browser, \sum=1}, ..., y_{t-1}]$$

4.2.5 Dataset

In our experiments, we use a public real-world RTB dataset. It was released by iPinYou, a leading RTB company in China. The dataset contains 19.5M impressions, 15k clicks and 1.2k conversions over 9 ad campaigns [161]. The iPinYou dataset was collected during 10 days in 2013. The first 7 days are used as training data and the last 3 days are the test data. Each bid request contains features including the profiles of the user, the publisher, and the ad slot description. The user profile includes *timestamp, user agent, IP address, region, city*, and *user tags*. The publisher is represented by *domain* and *url* and the ad slot is described by *slot ID, width, height, visibility, format, advertiser ID*, and *creative ID*. The visibility describes if the current ad slot is the first view or not. The format represents if the ad is a fixed slot, a pop-up window, or other formats. A more detailed statistical description of the dataset can be found in [161].

In this study, we exclude the features with high variety such as IP address and URL. An example of the bid log is shown in Table 4.4. We simplify the *time stamp* feature as *the day of the week* and the *hour*. The *user agent* feature is divided into the *operating system* and the *browser*. In this way, all the features have become categorical features. In addition, the bid log records the original bid price during the data collection and the historical paying price. The click column represents if the user clicks on the ad or not. The statistics of the training and the test datasets are summarized in Table 4.2 and Table 4.3 respectively.

Table 4.2 iPinYou Dataset Statistics **Training** Data.

Campaign ID	Time	Impression	Clicks	Cost	Win Ratio	CTR
1458	6-12 Jun.	3,083,056	2,454	212,400	20.97%	0.08%
2259	19-22 Oct.	835,556	280	77,754	27.97%	0.034%
2261	24-27 Oct.	687,617	207	61,610	31.84%	0.03%
2821	21-23 Oct.	1,322,561	843	118,082	24.99%	0.064%
2997	23-26 Oct.	312,437	1,386	19,689	30.69%	0.444%
3358	6-12 Jun.	1,742,104	1,358	160,943	46.44%	0.078%
3386	6-12 Jun.	2,847,802	2,076	219,066	20.21%	0.073%
3427	6-12 Jun.	2,593,765	1,926	210,239	18.48%	0.074%
3476	6-12 Jun.	1,970,360	1,027	156,088	29.35%	0.052%

Table 4.3 iPinYou Dataset Statistics **Test** Data.

Campaign ID	Time	Impression	Clicks	Cost	CTR
1458	13-15 Jun.	614,638	543	45,216	0.088%
2259	22-25 Oct.	417,197	131	43,497	0.031%
2261	27-28 Oct.	343,862	97	28,795	0.028%
2821	23-26 Oct.	661,964	394	68,257	0.060%
2997	26-27 Oct.	156,063	533	8,617	0.342%
3358	13-15 Jun.	300,928	339	34,159	0.113%
3386	13-15 Jun.	545,421	496	45,715	0.091%
3427	13-15 Jun.	536,795	395	46,356	0.074%
3476	13-15 Jun.	523,848	302	43,627	0.058%

Table 4.4 iPinYou Bid Log Example.

Feature	Example
Time stamp	20130606000104800
User agent	windows_ie
IP*	115.45.195.*
Region	216
City	219
User Tags	10006,1011
Ad exchange	1
Domain	trqRTJkrBoq7JsNr5SqfNX
URL*	f41292b3547399af082eccc2ad28f23c
Ad slot ID	mm_34022157_3445226_11175096
Slot width	336
Slot height	280
Slot visibility	2
Slot format	1
Creative ID	77819d3e0b3467fe5c7b16d68ad923a1
Advertiser ID	1458
Bidding price	300
Paying price	51
Click	0

4.2.6 Results

In this section, we present the corresponding results of the ARIMA, LSTM, and LSTM(X) models. The prediction of the mean market price per window from the ARIMA model is listed in Table 4.5 with the evaluation metric Mean Squared Error (MSE). For comparison, the results of the LSTM model are shown as a percentage of improvements, relative to the results of the ARIMA model in Table 4.6. Positive number represent a decrease of the MSE and vice versa. We take market prices from 10 previous steps as a sequence to predict the price for the next time step. Using more history, for example 20 or 50 time steps, does not improve the results any further. Since the environment is highly dynamic, using very long sequence results in taking the price from further past in the temporal space.

Our results show that in most cases, the LSTM model improves the prediction results significantly. It demonstrates the benefits of memorizing longer histories for the prediction. However, there are also performance drops at different aggregation levels in a few campaigns, e.g. 2821 and 2997. We further checked the number of auctions per second for these campaigns. It shows that the pattern of request arrival rate increases or decreases dramatically from training set to test set. In our settings, the window size is fixed. Therefore, with larger window size, the time span in one window varies in this case. The time series model learned with the training set fails to capture the different pattern in the test set. For the other campaigns, the amount of bid requests per second over time remains relatively stable across training and test data. Given our observations, it appears that there is a need to predict the traffic pattern as well. In Figure 4.5, the true average market price per window and the predicted value are presented.

Furthermore, we demonstrate how the features in the bid requests contribute to the market price prediction as described in Section 4.2.4. Table 4.7 compares the MSE between the results of LSTM taking only historical market price as input and the LSTM model with features in the bid request, LSTM (X). The results clearly shows the improvements with additional features. However, campaigns 2261 and 2997 have negative results, meaning including additional features increased prediction error. As shown in Figure 4.5, the mean market price per window in these two campaigns shows very little fluctuation, which suggests a relatively steady environment. In this case, adding additional information from the bid request may perturb the model and jeopardize the prediction accuracy. On the contrary, for campaigns with high fluctuations in the market price, the summary of the requests manages to capture the sudden changes in the market.

Table 4.5 MSE of ARIMA model.

campaign	window size		
	100	500	1000
	MSE	MSE	MSE
3358	159.75	211.07	279.11
3386	158.14	214.14	294.33
3427	176.05	235.17	263.70
3476	190.06	231.96	227.09
1458	153.49	158.94	129.44
2259	185.69	38.86	32.36
2261	174.88	43.10	27.66
2821	346.95	38.92	27.79
2997	76.86	20.35	13.66

Table 4.6 Comparing ARIMA and LSTM.

campaign	window size		
	100	500	1000
	Improves over ARIMA (%)		
3358	-10.28	2.17	5.16
3386	7.02	5.42	13.12
3427	-1.64	4.56	14.02
3476	11.92	36.48	44.12
1458	13.26	24.10	20.56
2259	59.18	7.89	12.06
2261	64.64	37.09	21.73
2821	22.34	-29.43	-42.02
2997	63.26	-3.65	-28.26

Table 4.7 Comparing LSTM and LSTM(X).

campaign	window size		
	100	500	1000
	Improves over LSTM (%)		
3358	-0.45	27.20	17.79
3386	10.01	63.17	144.01
3427	6.55	18.28	32.29
3476	17.69	76.76	66.79
1458	0.52	9.41	8.99
2259	17.86	34.24	-36.87
2261	-36.89	-21.13	-54.39
2821	83.41	87.79	46.21
2997	-22.65	-40.08	-50.68

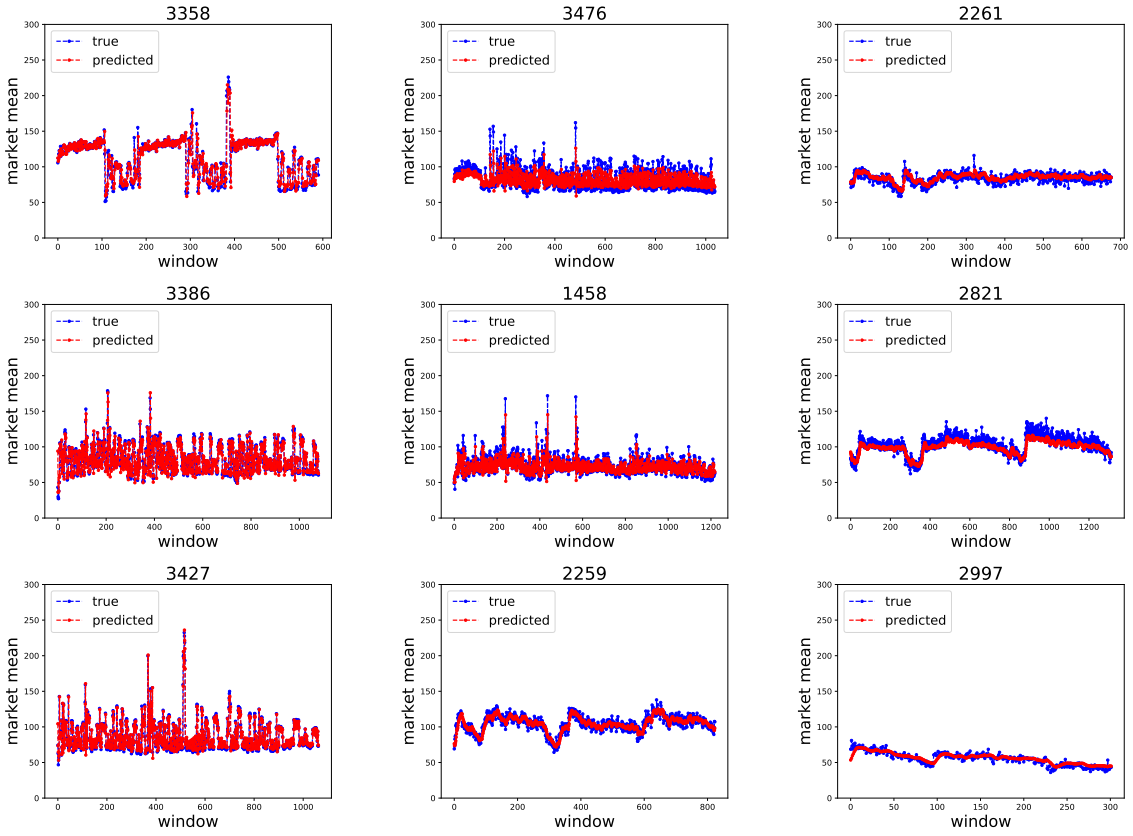


Figure 4.5 Market price prediction by LSTM model with window size 500.

4.2.7 Discussion

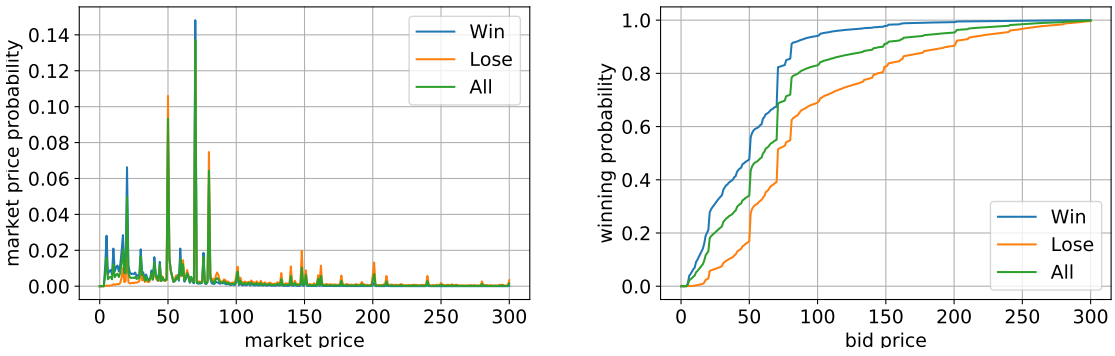
Our result shows the ARIMA model is a very strong baseline due to the strong stationariy in the time series of the averaged market price. As mentioned in Section 4.2.1, the ARIMA model requires the time series to be stationary and is based on linear correlations. Otherwise,

extra data transformation is needed to convert the time series to be stationary or to choose other models such as seasonal ARIMA model [19]. However, in practice, the time series may not always be stationary and may not be linearly correlated. Even in our dataset, the LSTM model does not always outperform the ARIMA model for the stationary time series. Comparing with the strict requirements of the ARIMA model, for more complex time series, the LSTM model provides more generalized support for different types of time series analysis without the need of stationary check.

4.3 Impression Level Market Price Distribution Prediction

4.3.1 Background and Motivations

The approach described in the previous section takes market price only from the winning impressions for training a global model. As discussed earlier, the winning price of the lost auctions are right censored. Figure 4.6a shows an example of the market price distribution of all the data. In addition, it shows the market price distributions of winning impressions and losing auctions separately. The winning and losing auctions are the results of a bidding simulation where the budget is set to be half of the total cost in the training data and the bid prices are generated by a linear bidding function [96]. The simulated bid prices are compared with the historical winning prices in the bid log to decide the win and the lose sets. From the price distribution, one can easily derive the winning probability plot as in Figure 4.6b. It shows, for instance, when the bid price is 100, the winning probability is over 90% estimated from the winning auctions only (the blue curve). If we take the real market price of the losing auctions into account, the true winning probability is around 80% (the green curve) which is lower than the previous result. Therefore, modeling the market only from the winning auctions results in a bias which overestimates the winning probability. Such limitations can be found in previous research works [160, 23].

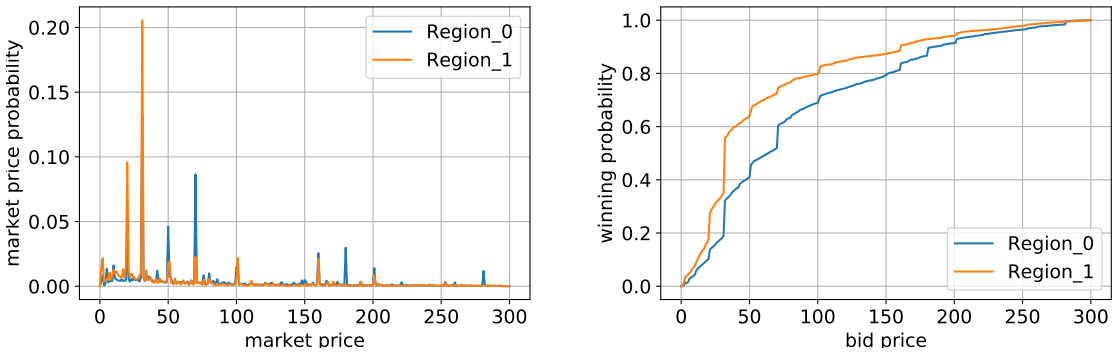


(a) Market price distribution over win / lose data. (b) Winning probability over win / lose data.

Figure 4.6 An example of biased market price and winning probabilities estimation.

In addition, a price based market prediction model cannot provide the winning probability for any given bid price. Since it predicts a real value as the estimated market price, the winning probability for an arbitrary price is either zero (when the bid price is lower than the estimated market price) or one (when the bid price is higher than the estimated market price). Therefore, the price based model is not directly applicable to estimate the winning rate for different bid prices. With the above approach of estimating the market price distribution, one global model fails to describe the divergence within the data. The market price distribution usually consists of implicit mixtures of several sub-distributions. Figure 4.7 shows an example of market distributions and the corresponding winning probabilities for the data collected from two regions. Clearly, the price distributions of the two groups are diverse from each other. With the same bid price 50, the winning probabilities can have about 20% difference. To address the divergence sub-groups in the data, previous works [142, 37] built decision tree based models to split user features into multiple clusters. The splitting criteria of each node is to maximize the KL-Divergence between the inter-cluster market price distributions. However, hyper-parameters like depth of the tree, number of leaf nodes, and leaf sizes are required to be tuned. In addition, in these studies, new input features which are not in the tree will randomly choose a node to follow. Consequently, when the number of new features increases, the prediction of the market price distribution becomes inaccurate.

Powered by deep neural networks, previous works [118, 117] propose a recurrent based Deep Landscape Forecasting (DLF) model to explicitly consider the sequential patterns in the user features space and meanwhile provides an unbiased market price distribution estimation through a survival model. Inspired by these research work, we follow their loss functions and



(a) Market price distributions of two regions.

(b) Winning probabilities of two regions.

Figure 4.7 An example of market price and winning probabilities of two groups from different regions.

release the sequential assumption in the user features by introducing a Transformer model-based approach to handle the user features in a more general and scalable way. The proposed model directly maps the user features into the impression-level market price distributions and outperforms the state-of-the-art models.

4.3.2 Preliminaries

Survival Analysis

Survival analysis is commonly used for estimating the time of occurrence of the target event which is also called *time-to-event* analysis [140]. The problem is described by the observed features of the subject, the time elapsed since the start of the measurement, and a label to indicate if the target event has happened. For instance, in Figure 4.8, on the y axis, each line represents an observed entity (e.g. a machine, a patient, a stock etc.). If the target event, for example, a breakdown of the machine, the death of a patient, or a sale price of the stock is observed, the event is marked as a cross. Otherwise, the dot shows that at the end of the observation period, the target event is not observed, which also means the survival of the entity. In this case, the target event may happen beyond the observation period, which is called the right-censorship problem. The survival analysis provides the prediction of the occurrence probability of the target event at any given time.

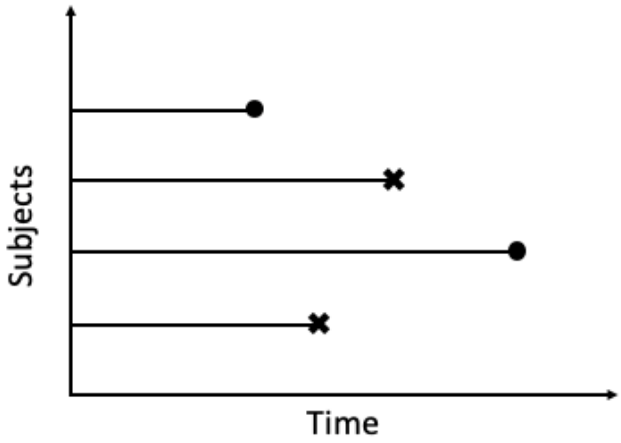


Figure 4.8 An example of the right censored problem.

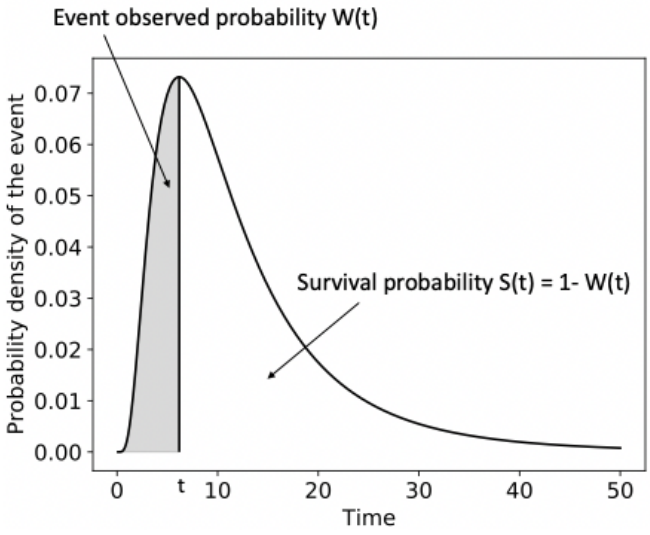


Figure 4.9 An example of survival probability.

We define T as the true time of the event occurs, thus $p(T)$ means the probability that the event happens at time T. As is shown in Figure 4.9, the probability that the event happens before time t is denoted as

$$W(t) = p(T \leqslant t) = \int_0^t p(T)dT \tag{4.9}$$

for continuous time variables. Correspondingly, the survival rate at time t is defined as

$$S(t) = p(T > t) = 1 - W(t) = \int_t^{\infty} p(T)dT \tag{4.10}$$

In the survival analysis, hazard function $h(t)$ describes the instantaneous event rate at time t given that the event has not been observed before.

$$h(t) = \lim_{\Delta t \to \infty} \frac{p(t \leq T < t + \Delta t | T \geq t)}{\Delta t} = \lim_{\Delta t \to \infty} \frac{W(t + \Delta t) - W(t)}{\Delta t \cdot p(T \geq t)} \tag{4.11}$$

From Equation 4.10, the hazard function can also be written as:

$$h(t) = -\frac{dS(t)}{dt} \cdot \frac{1}{S(t)} \tag{4.12}$$

Therefore, the hazard function can be used as an estimator to approximate the survival rate which is known as the Cox model [36]. It is based on the assumption of an exponential relation between the covariates and the hazard rate. Many variations of the Cox model have been applied on the censored data to study the risk factors in cancer diagnosis [75], survival time [21, 82], and personalized screening time for different diseases [6].

For the discrete variable applications like the market price prediction, the time variable can be uniformly discretized into K intervals, where $V_k = (t_k, t_{k+1})$ is the k-th interval. The hazard function can be represented as:

$$h_k = p(T \in V_k | T \geq t_{k-1}) = \frac{p(T \in V_k)}{p(T \geq t_{k-1})} \tag{4.13}$$

Correspondingly, the survival rate and the event rate at time t_k can be represented as:

$$\begin{aligned} S(t_k) &= p(T > t_k) \\ &= p(T \notin V_1, T \notin V_2, ..., T \notin V_k) \\ &= p(T \notin V_1) \cdot p(T \notin V_2 | T \notin V_1) ... \\ &\cdot p(T \notin V_k | T \notin V_1, ..., T \notin V_{k-1}) \\ &= \prod_{i:i \leq k} [1 - p(T \in V_i | T > t_{i-1})] \\ &= \prod_{i:i \leq k} (1 - h_i) \end{aligned} \tag{4.14}$$

$$W(t_k) = p(t_k \geq T) = 1 - \prod_{i:i \leq k} (1 - h_i) \tag{4.15}$$

Here k is the index of the interval. In the context of the market price distribution prediction, the event time T can be seen as the market price z and the observation time t_k is the bid price b_k. We use $W(b_k)$ as shown in Equation 4.15 to represent the winning probability of price b_k. Since in our experiments, the bid price is always an integer, the length of each price interval V_k is set to be 1. The probability of observing the market price $z \in V_k$ in Equation 4.16 can be derived from Equation 4.13 and Equation 4.14.

$$\begin{aligned} p(z \in V_k) &= p(z \in V_k | z \geq V_{k-1}) \cdot p(z \geq V_{k-1}) \\ &= h_k \prod_{i:i<k} (1 - h_i) \end{aligned} \tag{4.16}$$

As mentioned in Section 4.3, previous works have used RNN to model the sequential dependencies between the hazard rates. The Deep Landscape Forcasting (DLF) model proposed in [118] is shown in Figure 4.10. In this architecture, the hidden vector r_i contains the information from the previous steps and propagate it over each time step. Each prediction h_i depends on the calculations at all the previous steps which means the predictions do not support parallelization and the time complexity increases linearly with the length of the sequence. In addition, the DLF model treats all the features in the input vector $\mathbf{x}$ equally which neglects the different impacts they have on the hazard rate. For instance, a bid request generated during peak hours and a request from a specific region that the advertiser targets may have the same winning price z. Instead of recursively calculating the hazard rate from

bid price 1 to z for both cases, it would be beneficial to directly map the features to the hazard rate h_z.

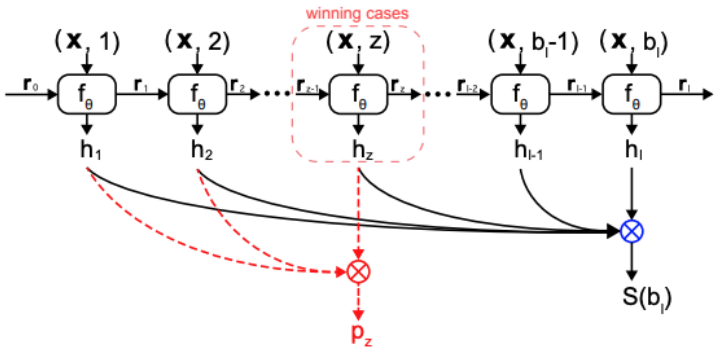


Figure 4.10 The DLF model from [118] where $\mathbf{x}$ is the input feature vector, h is the hazard rate, z is the true market price, and b is the bid price.

The Transformer model [136] has been widely applied in the natural language processing domain for sequence to sequence translations without any recurrent structure. The original Transformer model contains both encoder and decoder parts. In our problem, given the censorship in the data, the true market price is not always observable. Therefore, we only adopt the encoding part of the Transformer model as is shown in Figure 4.11[3]. The core idea of the transformer model is the self-attentive multi-headed attention layer. To encode a feature, the multi-headed Attention layer correlates each feature with all the other features from multiple latent spaces. As is shown in Figure 4.12[3], three matrices Q, K, and V are generated by matrix multiplication between the inputs and three randomly initialized weight matrices W^Q, W^K, and W^V. The reduce-sized Q, K, and V stand for: *Query*, *Key*, and *Value*. For each attention head i, the representation of Z_i is given by:

$$Z_i = softmax(\frac{Q_i \times K_i^T}{\sqrt{d_{K_i}}}) \times V_i \tag{4.17}$$

The final representation is given by multiplying the concatenation of the encoded Z_i from each head and a weight matrix W^o. In practice, multiple encoders can be stacked for getting larger model capacity.

[3]Inspired by http://jalammar.github.io/illustrated-transformer/

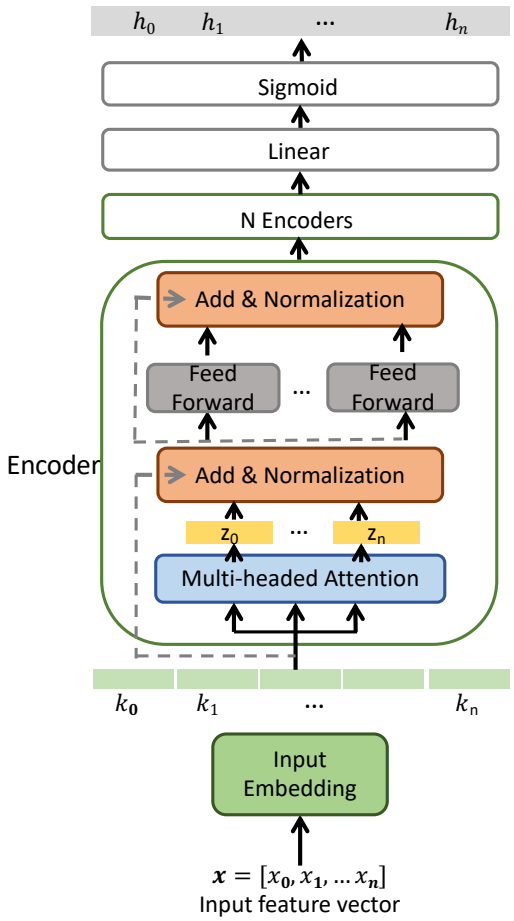


Figure 4.11 The Deep Attentive Survival Analysis model (DASA) illustration.

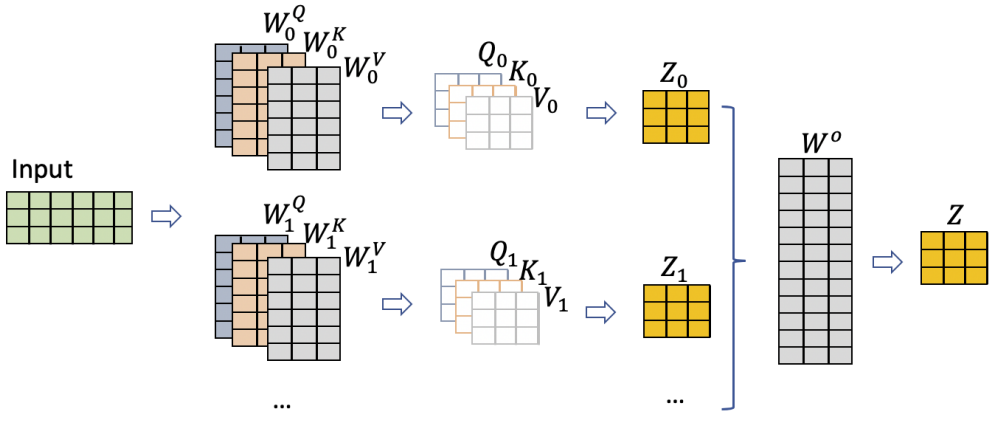


Figure 4.12 The Multi-headed Attention layer illustration.

Embedding

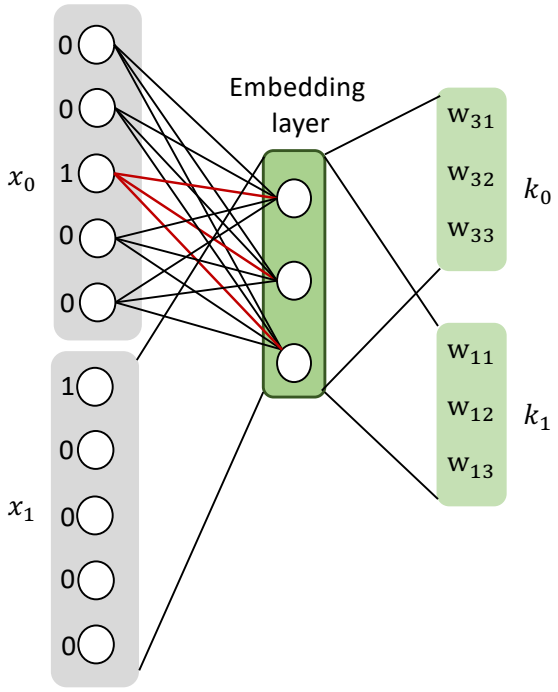


Figure 4.13 Embedding Layer.

Before feeding the input vectors to the transformer model, an additional embedding layer is added. As introduced in Section 4.2.5, the bid requests contain only categorical features. A popular way to transform the categorical features into numerical features is through One-hot-encoding [158, 122]. The dimensionality of the one-hot encoded vector equals to the total number of feature values, which results in a large and sparse representation of the input. For instance, assuming that the feature *City* contains 100 different values, to represent a value like *Beijing*, the one-hot vector contains 100 binary inputs with only one positive unit (1) and the rest are zeros. Such high dimensional input also increases the parameters that the neural network needs to learn which slows down the convergence. In addition to the sparsity, the features in the one-hot encoded vectors are treated independently by the neural network. The similarity between features like *City* and *Region* are not retained by the one-hot encoded vector.

To reduce the dimensionality of the one-hot encoded inputs and to preserve the feature similarities, the word embedding technique has been widely used in the natural language process area [134, 35]. Figure 4.13 demonstrates how the embedded vectors are represented.

Assuming that x_0 and x_1 are two one-hot encoded feature values with dimension of 5, the embedded vectors k_0 and k_1 are the corresponding weights of the fully connected embedding layer with reduced dimension of 3. For the natural language processing tasks, such representations can be pre-trained by using the Word2Vec model provided by Google [99]. However, for different applications, the embedding layer can be added as the first layer to realize the task specific embedding as is shown in Figure 4.11.

4.3.3 The Deep Attentive Survival Analysis (DASA) Model

In this section, we propose to use the encoder part of the Transformer model described in the previous section to prediction the hazard rate of the censored market price, which is called the Deep Attentive Survival Analysis (DASA) model. The structure of the DASA model is shown in Figure 4.11. It provides an end-to-end training scheme which takes the request features as input and predicts the hazard rate h. We follow the basic encoder structure in [136] and used one encoder with 8 attention heads. The market price distribution is derived from Equation 4.16.

Loss Function

In this section, we denote z as the market price. The probability density function (PDF) of z is $p_z(z)$. As is shown in Equation 4.16, the PDF of the market price can be calculated from the instant hazard function h_j which indicates the probability of the instant occurrence of the event at time j conditioned on the event has not happened prior to time j. In the RTB setting, $\prod_{j<z}(1-h_j)$ represents the losing probability of bidding less than the market price and h_z shows the probability of observing the market price z.

We take the features in the bid request $\boldsymbol{x}_i$ as the input and predict the hazard function h over the discretized bid price space at each impression level. The p_z is represented as in the Equation 4.18. We use b_{max} to represent the upper bound of the bid price. For the uncensored data, the true label is an one-hot encoded vector of size b_{max} with the element indexed by the market price as 1.

$$p_z(z) = P(z|\boldsymbol{x}_i, \boldsymbol{\theta}) = h_z \prod_{j<z}(1-h_j). \tag{4.18}$$

We followed the loss functions in [117]. For the uncensored data, the loss of the observed market price is defined as:

$$L_z = -\sum_{\boldsymbol{x}_i, z_i \in D_{\text{uncensored}}} [log(h_z) + \sum_{j<z} log(1-h_j)]. \tag{4.19}$$

, where z_i is the corresponding true market price for the i^{th} winning auction. For the censored data, it is certain to lose by bidding even lower than the current price. The corresponding loss is defined as:

$$L_{\text{censored}} = -\sum_{\boldsymbol{x}_i, b_i \in D_{\text{censored}}} \sum_{j<b_i} log(1-h_j).$$

, where b_i is the bid price for the i^{th} losing auction. In addition, for the winning auctions, by bidding at any price higher than the observed market price, it is guaranteed to win the auction. Such information can be shared with the censored data. The loss function is defined as followed:

$$L_{\text{uncensored}} = -\sum_{\boldsymbol{x}_i, b_i \in D_{\text{uncensored}}} log[1 - \prod_{j<b_i} (1-h_j)].$$

The total loss of the model takes the combination of the above losses as below where α balances the loss values.

$$L_{\text{total}} = \alpha L_z + (1-\alpha)(L_{\text{censored}} + L_{\text{uncensored}}).$$

4.3.4 Experiments and Results

We conduct experiments to compare the general behaviour of the DASA model with other survival analysis models. The DASA model consists of one transformer encoding layer. A transformer encoder layer has two sub-layers. The first is a multi-head attention mechanism, the second is a fully connected feed-forward network. Residual connections [55] are used around each of the two sub-layers, followed by layer normalization [14]. All sub-layers in the model produce outputs of dimension 512 in order to facilitate residual connections. We use a fixed learning rate of 0.001, state size of 128, batch size of 256 and 8 heads of multi-head attention. We set 20% of the training data as the validation set for deciding early-stopping and avoiding overfitting.

The baseline models are:

- **KM** represents the Kaplan-Meier Product-Limit [73] method. It is a non-parametric statistic model for survival analysis on the aggregated level over the entire dataset. It has been used by several works estimating the market price distribution [142, 164]. In the range of possible discretized bid prices, for each bid price b_i, it counts the number of winning auctions with bid price b_{i-1} and the number of auctions which cannot be won with bid price b_{i-1}, which are represented by d_i and n_i respectively. The winning probability $w(b_i)$ of a bid price b_i and the probability of a market price z_i are given by:

$$w(b_i) = 1 - \prod_{1:b_{i-1}} \frac{n_i - d_i}{n_i}. \tag{4.20}$$

$$p(z_i) = w(z_i + 1) - w(z_i) \tag{4.21}$$

- **Gamma** is selected as a parametric model which fits the market price distribution into a Gamma distribution [169] for each bid request. Comparing with the KM model, the Gamma model provides a more fine-grained impression level solution. However it strongly depends on the pre-defined model form.

- **DeepHit** [81] is a deep feed forward neural network solution which directly maps the request features into the market price probability density function releasing the restrictions of any specific form. Similar to the Gamma model, it provides the impression level prediction. However, in the feed forward network, the weights of each input feature are trained independently without considering when the context (the combination of other features) changes.

- **DLF** [118] is considered as the state-of-the art bid landscape model which is based on a Recurrent Neural Network (RNN) structure as is shown in Figure 4.10. It considers the sequential dependencies in the feature and price space.

- **DASA** is the deep attentive survival model proposed in this dissertation and described in the previous section. Instead of relying on the sequential pattern, the DASA model is based on the attention mechanism to encode the feature contexts and to map them to the price space.

Table 4.8 Performance comparison on ANLP of the herein proposed DASA with respect to the state-of-the-art. The DASA model gets significantly improvement over strong baselines.

Models	ANLP								
	1458	2259	2261	2821	2997	3358	3386	3427	3476
KM*	10.532	14.671	14.665	19.582	16.203	19.253	15.973	16.902	10.507
Gamma*	5.983	6.546	7.969	8.443	7.217	6.416	6.402	6.981	6.145
DeepHit*	5.498	5.798	5.497	5.617	5.615	5.621	5.541	5.552	5.530
DLF*	4.088	5.244	4.632	5.428	4.504	5.281	4.940	4.836	4.012
DASA	**1.8689**	**1.6111**	**1.7752**	**1.4437**	**2.0425**	**1.2382**	**1.4360**	**1.2095**	**1.7600**

The experiments are conducted on the iPinYou datasets described in Section 4.2.5. We reproduced the results in [117] by using the publicly available code[4] as the baseline results with * in Table 4.8. The evaluation metric is the Average Negative Log Probability (ANLP) of the market price which corresponds to the true market price likelihood loss. It is defined as:

$$ANLP = -\frac{1}{|D_{test}|} \sum_{(\mathbf{x}_i, z_i) \in D_{test}} \log p(z_i|\mathbf{x}_i) \tag{4.22}$$

where z_i is the market price in the test set, x_i is the feature vector of a bid request. The lower the ANLP value gets, the better. The result shows that the DASA model significantly outperforms other methods across all the campaigns. The DASA model is chosen to model other bidders' behaviour in Chapter 6.

4.3.5 Discussion

In this section, we proposed an attention-based market price distribution prediction model. In each campaign dataset, a linear bidding function was used to simulate the bidding process and the bid prices were compared with the historical market price in the log. In this way, we obtained new winning and losing sets and assume the market prices of the losing auctions are censored. The bidding datasets with different winning ratios can be generated to test the robustness of the model.

[4]https://github.com/rk2900/drsa.

4.4 Summary

In this chapter, we provide two market price prediction models for the RTB system, from the *price prediction* level and the *bid landscape* level. Since the dynamics of the market attributes to heterogeneous factors like unknown bidders, publishers, and different user profiles, for the price prediction, we have proposed a novel time-series framework with request features aggregation and integration. We have shown that the information from aggregated bid requests contribute to understand the latent bahavior of the bidding market. Furthermore, to generalize the prediction results such that the winning probability can be derived for any given bid price, we have proposed a *bid landscape* prediction model which maps the request features directly to the market price distribution at the impression level. The proposed model provides better predictions than strong state-of-the-art baselines.

Chapter 5

Improving Real-Time Bidding Using a Constrained Markov Decision Process

5.1 Background and Motivations

Programmatic platforms such as RTB gradually takes over as the major tool for the trading of digital ads [60]. Instead of bidding on keywords like in sponsored search [10], or on the context of the website as in contextual advertising [25], RTB targets the best match of users and campaigns at each ad impression level.

In an RTB system, the DSPs play the role of bidding for ad impressions on behalf of the advertisers. An ad exchange receives bids from DSPs and holds second-price auctions; the DSP with the highest bid wins the auction but pays the second highest price, known as the *market price*. According to the Bayesian-Nash equilibrium in the auction theory [76], each bidder's optimal strategy in a second-price auction is to bid the value of each impression evaluated from its own perspective. This is known as truth-telling bidding.

However, in reality, truth-telling bidding may not be the optimal solution due to the budget limit for each ad campaign. Bidding constantly at the true value can lead to running out of budget quickly without covering a wide range of users and impressions [148]. Consequently, the bidder fails to obtain potential profits and might even be subject to heavy losses since the payback of the impressions may be less than the total cost of winning the auction.

The optimization of bidding strategies has been widely studied in the computational advertising industry [30, 160]. The goal of an optimal bidding strategy is to intelligently set the bid price for each ad auction in order to maximize the total number of clicks or profits [111] with a certain budget. This optimization problem fits perfectly into the framework of a Constraint Markov Decision Process (CMDP) [9], which allows to maximize one criterion while keeping another criterion below a given threshold.

In this chapter, we cast the optimization of the sequential bid requests as a CMDP. This is done in order to find the optimal bid price under budget constraints for each auction. A CMDP is defined by the tuple $< S, A, P, R, C, V >$, which correspondingly represents the state set, action set, state transition probability, reward function, cost function, and the value of the constrains. We consider the predicted CTR (pCTR)[1] as the *state*, the bid price as the *action* to take, the *number of clicks* as the *reward* to maximize, the *market price* as the *cost*, and the budget limit as the *constraint*. We integrate the optimization problem and the condition of budget limit into the model and use the linear programming method [46] to solve the CMDP. The policy derived from the solution gives an optimal bid price for each state.

Our contributions are summarized as follows:

- We formalize the bidding optimization problem as a CMDP which optimizes the bidding performance on the impression level. Instead of directly using the features from the impression space, our approach simplifies and limits the state as the discretized pCTR. This results in a significant decrease of the dimensionality of the state space. Another benefit is that we maximize the number of clicks within the constrained budget.

- We introduce the use of conditional market price distribution derived from the joint distribution of historical market price and the pCTR. This captures the correlation between the winning probability and the user level information.

- We show how the well-tuned bidding functions handle the dynamics of the market price, by simulating scenarios where different bidders compete with each other in the same environment. Previous studies compare the bid price of their proposed bidding strategies with only the historical winning price.

[1] The CTR can be seen as the probability of a user clicking on the ad being shown. The pCTR is a prediction of this probability based on features of the publisher site/app and the user visiting it.

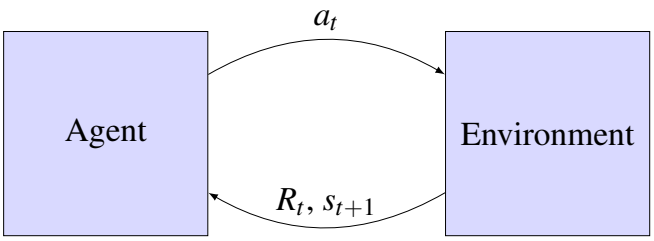


Figure 5.1 Graphical representation of an MDP. At each time t the agent knows the environment state s_t. Based on the transition probability model, it takes action a_t, receives the reward R_t and observes the next state s_{t+1}.

5.2 Problem Formulation

When no additional constraints exist, reinforcement learning is usually formalized as a Markov decision process [131]. However, RTB requires to keep the budget under certain constraints and in the meantime maximize the total number of clicks. A constrained Markov decision process is a class of MDP which can set more than one conflicting objective. A typical case of CMDP is the situation where we want to maximize one criterion while keeping another below a given threshold. Therefore we relied on such models to describe the bidding function.

Figure 5.1 shows a graphical representation of an MDP at time t. A CMDP is defined by the tuple $< S, A, P, R, C, V >$.

- S is the state set.
- A is the action set.
- $P(s'|s,a)$ is the transition probability function, such that $P(s'|s,a)$ is the probability that the system moves to state s' given that it is in state s and the agent takes an action a.
- $R(s,a)$ is the expected reward to maximize, when the system is in state s and action a is taken.
- C is the constraint cost function. $C(s,a)$ is the expected cost acquired when the system is in state s and the agent chooses an action a.
- V is a vector of values that correspond to each constraint.

A *policy* is defined as a function $\pi : S \mapsto A$ which maps the state space S to the action space A, and specifies the action $a = \pi(s)$ that the agent will choose when being in state s.

In general, the policy π can be either stochastic or deterministic. A stochastic policy means that in each state s, action a is chosen with a probability $w(s,a)$. That is:

$$\forall s \in S, \quad \forall a \in A, \quad \pi(s) = a \quad \text{with probability } w(s,a) \tag{5.1}$$

It means that even after finding the optimal action to take in each state based on the historical data, there is still a trade-off between exploration and exploitation. The stochastic policy explores the action and state space by taking actions other than the optimal one and observes the reward. In our work, we focus on finding an optimal deterministic policy and leave the stochastic policy searching as the next step.

The objective of the MDP framework is to find an optimal policy π which maximizes the long term expected reward for the agent. The most frequently used reward function in the application of MDP are the *expected discount reward* and the *average reward*.

In the *discount reward* setting, the rewards received at different time have different weights. The more recent reward is more important than the reward after several steps. At each time step t, the immediate reward is denoted as R_t. The expected discount reward is defined as

$$\overline{R} = \lim_{n\to\infty} \sum_{t=0}^{n} \gamma^t E(R_t|s_t,a_t) \tag{5.2}$$

where γ is the discount factor satisfying $0 < \gamma < 1$, t is the index of each time step, and n is the total number of time steps which can be infinite. $E(R_t|s_t,a_t)$ is the expectation over the possible values of R_t given s_t and a_t. That is,

$$R(s_t,a_t) = E(R_t|s_t,a_t). \tag{5.3}$$

On the other hand, the *average reward* does not distinguish the rewards received at different time so that the w. The expected average reward is given by

$$\overline{R} = \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n} E(R_t|s_t,a_t), \tag{5.4}$$

Table 5.1 Notations and descriptions.

Notation	Description
s	The state represented by the predicted CTR.
a	The action to take represented by a bid price.
$\rho(s,a)$	The probability that the agent is in state s and takes the action a.
$R(s,a)$	The immediate reward after taking an action a in state s.
$C(s,a)$	The cost of taking action a in state s.
V	The constraint of cost.
S	The state space (predicted CTR).
A	The action space (possible bid prices).
$\mathbf{x}$	The user features in each bid request.
θ	The predicted CTR of each bid request.
δ	The market price.
p_{MP}	The conditional market price distribution.

In this chapter, the reward in a RTB system is either the number of clicks or purchases from the users, which weigh the same regardless of time. Furthermore, it can be seen as a continuous process with no terminate state. Hence, in this work the main objective is to maximize the expected *average reward* as defined in Equation 5.4. The notations are summarized in Table 5.1.

The objective of a CMDP is to solve the following optimization problem

$$\begin{aligned} \max_{\rho} \quad & \overline{R} = \sum_{s,a} \rho(s,a) R(s,a) \\ \text{s.t} \quad & \sum_{s,a} \rho(s,a) C(s,a) \leq V \\ \text{and} \quad & \sum_{s \in S} \sum_{a \in A(s)} \rho(s,a) = 1 \end{aligned} \tag{5.5}$$

where ρ is a vector of length $|S| * |A|$ in which each element corresponds to the probability of being in state s and taking action a. The optimal solution of Equation 5.5 is denoted as ρ^* in which the action $a*$ of each state forms the optimal policy π^*.

5.2.1 RTB as a Model-based CMDP

In this section, we present how the optimal policy search in RTB is modeled into a model-based CMDP. As shown in the previous section, we define the bid request as the state. Since all the features in the bid request are categorical features such as country, browser, and device type, a common method is to use one-hot-encoding [56] converting them into binary-valued vectors, which leads to an expansion of the state space.

Directly using the high dimensional binary vector $\mathbf{x}$ in the bid request as the state is very difficult because of the sparsity of the data. However, mapping this feature vector into a lower dimensional space is possible through the CTR prediction $\theta(\mathbf{x})$. The latter takes the feature vector $\mathbf{x}$ as input and calculates the probability of a click. This method has been used to optimize a non-linear bidding strategy [160]. The underlying assumption is that the state dynamics of the RTB system can be completely captured by CTR.

We therefore assume that user dynamics are described by the pCTR $\theta(\mathbf{x})$ and thus project the high dimensional feature space into an 1-dimensional space. The pCTR is the state of the RTB system, $S = \Theta$. The state transition means observing different bid requests.

The probability of observing a state $p(\theta)$ is obtained from the PDF of the predicted CTR distribution using a kernel density estimation. The set of actions A, consists of the set of permitted bids, i.e., $A = \{0, 1, ..., a_{max}\}$, where a_{max} is the maximum bid that a bidder wants to pay for showing its ad.

The reward of an RTB system is usually defined by the advertisers. For branding purposes, the goal of the advertisers can be to maximize the number of ad impressions. However, more commonly, the advertisers are not satisfied by only displaying their ads. Thus, they set the goal as acquiring user interactions like clicks or even further, purchases. In this chapter, the reward of an RTB system is the number of clicks instead of purchases. This is chosen, because for example, in the iPinYou dataset, there are 5 out of 9 campaigns without any purchase in both training and test datasets.

It is a chain process to calculate the expected reward. Firstly, the bidder needs to win an ad auction by placing a bid. The winning probability is derived from the market price distribution. After winning the auction, the actual reward is the probability of a click. Thus, the reward function is a product of the winning probability and the pCTR.

The cost is defined as the market price each bidder pays for a winning auction. The market price, which is the highest among all losing bids, determines how much the winner pays for the winning auction, in other words, how much budget is spent. If the bid price of a bidder is not the highest among all the bidders, the bidder loses the auction with no cost. Modeling the market price reflects the behavior of other bidders, which is also part of the environment which we cannot control. However, since all the bidders set bid prices for the same bid request, our assumption is the market price correlates with the information in the bid request. Therefore, we estimate the market price distribution p_{MP} conditioned on the pCTR (θ), which represents the bid request information. The conditional market price distribution provides the probability of each market price. In this way, the cost can be estimated as a weighted sum of the market price being less than the bid price, as is shown in Equation 5.7.

The system reward, R, and cost, C, are given by

$$R(\theta, a) = \theta \sum_{\delta=0}^{a} p_{\mathrm{MP}}(\delta|\theta) \tag{5.6}$$

$$C(\theta, a) = \sum_{\delta=0}^{a} \delta p_{\mathrm{MP}}(\delta|\theta) \tag{5.7}$$

where p_{MP} is the PDF of the market price that can be derived from historical data. Since CTR is a continuous value ranging from 0 to 1, it is discretized into bins and δ denotes the market price of a bin of CTR. The $R(\theta, a)$ represents the probability of winning an auction by bidding a, multiplied with the probability of receiving a click after winning the auction. The $C(\theta, a)$ represents the expected cost of winning an auction by bidding a.

Our objective is to maximize the expected reward while keeping the expected cost below a certain threshold, V. We interpret V as the maximum of the average cost per impression each bidder is willing to spend. The derived policy from CMDP determines the bid price to set in each state.

5.2.2 Learning from Historical Data: Batch CMDP

In the CMDP model, the correlation between the CTR and the real feedbacks (clicks of impressions) in the historical data is neglected. We argue that it should be utilized as valuable experience to learn from. We thus leverage batch RL to derive the best policy from a set of a priori-known transition samples [78]. The objective of batch RL is to derive a model

reflecting the reality learned from the historical data. The advantage of such an approach is the efficiency in the learning process compared to the model free approaches, like the Q learning algorithm [131]. The latter needs a huge amount of interactions with the environment to converge to the optimal solution which is often not possible in real life applications.

We modify the model proposed in Section 5.2.1 to derive the best policy from historical data. In the CMDP model, the only variable not derived from historical data is the probability for a click, θ, used in calculating the reward in Equation 5.6. We adopt the reward function from the previous model to replace the estimated CTR by the real CTR. For each bin of θ, we calculate the corresponding probability of a click using the true label in the training set. We denote $f(\theta)$ as the probability of a click given θ. In this way, we calibrate the expected reward for each state by using the true reward from the past. We call this new model *Batch CMDP*, formally defined as:

$$R(\theta, a) = f(\theta) \sum_{\delta=0}^{a} p_{\mathrm{MP}}(\delta|\theta) \tag{5.8}$$

$$C(\theta, a) = \sum_{\delta=0}^{a} \delta p_{\mathrm{MP}}(\delta|\theta) \tag{5.9}$$

5.2.3 Market Price Distribution

The market price can be seen as drawing from an unknown distribution generated from the online marketplace. In [23, 37], the authors directly model the market price distribution. However, since the winning probability also relies on the CTR estimation, we introduce the correlation between the winning price and the CTR in the estimation of the market price distribution. We estimate the probability distribution function of the market price p_{MP} using Equation 5.10. This derives implicitly from the joint distribution of the winning price and corresponding CTR, as well as from the distribution of the pCTR according to the Bayes' theorem [17].

$$p_{\mathrm{MP}}(\delta|\theta) = \frac{p(\delta,\theta)}{p(\theta)} \tag{5.10}$$

In order to validate our approach, we prove that a strong correlation exists between the market price and the CTR. A commonly used method for this purpose is to calculate the Pearson's correlation coefficient [5]. This technique is efficient in linear correlation cases, however it fails to capture non-linear relationships. *Mutual Information* (MI) [12] is one of the measures that captures any type of non-linear dependencies between two random variables. MI quantifies the amount of information obtained about one random variable given another random variable. In other words, it measures the degree of uncertainty of one variable knowing the other variable. Formally, the mutual information of two random variables X and Y is defined as

$$MI(X;Y) = \int_Y \int_X p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy \tag{5.11}$$

where $p(x,y)$ is the joint probability density of X and Y, $p(x)$ and $p(y)$ are the probability density function of X and Y respectively. If the two variables are independent, the mutual information equals to 0. Thus, bigger numbers indicate higher dependencies of the two variables. The results of the mutual information between the market price and the CTR are presented in the following section.

5.3 Experiment and Results

We have implemented a CMDP model for bidding trained on two real-world RTB datasets. The bidding results are compared with several state-of-the-art bidding algorithms. In this section, we elaborate the experiments and discuss the results.

5.3.1 Datasets and CTR Prediction

In our experiments, two real-world datasets are used. Due to privacy reasons, the public dataset of RTB bidding logs is very limited. A detailed RTB dataset was released by iPinYou, a leading RTB company in China, for a bidding competition in 2013. This is the only public dataset which contains the historical market price. The other dataset is from OLAmobile, a global mobile advertising company in Luxembourg. The data are collected from 8 campaigns over 6 days, which include 800k impressions and 6k clicks.

The detailed description of the iPinYou dataset can be found in Chapter 4, Section 4.2.5. We applied the data pre-processing procedure[2] used in [161], which utilizes the one-hot-encoding method to convert the categorical features into binary features and we used the logistic regression training like in [160] to estimate the pCTR.

For the reproducibility, our code is available online[3]. We mainly report and publish the results on the iPinYou dataset. Due to the privacy reason, the OLAmobile dataset is not released, but the results are listed as supplementary.

5.3.2 The Correlation Between Market Price and CTR

As introduced in Section 5.2.3, the mutual information is selected as the metric to measure the dependency between the CTR and the market price. Each bidder in the market calculates a bid price based on the same bid request in each auction, therefore we assume the CTR which is computed from the request features correlates with the competition in the market. Table 5.2 and Table 5.3 shows the results of the normalized mutual information of δ and θ calculated for the iPinYou and OLAmobile datasets. The normalization of the mutual information scales the results between 0 and 1, where 0 means the two variables are independent. It can be inferred from the two tables that for all campaigns, $MI(\delta,\theta)$ is higher than 0. The closer the number is to 1, the more mutual information the two variables share. Therefore. we conclude that δ and θ are dependent on each other in both datasets. This supports the rationale of our approach to model the relationship between δ and θ as a batch CMDP.

Table 5.2 Mutual information of the market price δ and CTR θ for the iPinYou dataset.

iPinYou Camp.	1458	2259	2261	2821	2997	3358	3386	3427	3476
$MI(\delta,\theta)$	0.50	0.58	0.59	0.55	0.55	0.56	0.50	0.51	0.53

Table 5.3 Mutual information of the market price δ and CTR θ for the OLAmobile dataset.

OLA Camp.	1	2	3	4	5	6	7	8
$MI(\delta,\theta)$	0.18	0.22	0.40	0.35	0.41	0.16	0.36	0.44

[2]http://data.computational-advertising.org/
[3]https://github.com/manxing-du/cmdp-rtb

5.3.3 Evaluation Methods

The evaluation of the bidding functions is carried out on a per campaign basis. In our experiments, we only focus on the total number of clicks, due to the insufficient number of conversions. In addition, since every campaign has a limited budget, our goal is to maximize the number of clicks given the budget constraint. Thus, the expected cost per click (eCPC) is also used to measure how efficiently the budget is spent.

5.3.4 Experiment 1: Compare Bid Prices To the Historical Data

In the first experiment, each bidding function computes a bid price for the same bid request and the price is compared with the historical market price. If the bid price is higher than the historical market price, then it wins the auction and the cost is the market price. The subsequent clicks are accumulated. Otherwise we assume that the auction is lost with no additional cost. We used the source code available[4] for the work in [23] to generate results for the *Mcpc*, *Lin*, and *RLB* functions.

- **Mcpc**. It sets a maximum eCPC which is the goal of the bidding function. The bid price is calculated by multiplying the max eCPC from the training data with the pCTR.
- **Lin**. As proposed by [111], the bid price depends linearly on the pCTR as $b_0 \frac{\theta(x)}{\theta_{avg}}$,where b_0 is tuned as in [23] and θ_{avg} is the average CTR in the training set.
- **RLB**. A reinforcement learning based bidding algorithm is presented in [23]. In this work, the market price distribution is independently derived from the historical data without considering its correlation with request features as in our proposed model. More details can be found in their paper [23].
- **CMDP**. Our proposed model of CMDP as described in Section 5.2.1
- **Batch CMDP**. The second model we proposed in Section 5.2.2 where the policy is learned by using the real feedback (click or not) as the reward.

We first compare each bidding strategy with limited budget. We determine the budget $B = CPM_{train} * c0 * N_{test}$, where $c0 = 1/32, 1/16, 1/8$, and $1/4$. The $c0$ setting is the same

[4]https://github.com/han-cai/rlb-dp

Table 5.4 Total number of clicks and (eCPC), c0 =1/32.

iPinYou Camp.	AUC	CTR	Mcpc	Lin	RLB	CMDP	Batch CMDP
1458	97.95%	0.084%	392(3.34)	464 (**1.09**)	424 (3.09)	**464** (2.71)	462 (2.8)
2259	67.12%	0.031%	10 (120.63)	7 (173.52)	12 (101.02)	**13**(**89.47**)	10 (119.16)
2261	62.69%	0.028%	7 (137.06)	9 (105.67)	**11** (**87.39**)	8 (118.10)	7 (116.46)
2821	61.28%	0.057%	17 (107.63)	40 (40.26)	**47** (**39**)	39 (45.32)	41 (45.05)
2997	60.79%	0.34%	62 (4.9)	64 (**2.73**)	**82** (3.7)	71 (2.95)	71 (2.98)
3358	97.48%	0.086%	180 (4.75)	189 (3.77)	199 (4.29)	**208** (**3.38**)	203 (4.28)
3386	77.39%	0.082%	56 (23.12)	55 (**5.52**)	61 (21.21)	**92** (12.99)	91 (14.26)
3427	97.23%	0.068%	227 (5.91)	203 (6.55)	261 (5.14)	**292** (4.47)	292 (**4.36**)
3476	95.88%	0.055%	101 (12.76)	162 (**5.92**)	131 (9.87)	181 (7.16)	**188** (6.82)

as in [23] to make our results comparable with theirs. In Table 5.4, the total number of clicks and the eCPC for $c0 = 1/32$ are listed. We find that (1) CMDP and Batch CMDP models outperform all the other bidding strategies in terms of number of clicks when the CTR estimation has higher AUC, since the state only contains the CTR which directly impacts the performance of our model. (2) In terms of eCPC, the CMDP solution does not always achieve the least cost. This is due to CMDP trying to keep the cost per impression under the averaged value in the training set while obtaining the maximum number of clicks. We did not directly set the goal as the cost per click because before getting clicks, we need to win a certain amount of impressions first. As we can see in Table 5.4, for example, the *Lin* function has lower eCPC for campaign 2997 while the number of clicks is fewer than *CMDP*. We should note that although all the bidding strategies have the same budget setting, each of them spends different amount of the budget until the end of the test. In other words, within the same budget limit, different algorithms win different numbers of auctions. The results suggests that CMDP set the bid price efficiently to cover a wider range of impressions and also receives more clicks.

Figure 5.2 illustrates the performance of the bidding functions with respect to different budget settings. The bidding functions are compared in terms of (1) number of clicks (2) Winning rate (3) eCPC and (4) CPM.

Without budget control, the *lin* and *Mcpc* bidding functions win fewer auctions and obtain fewer clicks than all the other bidding functions. Not surprisingly, with the same amount of budget, they win the auctions with high market price, so that the eCPC and CPM are both higher than the others. In general, *RLB*, *CMDP*, and *Batch CMDP* perform better and especially with the low budget setting, *CMDP* outperforms all the other functions.

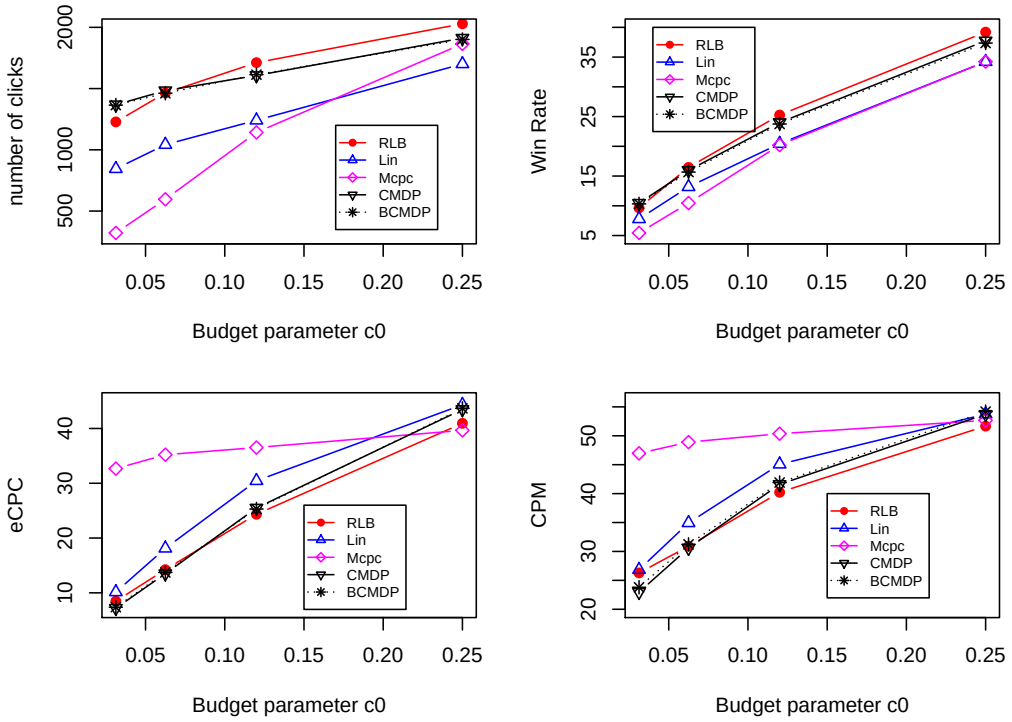


Figure 5.2 Overall bidding performance on iPinYou Data.

In Table 5.5, the results of the OLAmobile dataset show that *Batch CMDP* obtains the most clicks among all the bidding functions. In *Batch CMDP*, the reward is computed by Equation 5.8 in which the probability of a click is derived from the historical clicking probability given the state θ (pCTR). Since the OLAmobile dataset spans 1-2 days, the conditional distribution of the winning probability given the pCTR is more reliable to be used as a factor in the reward function. Thus, it shows that if the model of the environment reflects the reality, *Batch CMDP* provides the optimal policy for making bidding decisions. In the iPinYou dataset, the training data are from 7 days and the test data are from the following 3 days. Our interpretation is that the market price model changes over time, thus the model based on the long term history degrades the performance of *CMDP* and *Batch CMDP*.

We should also note that in Equation 5.5, the sum of $\rho(s,a)$ over the entire state and action is 1. In other words, the policy learned by CMDP also depends on the pCTR distribution in the training set. If the pCTR distribution in the test set changes dramatically, the policy may lead to budget overspending or underspending. The dynamics of the pCTR distribution is a strong focus of our future work.

Table 5.5 Total number of clicks and (eCPC), c0=1/32.

OLA Camp.	AUC	CTR	Mcpc	Lin	RLB	CMDP	Batch CMDP
1	69.96%	0.445%	1(46.41)	3(13.63)	3(15.42)	0 (NA)	**4**(30.59)
2	56.79%	0.466%	2(45.67)	2(39.56)	2(45.23)	3(29.84)	**4**(26.51)
3	73.22%	1.827%	**6**(2.74)	5(1.21)	6(2.51)	2(2.26)	3(1.53)
4	75.18%	0.938%	21(6.04)	22(4.34)	**26**(4.86)	14(8.94)	26(6.82)
5	71.35%	1.833%	3(3.4)	3(3.39)	4(2.54)	4(2.23)	**13**(2.64)
6	58.15%	0.465%	8(30.01)	16(14.28)	10(23.9)	6(39.55)	**28**(38.54)
7	67.69%	1.237%	4(60.95)	14(16.53)	5(48.42)	9(27.21)	**23**(23.83)
8	68.07%	0.554%	33(9.46)	51(5.78)	61(5.07)	71(4.37)	**88**(5.93)

In addition, we further explore the importance of correctly modeling the market price. In our implementation, we explain how to correlate the market price distribution with the pCTR in Section 5.2.3. Here we uniformly sample a set of market prices and use it to derive a policy from the CMDP model. Figure 5.3 shows the difference of winning probabilities between the true market price and a set of uniformly distributed market prices for campaign 1458. In this case, it clearly shows that the random market price model underestimates the winning probability. We replace the market price model in Equation 5.6 with the random model and show the results in Table 5.6. The result shows the negative performance impact only on campaign 1458 and 2997. For the other campaigns, the two policies get similar number of clicks.

However, since the winning probability and the cost are underestimated by the random model, in order to get more reward, the derived policy sets higher bids than the policy used for the first two columns in Table 5.6. Thus, in the bidding log, we observe that the budget is exhausted early in the test set resulting in the inability to bid for any more requests. On the contrary, the policy derived from the conditional market price model sets more low bids, when the data in the test set are exhausted, each campaign still has budget left. Therefore, each campaign can participate in more auctions and potentially get more clicks than using the random market model.

5.3.5 Experiment 2: Compare Bidding Functions in the Same Environment

In the previous experiment, the performance of the bidding functions is independently compared with the historical winning prices. However, in a real world scenario, every bidder

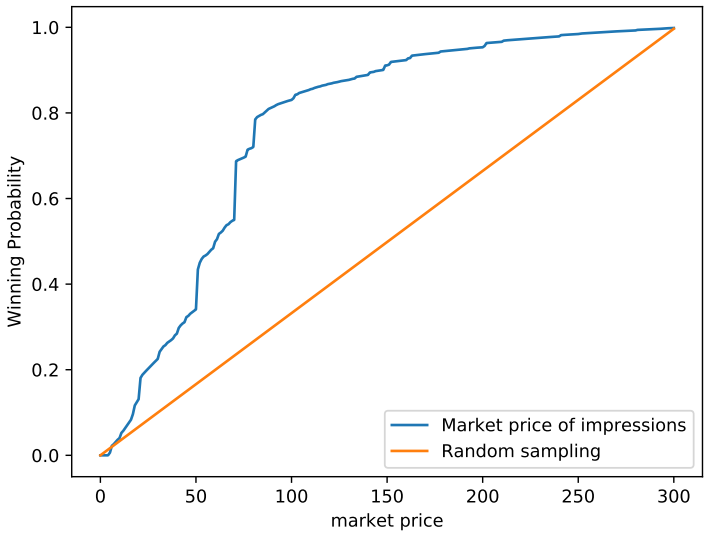


Figure 5.3 Winning probability comparison for camp.1458.

Table 5.6 Total number of clicks and (eCPC), c0 =1/32.

iPinYou Camp.	CMDP	Batch CMDP	CMDP random	Batch CMDP random
1458	464 (2.71)	462 (2.8)	312(4.24)	317(4.17)
2259	13(89.47)	10 (119.16)	13 (93.33)	12(101.1)
2261	8 (118.10)	7 (116.46)	10(96.28)	10(96.28)
2821	39 (45.32)	41 (45.05)	41(45.05)	42(43.97)
2997	71 (2.95)	71 (2.98)	56(5.49)	51(6.03)
3358	208 (3.38)	203 (4.28)	211(2.56)	210(2.28)
3386	92(12.99)	91 (14.26)	91(14.14)	90(14.57)
3427	292 (4.47)	292 (4.36)	298(4.19)	297(3.97)
3476	181 (7.16)	188 (6.82)	170(7.63)	183(7.09)

tries to improve his/her bidding functions at any time. Thus, we present the impact of the fluctuation of the market price distribution on the bidding strategies. We simulate the scenario by assuming that the historical winning prices come from a single virtual bidder and let the other bidding functions compete with each other as well as with the virtual bidder. The winner is the one which bids the highest and the winning price is the second highest price. If more than one bidders set the same price, all of them win the auction, which produces the maximum number of auctions and clicks each function can win in this setting.

Table 5.7 Comparing bidding functions in the same environment, c0=1/32.

Camp. 1458	mcpc	Lin	RLB	CMDP	Batch CMDP
Multi_win_clk	55	367	361	101	100
Sig_win_clk	9	46	28	2	14
Multi_win_imp	216	700	2978	3512	3350
Sig_win_imp	30991	84	13731	4755	41783
total_ecpc	23269	453	2608	4647	10059
Camp. 2997	mcpc	Lin	RLB	CMDP	Batch CMDP
Multi_win_clk	0	0	7	5	3
Sig_win_clk	45	0	48	4	10
Multi_win_imp	11	0	2753	2736	2369
Sig_win_imp	15560	0	25799	1051	3718
total_ecpc	7458	0	5441	3624	4077

If more than one function bids the same price, all of them are considered to be winners. The corresponding clicks and impressions are denoted as *Multi_win_clk* and *Multi_win_imp* respectively in Table 5.7. Meanwhile, the single winner case is represented by *Sig_win_clk* and *Sig_win_imp*. The *RLB*, *CMDP*, and *Batch CMDP* models are trained using the historical market price and the experiment was running on the test data. In this setting, the market price in the test set shifts towards higher prices when more than one function bids higher than the historical market price.

The result suggests that for the campaigns with high AUC (e.g. campaign 1458), the linear bidding function targets the right impressions to bid high. Other functions, for example, like *CMDP* wins 10 times more impressions but only get $1/3$ of clicks as *Lin*, which significantly increase the eCPC. On the contrary, *Lin* loses its advantage when the pCTR is not accurate since it only relies on pCTR to calculate the bid price. One extreme case is campaign 2997 having the lowest AUC in the dataset. *Lin* sets the bid price too low comparing to other functions and thus does not win any impression. The results also show that the eCPC should not be the only metric to evaluate how well the bidding function performs. For example, for campaign 2997, *CMDP* has a lower eCPC while having 6 times less number of clicks than *RLB*. In this case, *CMDP* bids more conservative than *RLB* since *CMDP* follows the policy learned from the pCTR density function in the training set.

5.3.6 Discussion

In our study, we notice that the CMDP model does not fully use the budget for each episode. The reason is that in order to generalize well, the constraint in the CMDP model is not set to be a specific number as the total budget. Instead, it is set as the expected cost per impression. As a result, the model neglects the current budget status. To solve this problem, the current budget can be added into the state.

Another limitation is that the CMDP model is solved by linear programming, which does not scale well with large number of states and actions. Meanwhile, continuous states and actions need to be discretized for the CMDP model. Deep reinforcement learning framework can be used to address such problems due to its capacity of fitting in large scale continuous inputs.

The CMDP model follows a deterministic policy without exploring other actions, which may fail to observe more rewards. For instance, by bidding aggressively to win more auctions, the bidder have more chances to show its ads to the users, which in turn may get more clicks. Thus, adding an exploration setting in the policy learning process is important.

5.4 Summary

In this chapter, we formalized the bidding problem in the RTB system as a constrained Markov decision process. We use linear programming to maximize the total reward with a cost limit. The reward is either derived from the CTR estimation (in CMDP) or from the historical observations (in Batch CMDP), in which case the policy is learned given the training data. We use Bayesian inference to obtain the market price distribution, which not only considers the correlation between the market price and the state (pCTR) but also captures the dynamics of market price. Our model outperforms the state-of-the-art bidding functions in terms of the total number of clicks constrained to a limited budget. However, when the bidding functions compete with each other, linear bidding performs the best for campaigns with a high AUC while RLB obtains more clicks for campaigns with a low AUC. The *CMDP* model relies on the correlation of the historical market price distribution and the pCTR distribution, thus bids more conservatively compared to the others.

Chapter 6

Learning in Real-Time Bidding with Partially Observable Opponents

6.1 Background and Motivations

RTB is a common online advertisement inventory trading mechanism in which each ad display is sold through *real-time auctions*. It allows the advertisers to target potential users at the level of individual ad impressions. Upon each user's visit, each ad slot on the publisher's site (or app) is sold through auctions. Most market-places use a so-called *second price auction* [76]. In such auctions, the bidder with the highest bid price wins the opportunity to show its ad, which is also called an *ad impression* and the winner observes and pays the second highest price, known as the *market price* or the winning price.

The market price is only known by the highest bidder, the winner. The other bidders, who lost the auction, only know that the market price could be equal to or higher than their bids. Therefore, the market price is considered to be right-censored. In addition, the bidding environment is highly dynamic. In each auction, an unknown number of bidders will participate and the set of bidders varies over different auctions. To compete in such second price auctions, in theory, bidders would benefit from bidding their estimation of each impression's true value as the bid price [76]. In this way, the market converges to the Nash equilibrium in which no bidder can benefit more by unilaterally changing its strategy.

However, in practice, the bidder may not always follow the optimal Nash equilibrium strategies due to various reasons. For instance, since the bidders are usually constrained by a certain budget, to avoid running out of money quickly without observing more valuable impressions, the optimal bid price usually deviates from its true value. In addition, the number of participants in each auction is unknown and from each bidder's perspective, it may compete with different opponents at every step during its lifetime.

To obtain an optimal bidding strategy in such a stochastic environment with a large number of unknown participants is the major challenge in RTB. From one bidder's perspective, when other bidders adapt their strategies, they modify the competitions in the environment and consequently change the reward distribution for each bidder. The environment changes due to the concurrent actions of the bidders are called the *environment dynamics*. Every bidder learns to act and adapt to the changing environment concurrently which makes the environment to be *non-stationary*. The single agent RL based bidding solutions focus on modeling one bidder and its interaction with a stationary environment [23] which is not suitable for dealing with the non-stationary environment. In the latter, it is important for each bidder to adapt its strategy based on the *joint actions* of all the participants.

From one bidder's perspective, the other bidders are called opponents. The next challenge is to deal with the observability of the opponents. The opponents can be partially observable or fully observable under different formulations. Zhao et al. [166] design a multi-agent deep reinforcement learning RTB algorithm for a sponsored search system where the ad agents share a global reward function as the cooperative goal. In this study, the common reward is observable to all the agents but the states and actions of each agent remain unknown to each other. Jin et al. [68] handle the large number of advertisers and customers on Taobao, a popular e-commerce platform in China, by aggregating them into clusters and formalize a multi-agent bidding process on the cluster basis. They assume that each cluster has full observations of each others' status and has either individual reward function (in the *competitive* mode) or shared reward function (in the *cooperative* mode). In practice, in the bidding market, each DSP works on its own without knowing either its opponent's strategy, reward, or budget status. For each bidder (a DSP), the only possible information available of its opponents is the partially observable market price.

With the number of opponents increasing, modeling every opponent's action becomes implausible and computationally expensive. To analyze such highly dynamic environments with a large number of learning agents, recent research works [151, 51, 63] have been focused

on using MFE to approximate the Nash equilibrium. To avoid modeling the interactions between every two agents, MFE based models approximate each agent interacts with the estimated average action of its neighboring agents. Motivated by the MFE concept, we have designed, implemented and evaluated an *opponent-aware* bidding algorithm for the multi-agent environment which requires no prior assumptions on the opponents' bidding distribution.

This algorithm differs from all existing multi-agent solutions in the RTB domain. Unlike other solutions targeting the advertisers which have full observations of each other's state and reward, our algorithm optimizes the campaign performance for a single advertiser which only partially observes others' actions (market prices). We provide a prediction model to infer the opponent's actions. Relying on the MFE, the proposed model simplifies the opponent modeling by taking the market price as the *aggregated* actions. We address the prediction of partially observable opponent actions and adopt the DASA model proposed in Chapter 4 to map the state features to the market price distribution. Furthermore, our solution integrates the opponent model into the policy learning framework for the bidding agents. The proposed bidding algorithm enables each bidder to adapt and optimize its strategy given its estimation of the opponents actions. The experiments show that the market reaches the equilibrium under different budget constraints and the opponent-aware bidding algorithm accelerates the convergence in the multi-agent environment.

6.2 Preliminaries

In this section, we introduce the background knowledge of game theory to support defining the solution in the multi-agent bidding environment in the next section.

6.2.1 Game Theory

Game theory [102] is defined as the study of making strategic decisions for multiple rational individuals interacting with each other. A set of terminology is defined in the game theory domain in a similar fashion as in the domain of reinforcement learning. The terminology is summarized in Table 6.1. The interaction between at least two entities is called a *game* and the entities involved are called *players*. In a *game*, the goal of a player is to maximize its

Table 6.1 Summary of terminologies used in Game Theory and Reinforcement Learning.

Game Theory	Reinforcement Learning
player	agent
action	action
game	environment
strategy	policy
payoff	reward

Table 6.2 The normal-form payoff matrix of the prisoner's dilemma game.

		Player 2	
		Cooperate	Defect
Player 1	Cooperate	-3, -3	0, -5
	Defect	-5, 0	-1, -1

expected reward, which is also called the *payoff*. In the following sections, the terminologies in these two domains are used interchangeably.

A classic example of a two-action two-player game is known as the *Prisoner's Dilemma* [114] where the payoff of the two players can be summarized as is shown in Table 6.2. In this game, the two actions that each player can choose from are to cooperate or to defect. In each cell, the two numbers represent the *payoffs* of each player respectively. A game which can be represented in this form is usually called a Normal-form game defined in Definition 1 [57].

Definition 1 (Normal-form game) *A Normal-form game is represented by a tuple* $< N, A, U >$*, where N is a finite set of players;*
$A = A_1 \times \ldots \times A_N$ *where* A_i *represents the finite actions that the player i can choose;*
$U = (U_1 \ldots U_N)$ *where* $U_i : A \rightarrow \mathbb{R}$ *is the utility function which maps the action space to the real-valued payoff of player i.*

In such a game, each player aims to find the optimal action in order to maximize its own payoff while assuming the other players are planning in the same way. In a multi-agent game, the best strategy of each player depends on other agents' strategies as well. Game Theory introduces the *Nash equilibrium* (NE) concept [103] to describe the optimal strategies in such contexts.

Definition 2 (Nash equilibrium) *The optimal set of strategies is denoted by* $\boldsymbol{\pi}^* = (\pi_1^* \ldots \pi_N^*)$ *where* π_i^* *is the optimal strategy for player i. The Nash equilibrium is reached when the utility function of the strategy* π_i^* *satisfies:* $U_i(\pi_i^*, \pi_{-i}) \geq U_i(\pi_i, \pi_{-i})$.

where $\pi_{-i} = \pi_1 \dots \pi_{i-1}, \pi_{i+1} \dots \pi_N$ denotes the strategies of all players other than player i. Under the equilibrium, every player follows its own strategy which maximizes its utility function while other players keep their strategies unchanged. No player can gain more by unilaterally changing its strategy.

When the number of players and the number of possible actions get larger, it is not scalable to represent the utility functions for every action. In addition, a more complex game may require each player to make sequential decisions under different states as in the single agent reinforcement learning scenario. Here we introduce the concept of *stochastic games* [128] to describe such situations.

Definition 3 (Stochastic games) *A stochastic game is a tuple (S,N,A,P,R) where S is a finite set of states; N is the finite set of n players;*
$A = A_1 \times \dots \times A_N$ where A_i represents the actions available for player i;
$P : S \times A \times S \to \mathbb{R}$ is the transit probability function for state and action pairs, where $P(s,a,s')$ is the probability that a player takes an action a and transits from state s to s';
$R = r_i \dots r_N$ where $r_i : S \times A \to \mathbb{R}$ is the payoff function for a player i taking an action a at the state s.

6.3 Problem Formulation

In this section, we formulate the sequential second price auctions as an n-player stochastic game represented by a tuple $< S,N,A,P,R >$. Here, the players are the bidders who compete in the same bidding environment. The environment dynamics come from the simultaneous actions from all the bidders, a.k.a the bid prices. We refer to the bidders as the agents interchangeably in the rest of this dissertation. At each time t, the state of n-agents is denoted as $\boldsymbol{s}_t = (s_t^1 \dots s_t^n) \in S$. Correspondingly, the joint action of all agents at time t is represented as $\boldsymbol{a}_t = (a_t^1 \dots a_t^n) \in A$. The policy of an agent is defined as: $\pi^i : S^i \mapsto A^i$. After taking an action a_t^i, agent i transits to the next state: $s_{t+1}^i \sim P_t^i(s_t^i, \boldsymbol{a}_t)$ and receives a reward $r_t^i \sim R_t^i(s_t^i, \boldsymbol{a}_t)$. We note that the transition probability and the reward are determined by the joint action $\boldsymbol{a}$. Since all the agents set their prices simultaneously, from each agent's point of view, only the winner pays the market price and the others keep their budget unchanged for the next auction. Correspondingly, only the winner has the chance to get the user behaviour dependent reward, e.g. the click through rate (CTR) while the others receive zero reward.

Unlike other n-player games, where the full environment state is usually considered in the equations above, in RTB, each agent only observes its own state s_t^i at each step. The agents are coupled only through their actions.

The value function of a certain policy π is defined as:

$$v_\pi^i(s) = \sum_{t=0}^{\infty} \gamma^t \, \mathbb{E}_{\pi,P}[r^i(s_t, \boldsymbol{a}_t)|s_0 = s, \pi], \tag{6.1}$$

where $\gamma \in [0,1)$ denotes the reward discount factor over time.

As stated in [151], a common assumption of such an n-player game is that each agent is unaware of the game dynamics or the reward function, but it observes the previous action and the immediate reward of other agents. For a single agent, the Q-function is extended by taking the joint actions of all agents $\boldsymbol{a} \triangleq [a^1, \ldots, a^n]$ as the formulation below:

$$Q_{\boldsymbol{\pi}}^i = r^i(s, \boldsymbol{a}) + \gamma \mathbb{E}_{s' \sim P}[\mathbb{E}_{a \sim \boldsymbol{\pi}}[Q_{\boldsymbol{\pi}}^i(s', \boldsymbol{a})]]. \tag{6.2}$$

The objective of each agent in an n-player game is to derive an optimal policy π_* which maximises its value function as shown in Equation 6.1. However, each player has its own reward function and is dependent on the other players. Hence, it is not as clear a concept define an optimal policy as in single-agent problems that only maximize the state value given in Equation 6.1. It may be ineffective due to players having different rewards since one agent might hamper the objectives of other agents, the bidding environment becomes non-stationary as well.

To address this problem, one common solution is equilibrium-based approaches. We start from one agent's view, the stochastic formulation of RTB can be simplified as a two players game, where the other player is the one with the second highest price in the market. The winning price can be modeled as the joint action from all the other opponents and is partially observable. Then, we adopt an equilibrium-based policy, also called the Nash equilibrium (NE) policy [61], which satisfies

$$v^i(s, \boldsymbol{\pi}_*) \geqslant v^i(s; \pi^i, \pi_*^{-i}). \tag{6.3}$$

where $\boldsymbol{\pi}_*$ is the NE optimal joint policy of all the agents. In the equilibrium stage, no agent can further improve its value function while other agents keep their policies unchanged.

Here π_*^{-i} denotes the optimal joint policy of all the other agents except agent i: $\boldsymbol{\pi}_*^{-i} \triangleq [\pi_*^1, \ldots, \pi_*^{i-1}, \pi_*^{i+1} \ldots, \pi_*^N]$.

When the number of agents $n \to \infty$, the classic multi-player game becomes intractable, thus in [62], the authors proposed the Mean Field Game (MFG) to model the large number n-player game. The conventional MFG assumes the agents have complete information of the actions and the rewards of other agents [151]. On the contrary, in RTB, the actions of other agents are not observable unless the agent wins the auction and the highest bid from the other bidders is revealed. To generalize the conventional MFG to the MFG with incomplete information, in this study, we propose an opponent model to infer the unobservable actions of other players. From one agent's perspective, at each time t,

$$s_{t+1}^i \sim P(\cdot|s_t^i, \boldsymbol{a}_t) = P(\cdot|s_t^i, a_t^i, \boldsymbol{a}_t^{-i}). \tag{6.4}$$

where $\boldsymbol{a}_t^{-i}$ denotes the actions taken by the agents other than agent i. If all the other agents follow their optimal policies, then $\boldsymbol{a}^{-i} = \boldsymbol{a}_*^{-i}$. It suggests that the bid distribution of other agents is fixed. Therefore, Equation 6.3 can be written as

$$v^i(s, \pi_*^i, \boldsymbol{a}_*^{-i}) \geqslant v^i(s; \boldsymbol{\pi}^i, \boldsymbol{a}_*^{-i}). \tag{6.5}$$

To learn an equilibrium joint policy, updates of Q-values rely on the computation of an equilibrium metrics, Nash-Q, defined in [61]:

$$\boldsymbol{Q}^{\text{Nash}}(s, \boldsymbol{a}) = \mathbb{E}_{s' \sim P}[\boldsymbol{r}(s, \boldsymbol{a}) + \gamma v^{\text{Nash}}(s')]. \tag{6.6}$$

The Q-function in Equation 6.6 is approximated by neural networks and parameterized with the weights ω. As discussed above, we replace the joint actions of other bidders $\boldsymbol{a}_t^{-i}$ by a virtual and aggregated opponent whose bids are always the market price. Thus, the Q-function can be expressed as

$$Q^i(s, \boldsymbol{a}) = \int_{a^{-i}} Q(s, a^i, a^{-i}) \phi(a^{-i})\, da^{-i}. \tag{6.7}$$

In Equation 6.7, $\phi(a^{-i})$ represents the distribution of the opponent's actions.

In the ideal MFE scenario, it supposes that all agents take a fixed and steady bid distribution and their own belief of the bid valuation as the prior knowledge to optimise their strategy [63].

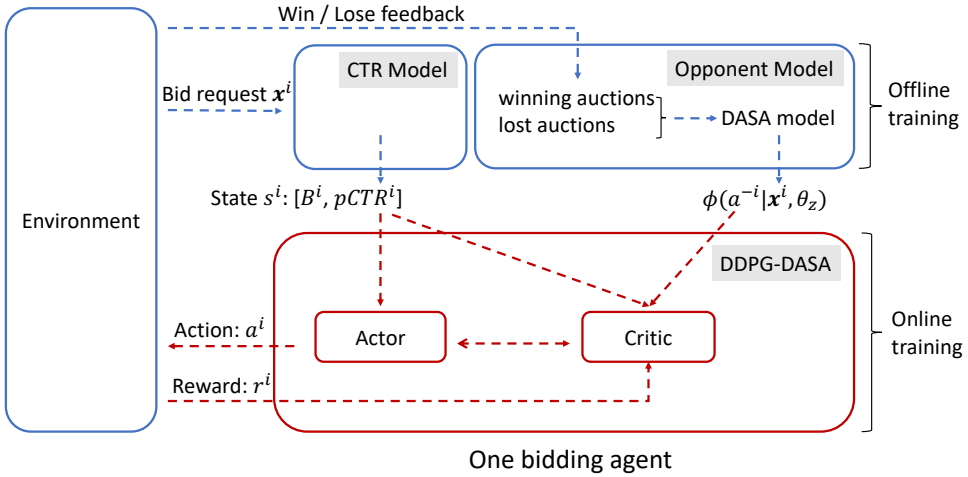


Figure 6.1 The architecture of the DDPG-DASA model.

The policy that each agent follows is stationary. In practice, the bid valuation is estimated by the CTR prediction model and the opponent bid distribution can be estimated as the market price model. Thus, in this work, we adopted two pre-trained models into the framework to fulfill the above assumption.

Figure 6.1 depicts the architecture of the components used in this work. The CTR model takes the feature vector $\mathbf{x}$ in the historical bid requests as input and binary labels 1 and 0 indicating an ad click or no click respectively. The pCTR is later used to construct the agent state s and the reward r in the updated DDPG model with the opponent model DASA (DDPG-DASA). In the following sections, the opponent model (DASA) and the bidding model DDPG-DASA are described in details.

6.3.1 Bidding Model

Under our repeated second-price auctions setting, in every auction, all the agents are facing the same bid request. The agents bid for the same ad campaign upon different requests with unknown number of opponents at each auction. The RL agent adopts the framework of DDPG [89] method to learn the policy for setting a bid price in the continuous space. We simulate the bidding process as an episodic task. Each episode consists of 1000 auctions and for each agent i, the budget B_i is reset for every episode.

State. For the DDPG agent i, we take the budget left in an episode B^i and the $pCTR^i$ as the state $s^i =< B^i, pCTR^i >$

Action. Following the settings in [68], the action a_i is set to be a scalar which controls the bid price and is bounded to be in the range of $[0,1]$. The final bid price is calculated by $b_f = \min(b_{max} \times a^i, B^i)$, where b_{max} is the upper bound of the bid price. The market price or the aggregated actions from the opponents are denoted as a^{-i}.

Reward. The reward is usually the Key Performance Indicator (KPI) defined by the advertisers, for instance, a click, a purchase or the profits. But such reward signal is usually too sparse for the agent to learn. Therefore, in this study, we assign the $pCTR$ as the reward for all the winning auctions, even without the real click [144]. For the losing auctions, since no price is paid, the reward remains as zero. For an agent i, the actor network takes state s^i, which consists of the predicted CTR and the budget left in the current episode as input for a deep neural network parameterized with θ_π. The output of the actor network is an action a^i to take in the range of $[0,1]$.

Opponent Modelling

The DASA model serves as the opponent model in this chapter. The thorough study of the DASA model can be found in Chapter 4, Section 4.3. Here, we adapt the prediction target into the notations used in the current chapter. We use a^{-i} to represent the action taken by the opponents, a.k.a the highest price from all the other participants in the auction. In this section, we denote $a^{-i} = z$, where z is the market price. The probability density function (PDF) of z is $\phi(z)$.

$$\phi(z) = \phi(a^{-i} = z|\boldsymbol{x}^i, \theta_z) = h_z \prod_{j<z}(1-h_j). \tag{6.8}$$

Action function

$$a^i = \pi^i(s^i, \theta_\pi) = \pi^i([b^i, pCTR^i], \theta_\pi). \tag{6.9}$$

In the vanilla version of DDPG algorithm, the critic function $Q(s^i, a^i)$ takes the state and action pair from a single agent. In our model, the Q-function is approximated by the mean field theory by integrating the opponent's action distribution. As is shown in Equation 6.10, $\phi(a^{-i}|\boldsymbol{x}^i, \theta_z)$ is the market price distribution obtained from the opponent model. The action a^{-i} is not directly observed from the environment, since the result of the auction can only be see after placing a bid price. The market distribution provides the agent's belief of the opponents actions. The indicator function allows the agent to account for the Q value only in

the case of bidding higher than the market price. Since when the action a^i is lower than a^{-i}, the agent cannot win such auctions, thus, the Q value should be zero.

Critic function

$$Q^i(s^i, \boldsymbol{a}) = \int_{a^{-i}} Q(s^i, a^i, a^{-i}) \phi(a^{-i}|\boldsymbol{x}^i, \theta_z) \mathbb{1}[a^{-i} < a^i] \, da^{-i}. \tag{6.10}$$

The pseudo code of the DDPG-DASA algorithm is shown in Algorithm 1.

Algorithm 1: DDPG-DASA.

Initialize actor network $\pi(s, \theta_\pi) = a_i$ and critic network $Q(s, \mathbf{a}|\omega)$ with weights θ_π, ω
Initialize target network π' and Q' with $\theta'_\pi \leftarrow \theta_\pi$ and $\omega' \leftarrow \omega$
Initialize replay memory with size K;
for *episode = 1 to E* **do**
  receive state s_0 and sample $a_0 \sim \pi(s_0, \theta_\pi)$;
  Initialize a noise generator N for action exploration
  while s_t *not terminate* **do**
    Select an action $a_t = \pi(s_t, \theta_\pi) + N_t$ and execute ;
    Observe r_t, s_{t+1};
    Store (s_t, a_t, r_t, s_{t+1}) in the replay memory;
    if $t \equiv 0$ *mod* K **then**
      sample a minibatch M from the replay memory
      $y_j = r_j + \gamma Q'(s_{j+1}, \boldsymbol{a}'_{j+1}, |\omega')$
      update critic by minimizing the loss $L = \frac{1}{M} \sum_j (y_j - Q(s_j, \boldsymbol{a}_j|\omega))^2$;
      update actor $\theta_\pi \leftarrow \theta_\pi + \frac{1}{M} \sum_j \nabla_a Q(s_j, \boldsymbol{a}_j|\omega)|_{s=s_j, a=\pi(s_j)} \nabla_{\theta_\pi} \pi(s_j|\theta_\pi)|_{s_j}$;
      update target network:
      $\theta' \leftarrow \tau\theta_\pi + (1-\tau)\theta'_\pi$;
      $\omega' \leftarrow \tau\omega + (1-\tau)\omega'$
  end
end

6.3.2 Multi-Agent Mean Field Approximation

The mean field equilibrium in RTB requires a consistency check of the bid distribution [63]. Let ϕ be a bid distribution and π^i denote a stationary policy for an agent facing bidding decision. The mean field equilibrium is achieved if it satisfies the following definition:

Definition 4 *The repeated second-price auction Mean Field games admit at least one MFE[63], with strategy π, if:*

1. *$\pi(\cdot|\phi)$ is an optimal strategy given ϕ.*
2. *ϕ is the steady state bid distribution given π.*

Single-Agent Steady Market Distribution

We start from the simplest scenario: a single learning agent bids against a steady market price distribution. In this setting, we assume the linear bidders have fixed strategies which means they do not update their strategies upon other bidders' actions. In addition, given the dynamic attributes of the bidders, from bidder i's point of view, the bids from its opponents are identically and independently distributed. As we discussed in Section 6.1, in practice, the bidders participating in every auction change over time. Here we assume the departure and the arrival rate of bidders remains steady, which guarantees the stationarity of the bid price distribution of the opponents. Although the opponent bids are partially observable, this allows us to approximate a fixed opponent model and use it in the mean field model.

Multi-Agent Dynamical Market Distribution

From the above single agent scenario, here we extend to discuss the multi-agent bidding environment, where all bidders are learning and adopting their strategies and they share one steady bid distribution ϕ. Taken ϕ as the prior knowledge, each agent optimises their bidding strategy by adopting the DDPG-DASA algorithm which in turn induces dynamics in the overall bid distribution. In the multi-agent bidding process, we assume that the number of competing agents is large. In each auction, a finite number of agents is randomly selected. The goal of winning the second-price auctions is to bid higher than the second highest price. The losing bid prices from other bidders are not observable. To summarize, in each auction, 3 types of bidders can be observed, namely a winner, a bidder with the market price, and the losing bidders. Same as in the previous section, we assume the departure and the arrival rate of bidders remains steady. Based on this assumption, we use 3 bidders to simulate a multi-agent bidding environment. At either the end of the episode or when the budget of a bidder has been exhausted, a new episode starts and we initialize the same budget to

Table 6.3 Dataset Statistics.

campaign ID	Impressions	Clicks	CTR	# Discrete Feature Values
2259 (Trainning Set)	835,556	280	0.034%	40347
2259 (Test Set)	417,197	131	0.031%	40347

every bidder. In our experiments, each bidder trains its own DDPG-DASA strategy. Due to a learning rate decay setting, eventually each bidder will converge to a stationary agent (learning rate ≈ 0), thus the normal theorem by [63] holds. In the MFE, each bidder is facing i.i.d highest opponent bids and has no incentive to change its bidding strategy. However, it is important to note that before the equilibrium is reached, the bid distribution would change as the market evolves. Thus, it is important for the agents to infer the bid distribution over time.

6.4 Experiments

In this section, we present the empirical study of the DDPG-DASA bidding algorithm in both single-agent and multi-agent scenarios on a real-world bidding dataset. We have published the implementation code of the experiments.[1]

6.4.1 Datasets and Experimental Setup

In this work, the bidding experiments are conducted over the public real-world dataset, iPinYou, one of the leading ad companies in China. The detailed description can be found in Chapter 4, Section 4.2.5. The statistics of the ad campaigns selected in the study are shown in Table 6.3.

We follow the data pre-processing and feature engineering procedure in [161]. Since in the iPinYou dataset, the original market price of the impressions is recorded, we initiate all the agents with the budget to be proportional to the total cost in the training data. In this way, it allows us to simulate the auctions offline. Given each bid request, each agent places a bid price and follows the second-price auction principles to decide the winner of the auctions and the click labels in the log are used to train the CTR model. We compare the bid price

[1] https://github.com/manxing-du/Know-your-enemies.git.

generated by the agents in our experiment, thus the original market price in the iPinYou bidding log is not included.

As is shown in Figure 6.1, the CTR estimator is trained offline by adopting the widely used FTRL-logistic regression model [96]. In both single and multi-agent scenarios, we begin with running the bidding simulation over the training set and log the bid price of each agent and select the second highest price as the market price. The opponent model in Figure 6.1 takes the simulated bid log and the features in the original bid requests as input to predict the impression level market price distribution as described in Chapter 4.

Once the CTR model and the opponent model are trained, we repeat the bidding simulation on both training and test sets. In this round, the DDPG agent learns the policy while having the prediction of the distribution of its opponent. We begin with setting one DDPG agent in the environment and keep the other bidders using simple and static bidding strategies, for instance, linear bidding function. In this setting, we demonstrate the advantage of the learning agent over the static agent without the learning process. Furthermore, we extend to the multi-agent scenario where all the agents learn their strategies with its estimated opponent model.

In this study, we consider the bidding process as an episodic task and each episode consists of $K = 1000$ auctions. Each episode has a fixed budget $B = \text{CPM}_{\text{train}} \times 10^{-3} \times K \times c_0$, where $\text{CPM}_{\text{train}}$ is the cost per mille impressions in the training data and c_0 is the budget constraint ratio: c_0= 0.125, 0.25, and 0.5.

6.4.2 Single DDPG agent with Steady Market Price Distribution

In this section, we assume that there is one learning agent running DDPG algorithm that competes against N bidders with fixed strategies, for example, a linear bidding function: $b_i = pCTR * \alpha_i$, where α_i is a fixed linear ratio. In practice, N is always unknown and in each auction, a random set of the N bidders is selected. In the second-price auction, the most important opponent is the bidder with the second highest price among all the bidders. In this study, we set up two linear bidders and one DDPG bidder. The bidders take the same *pCTR* from the CTR prediction model. By injecting Gaussian noises into the *pCTR*, we simulate the stochastic environment of random bidders with different *pCTR* as their states in each

auction. Comparing the bid price generated by the three bidders, we log the market price and use it for training the DASA model offline as described in Chapter 4, Section 4.3.

In the next round, we replay the bidding game again to train the same DDPG agent from scratch with the opponent model integrated, a.k.a the DDPG-DASA model. We run the same experiment with three random seeds and show the averaged results as follows. In Figure 6.2, the number of clicks obtained by the three bidders are listed for one selected ad campaign, 2259. In general, the agent with the DDPG-DASA model obtains more clicks than the vanilla DDPG model when competing with the fixed linear bidders. In Figure 6.3, it shows the number of impressions each agent obtains. Same as in Figure 6.2, the rows represent three budget settings, where c_0= 0.125, 0.25, and 0.5. The left column shows the results from the DDPG agent without the opponent model as the baseline while the right column shows the results of the agent with the DDPG-DASA model. The DDPG-DASA agent converges to the equilibrium faster than without the opponent model. The learning curve of the Q function shown in Figure 6.4 confirms the faster convergence. The result suggests benefits for the bidding companies who newly join the auctions.

We need to note that, the budget was set by referring to the original market price in the iPinYou bidding log. However, the new market price generated by the agents are different and lower. At the beginning of the campaign lifetime, without any information of the market, the learning agent converges to a steady but sub-optimal strategy. However, if the learning agent infer its opponents' strategies quickly and the opponents have fixed strategies, the DDPG-DASA model facilities the bidder to converge to a more dominant strategy which obtains more clicks in the market. If the other agents adopt learning process into their strategies which evolves the bid distribution, the challenge would be to show the asynchronous best response from all the agents and converge to the MFE which is shown in the next section.

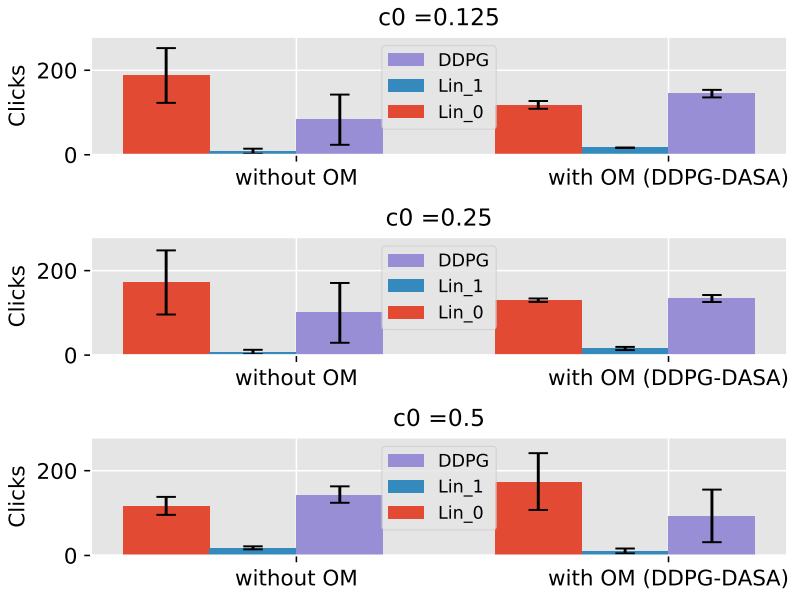


Figure 6.2 The number of clicks won by every bidder of Campaign 2259, where Lin_* refers to the linear bidders.

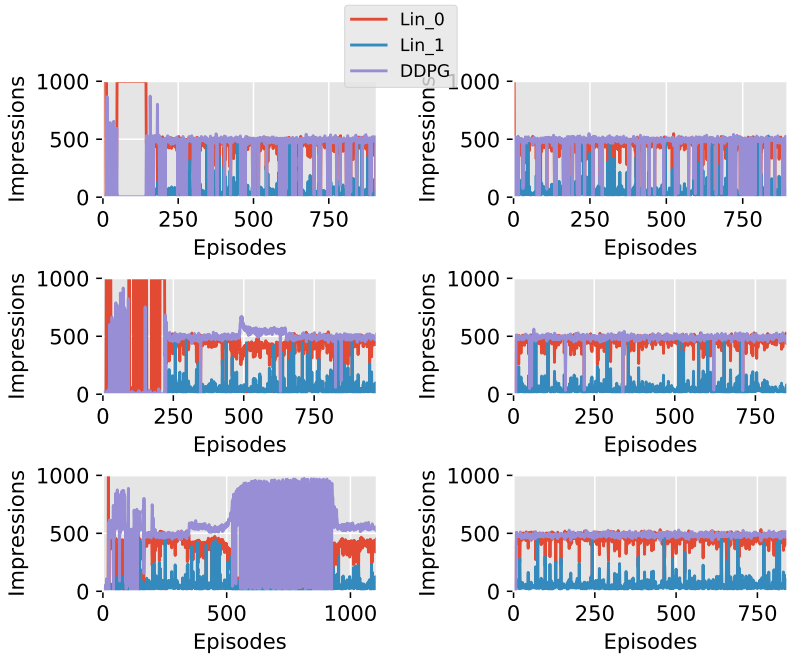


Figure 6.3 The number of impressions won by every bidder of Campaign 2259.

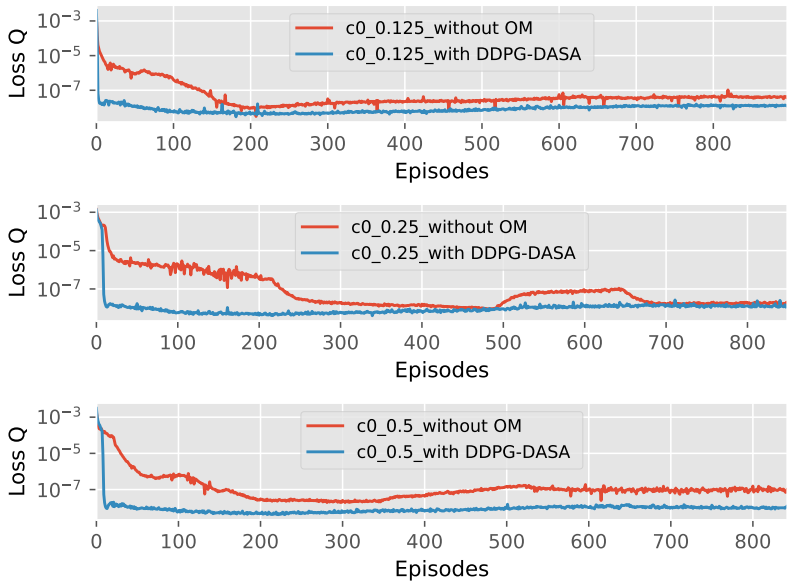


Figure 6.4 Learning curves over Campaign 2259 under different budget settings.

6.4.3 Multi-Agent Game

In this section, the experiment is extended to have multiple learning agents in the same environment. Assume the number of agent $N = 3$, same as in the previous section, the three agents have the same budget setting for every 1000 auctions as one episode. As is shown in the first row in Figure 6.5, the three agents start with bidding by only learning from their own reward without referring to other bidders' behaviour. After 200 episodes, the game converges to the equilibrium where the number of impressions won by each agent is roughly evenly distributed. We continue the experiment by introducing a random market price distribution as the opponent model for each agent. For each auction, the market price distribution is sampled from a uniform distribution. The second row in Figure 6.5 shows that the random opponent model increased the variance of the number of winning impressions for each agent and some agent may converge to a dominating strategy. With the well trained opponent model, we take the bidding log generated by the first game and trained a market model separately for each agent based on the set of impressions they won. With the information of the market, we reset the game for the training set, as is shown in the third row in Figure 6.5. The agents converge to the optimal strategies within 100 episodes which is 50% less than the results in the first row. We further test the model on the test set, which shows the model is generalized well and the equilibrium is reached.

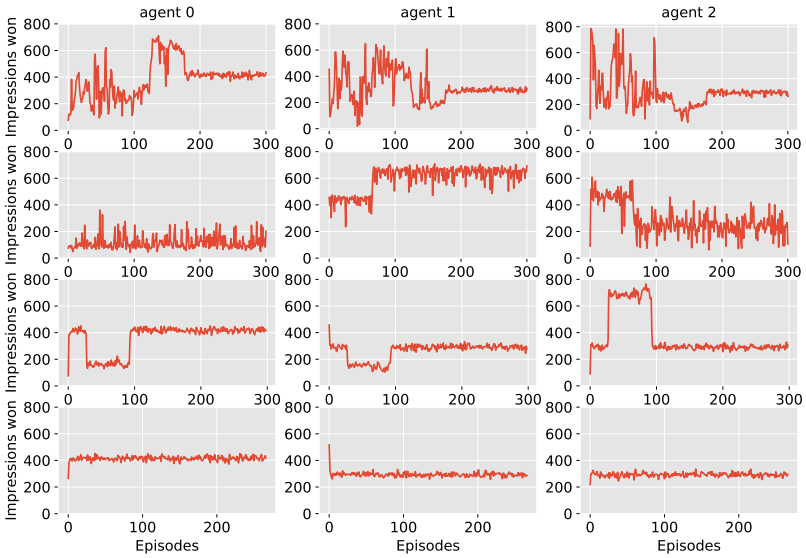


Figure 6.5 Three DDPG agents bidding game. Row 1: DDPG without opponent model. Row 2: DDPG with random market model. Row 3: DDPG-DASA model on the training set Row 4: DDPG-DASA model on test set.

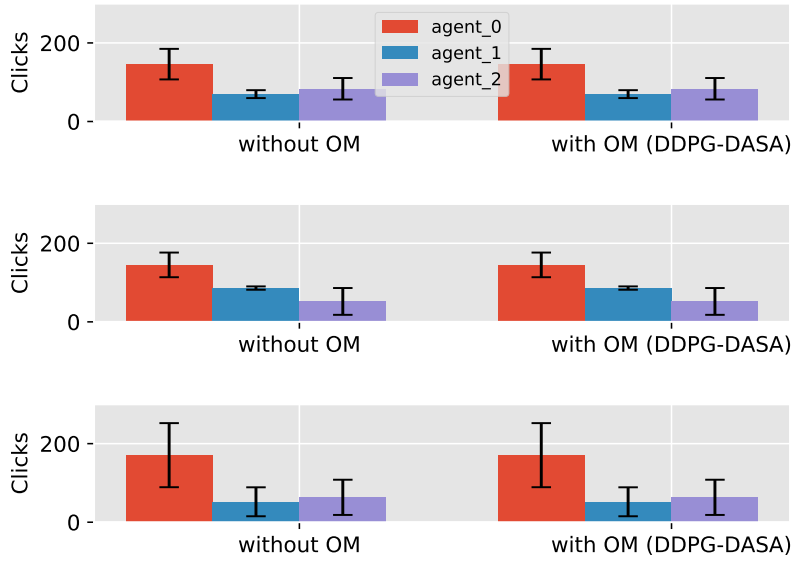


Figure 6.6 The number of clicks each learnign agent gets in the multi-agent secenario.

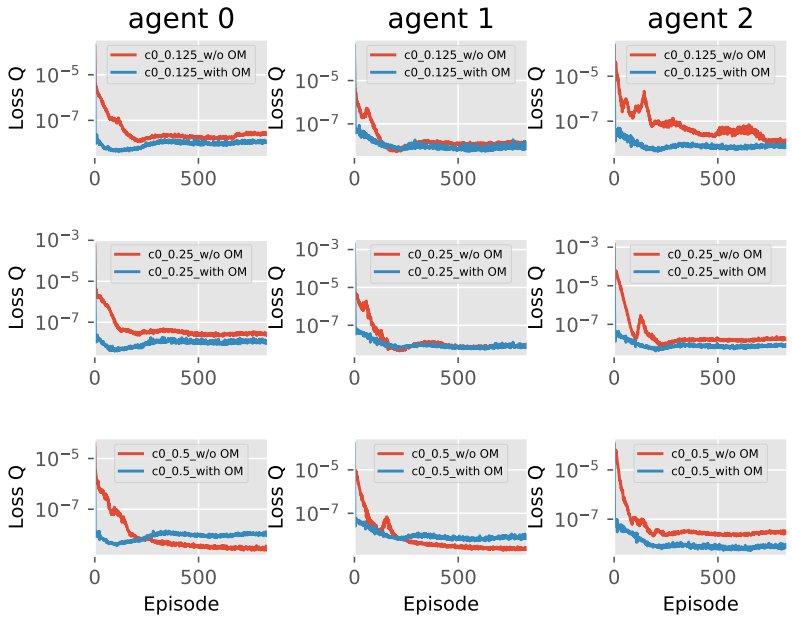


Figure 6.7 Learning curves of DDPG-DASA model for each learning agent.

Figure 6.6 shows the number of clicks each agent won, which stays roughly the same. It demonstrates that since the reward function is not changed, once the equilibrium is reached, the multiple learning agents do not gain more profits. The benefits introduced by using the DDPG-DASA model is the fast convergence speed. In Figure 6.7, it shows the DDPG-DASA model for each learning agent converges generally faster than the DDPG model.

6.5 Summary

In this chapter, we proposed a general opponent aware bidding algorithm with no prior assumptions on the opponents bidding distribution. To the best of our knowledge, it is the first experimental implementation in the real-time bidding domain to infer the partially observable opponents in the policy learning process. We proposed a deep attentive survival model as the impression level opponent model. The multi-agent bidding simulations show the benefits of improved convergence rates for the DDPG model across all budgets which augmented with an opponent model. For the future work, instead of using a pre-trained model, we will investigate the adaptive training for both the opponent model and the reward function in the multi-agent bidding game.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this dissertation, the major research problem is to design an optimal bidding strategy competing in a RTB environment for the display advertising. The research scope is defined to be from the perspective of advertisers and the type of real-time auctions is the second-price auctions. In the following, we summarize our contributions from three aspects of the research topic.

7.1.1 User Response Prediction

For the advertisers, the CVR is an important metric to evaluate the performance of their ad campaigns. It directly shows the true intentions of the users towards the content of the ad. Therefore, CVR, as an important indicator of a high value impression, serves as a part of the optimization goal for the bidding decision making process. Either over-estimating or under-estimating the CVR leads to inefficient budget spending and lower returns for the advertisers.

Conventional CVR models take only the bid request features as input. In this dissertation, we formulate the purchase time as a time series and construct two time series features to enhance the prediction performance. The first feature is the purchase probability at any given time

provided by a self-exciting model. It captures the purchasing intention cascade and the time decaying effect. The second feature is an indicator showing the historical CVR changing trend which serves as the short-term memory. By integrating these two time series related features, the CVR prediction model outperforms the state-of-the-art models which only use the basic feature set.

7.1.2 Cost Estimation

The competition in the RTB market decides the cost of each ad impression. The paying price influences how fast or slow the budget is consumed for the advertisers and also has an impact on their total profit. The ideal scenario is to win the high value impressions and pay less for the high returns. Due to the great amount of bidders in the market and their unknown strategies, the market price fluctuates heavily over time. Within each second, the advertisers can receive over hundreds or thousands of bid requests which makes predicting the price pattern a difficult task.

To solve the above problems, we provide a price-based market prediction model. We aggregate the market prices over time into window slices and formulate the average price per window slice as a time series. Unlike the individual prices, the aggregated prices show clear temporal patterns. A multi-variate LSTM based RNN is proposed to take the aggregated features in the bid requests and the historical price as inputs and to predict the average market price for the next window slice. The proposed model captures the temporal patterns both in the price and in the feature space. It outperforms the state-of-the-art univariate time series models in terms of the mean squared error in the price prediction.

The price-based model demonstrates high competence of estimating an average price for a group of requests as a reference. However, it lacks the generality of directly providing an estimate of the winning probability for a single auction. Therefore, we take the winning price as a stochastic variable and predict its distribution which is also called the *bid landscape*. We propose a deep attentive survival analysis model to address the right-censored problem and to transform the input features into the price distribution domain. The proposed model releases the recurrent constraints in the RNN based network and learns the importance of the feature values in parallel through the attention mechanism. It provides an end-to-end solution for predicting the market price distribution. The prediction performance greatly surpasses both the basic statistical models and the deep learning based models. In this dissertation, the

proposed DASA model is chosen to be the opponent model facilitating the bidding strategy design in the multi-agent bidding scenario.

7.1.3 Optimal Bidding Strategies

To compete in the RTB environment, from the advertiser's perspective, the goal is to maximize its target interests (e.g. clicks, conversions, or profits) by sequentially making bidding decisions within a certain budget constraint. From a single bidder's perspective, we modeled the interactions between the bidder and the bidding market as a model-based reinforcement learning framework. The proposed model does not reply on any pre-defined model form for the bidding strategy optimization. We introduce the conditional market price distribution into the reward function and the cost function to provide impression level bidding decisions. The proposed model outperforms the state-of-the-art bidding functions with low budget constraints.

The previous model is based on the assumption that the bidding market is static. However, in practice, in each RTB auction, each bidder is interacting with an unknown number of bidders which may have different strategies or learn to adapt their strategies. Such phenomenon should not be neglected, even though it is infeasible to infer other bidder's behavior individually. We formalize the problem as a multi-agent learning scenario in RTB. Each bidder learns to explore and to optimize its own strategy. Inspired by the mean-field theory, the uncertainty of stochastic behaviors from an unknown number of bidders is averaged out by introducing an opponent model. With an inference of other bidder's behavior, we demonstrate faster convergence for the strategy learning process, so that all the bidders reach the Nash equilibrium in which not a single bidder can dominate the bidding game.

7.2 Future Work

In this section, we present potential future work for the above research topics. Firstly, the purchase events are sparse in general and greatly depend on the type of the products as well. For instance, the conversion rate of a car advertisement can be genetically lower than the conversion rate of groceries. Secondly, new advertisements or new customers without any historical data suffer from the *cold start* problem. In the domain of recommender

systems, a widely applied approach is collaborative filtering [91, 125] which associates similar items and similar users based on the historical records. However, it may still have sparse user-item pair in the log. To collect and ensemble more information of users and products becomes the key to providing better predictions. Recent studies have been focused on building *Knowledge Graphs* (KGs) which connect products, user profiles, and their online behaviors to construct deep and comprehensive understandings among them. Examples of large scale commertial KGs are: Google Knowledge Graph[1] and Microsoft Bing Satori Entity Database[2]. A few recent works of using KGs based networks for recommender systems can be found in [137, 138]. To the best of our knowledge, such methods have not been studied in the conversion rate prediction in the RTB domain.

Lastly, in our single agent bidding experiments, comparing with other bidding functions, at the end of each episode, the proposed CMDP model still has unspent budget. It achieves low cost per click but results in inefficient budget usage. Another limitation is that the proposed model learns to react to a static market. Therefore, when it competes with other bidding functions and the market changes, the proposed strategy fails to adapt to the new scenario. In the multi-agent scenario, the assumption is that all the bidders adapt towards the Nash equilibrium. This type of opponent modeling is based only on the opponent's historical behaviors without any recursive reasoning.

However, as humans, it is natural to reason about other people's behaviours and plan our own moves. Wen et al. [143] consider the level-1 recursive reasoning for each player in multi-agent games. Each player plans its own strategy based on an estimation of how other players react to its possible future actions. The proposed model in this work generalizes the requirement of defining a certain type of opponent. However, this work is based on fully observable opponents. It is promising to study the level-1 recursive reasoning for partially observable opponents in the RTB domain.

[1] https://developers.google.com/knowledge-graph/
[2] https://searchengineland.com/library/bing/bing-satori

References

[1] Adikari, S. and Dutta, K. (2015). Real time bidding in online digital advertisement. In *New Horizons in Design Science: Broadening the Research Agenda*. Springer.

[2] Agarwal, D., Broder, A. Z., Chakrabarti, D., Diklic, D., Josifovski, V., and Sayyadian, M. (2007). Estimating rates of rare events at multiple resolutions. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

[3] Agarwal, D., Chen, B.-C., and Elango, P. (2009). Spatio-temporal models for estimating click-through rate. In *Proceedings of the 18th international conference on World wide web*, pages 21–30. ACM.

[4] Agarwal, D., Ghosh, S., Wei, K., and You, S. (2014). Budget pacing for targeted online advertisements at linkedin. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1613–1619. ACM.

[5] Aggarwal, C. C. (2015). *Data mining: the textbook*. Springer.

[6] Ahuja, K., Zame, W., and van der Schaar, M. (2017). Dpscreen: Dynamic personalized screening. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 1321–1332. Curran Associates, Inc.

[7] Aït-Sahalia, Y., Cacho-Diaz, J., and Laeven, R. J. (2015). Modeling financial contagion using mutually exciting jump processes. *Journal of Financial Economics*, 117(3):585–606.

[8] Akaike, H. (1998). Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*, pages 199–213. Springer.

[9] Altman, E. (1999). *Constrained Markov decision processes*. CRC Press.

[10] Amin, K., Kearns, M., Key, P., and Schwaighofer, A. (2012). Budget optimization for sponsored search: Censored learning in mdps. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*. AUAI Press.

[11] Andrews, E. (2014). Multi-armed bandits and reinforcement learning approaches to targeted advertising. In *Seminar: Reinforcement Learning and Its Applications*. Spring.

[12] Applebaum, D. (2008). *Probability and Information: An Integrated Approach*. Cambridge University Press, 2 edition.

[13] Athey, S. and Nekipelov, D. (2010). A structural model of sponsored search advertising auctions. In *Sixth ad auctions workshop*, volume 15.

[14] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[15] Baccot, B., Grigoras, R., and Charvillat, V. (2010). Reinforcement learning for online optimization of banner format and delivery. *Online Multimedia Advertising: Techniques and Technologies: Techniques and Technologies*, page 13.

[16] Baddeley, A., Turner, R., Møller, J., and Hazelton, M. (2005). Residual analysis for spatial point processes (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(5):617–666.

[17] Barber, D. (2012). *Bayesian reasoning and machine learning*. Cambridge University Press.

[18] Barbieri, N., Silvestri, F., and Lalmas, M. (2016). Improving post-click user engagement on native ads via survival analysis. In *Proceedings of the 25th International Conference on World Wide Web*, pages 761–770. International World Wide Web Conferences Steering Committee.

[19] Bartholomew, D. J. (1971). Time series analysis forecasting and control. *Journal of the Operational Research Society*, 22(2):199–201.

[20] Bateni, M., Feldman, J., Mirrokni, V., and Wong, S. C.-w. (2014). Multiplicative bidding in online advertising. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 715–732. ACM.

[21] Bender, R., Augustin, T., and Blettner, M. (2005). Generating survival times to simulate cox proportional hazards models. *Statistics in medicine*, 24(11):1713–1723.

[22] Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. (2012). Modeling temporal dependencies in high-dimensional sequences: application to polyphonic music generation and transcription. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1881–1888. Omnipress.

[23] Cai, H., Ren, K., Zhag, W., Malialis, K., and Wang, J. (2017). Real-time bidding by reinforcement learning in display advertising. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining (WSDM)*.

[24] Cao, Q., Shen, H., Cen, K., Ouyang, W., and Cheng, X. (2017). Deephawkes: Bridging the gap between prediction and understanding of information cascades. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1149–1158. ACM.

[25] Chakrabarti, D., Agarwal, D., and Josifovski, V. (2008). Contextual advertising by combining relevance with click feedback. In *Proceedings of the 17th international conference on World Wide Web (WWW)*.

[26] Chang, I., Tiao, G. C., and Chen, C. (1988). Estimation of time series parameters in the presence of outliers. *Technometrics*, 30(2):193–204.

[27] Chapelle, O. (2014). Modeling delayed feedback in display advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1097–1105. ACM.

[28] Chen, D., Chen, W., Wang, H., Chen, Z., and Yang, Q. (2012). Beyond ten blue links: enabling user click modeling in federated web search. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 463–472. ACM.

[29] Chen, J., Sun, B., Li, H., Lu, H., and Hua, X.-S. (2016). Deep ctr prediction in display advertising. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 811–820. ACM.

[30] Chen, Y., Berkhin, P., Anderson, B., and Devanur, N. R. (2011). Real-time bidding algorithms for performance-based display ad allocation. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1307–1315. ACM.

[31] Chen, Y., Pavlov, D., and Canny, J. F. (2009). Large-scale behavioral targeting. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

[32] Cheng, H. and Cantú-Paz, E. (2010). Personalized click prediction in sponsored search. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 351–360. ACM.

[33] Cheng, H., Zwol, R. v., Azimi, J., Manavoglu, E., Zhang, R., Zhou, Y., and Navalpakkam, V. (2012). Multimedia features for click prediction of new ads in display advertising. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 777–785. ACM.

[34] Choi, S. P., Yeung, D.-Y., and Zhang, N. L. (2000). An environment model for nonstationary reinforcement learning. In *Advances in neural information processing systems*, pages 987–993.

[35] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

[36] Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202.

[37] Cui, Y., Zhang, R., Li, W., and Mao, J. (2011). Bid landscape forecasting in online ad exchange marketplace. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*.

[38] Davis, H. (2006). *Google advertising tools: Cashing in with AdSense, AdWords, and the Google APIs*. " O'Reilly Media, Inc.".

[39] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[40] Ding, W., Qin, T., Zhang, X.-D., and Liu, T.-Y. (2013). Multi-armed bandit with budget constraint and variable costs. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

[41] Du, M., Sassioui, R., Varisteas, G., Brorsson, M., Cherkaoui, O., and State, R. (2017). Improving real-time bidding using a constrained markov decision process. In *International Conference on Advanced Data Mining and Applications*. Springer.

[42] Edelman, B., Ostrovsky, M., and Schwarz, M. (2007). Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American economic review*, 97(1):242–259.

[43] Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874.

[44] Fernandez-Tapia, J. (2019). An analytical solution to the budget-pacing problem in programmatic advertising. *Journal of Information and Optimization Sciences*, 40(3):603–614.

[45] Gamerman, D. and West, M. (1987). An application of dynamic survival models in unemployment studies. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 36(2-3):269–274.

[46] Geibel, P. (2006). Reinforcement learning for mdps with constraints. In *European Conference on Machine Learning*.

[47] Ghosh, A., Rubinstein, B. I., Vassilvitskii, S., and Zinkevich, M. (2009). Adaptive bidding for display advertising. In *Proceedings of the 18th international conference on World wide web*, pages 251–260. ACM.

[48] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[49] Graepel, T., Candela, J. Q., Borchert, T., and Herbrich, R. (2010). Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *Proceedings of the 27th International Conference on Machine Learning ICML*.

[50] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE.

[51] Gummadi, R., Key, P., and Proutiere, A. (2012). Repeated auctions under budget constraints: Optimal bidding strategies and equilibria. In *the Eighth Ad Auction Workshop*.

[52] Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. (2017). Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*.

[53] Hamilton, J. D. (1994). *Time series analysis*, volume 2. Princeton university press Princeton.

[54] Hawkes, A. G. (1971). Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90.

[55] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[56] He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., Shi, Y., Atallah, A., Herbrich, R., Bowers, S., et al. (2014). 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.

[57] Hernandez-Leal, P., Kaisers, M., Baarslag, T., and de Cote, E. M. (2017). A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*.

[58] Hernandez-Leal, P., Kartal, B., and Taylor, M. E. (2018). Is multiagent deep reinforcement learning the answer or the question? a brief survey. *arXiv preprint arXiv:1810.05587*.

[59] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8).

[60] Hoelzel, M. and Ballvé, M. (2015). *The Programmatic-Advertising Report: Mobile, video, and real-time bidding drive growth in programmatic*. BI Intelligence.

[61] Hu, J. and Wellman, M. P. (2003). Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069.

[62] Huang, M., Malhamé, R. P., Caines, P. E., et al. (2006). Large population stochastic dynamic games: closed-loop mckean-vlasov systems and the nash certainty equivalence principle. *Communications in Information & Systems*, 6(3):221–252.

[63] Iyer, K., Johari, R., and Sundararajan, M. (2014). Mean field equilibria of dynamic auctions with learning. *Management Science*, 60(12).

[64] Jansen, B. J. and Mullen, T. (2008). Sponsored search: an overview of the concept, history, and technology. *International Journal of Electronic Business*, 6(2):114–131.

[65] Jauvion, G. and Grislain, N. (2018). Optimal allocation of real-time-bidding and direct campaigns. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 416–424. ACM.

[66] Jiang, C. (2015). *Online advertisements and multi-armed bandits*. PhD thesis, University of Illinois at Urbana-Champaign.

[67] Jiang, Z. (2016). Research on ctr prediction for contextual advertising based on deep architecture model. *Journal of Control Engineering and Applied Informatics*, 18(1):11–19.

[68] Jin, J., Song, C., Li, H., Gai, K., Wang, J., and Zhang, W. (2018). Real-time bidding with multi-agent reinforcement learning in display advertising. *arXiv preprint arXiv:1802.09756*.

[69] Joaquin, F. (2015). Real-time bidding rules of thumb: analytically optimizing the programmatic buying of ad-inventory.

[70] Johnson, G. and Lewis, R. A. (2015). Cost per incremental action: Efficient pricing of advertising.

[71] Juan, Y., Lefortier, D., and Chapelle, O. (2017). Field-aware factorization machines in a real-world online advertising system. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 680–688. International World Wide Web Conferences Steering Committee.

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

[73] Kaplan, E. L. and Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, 53(282):457–481.

[74] Kingman, J. F. C. (1993). *Poisson processes*. Wiley Online Library.

[75] Koene, R. J., Prizment, A. E., Blaes, A., and Konety, S. H. (2016). Shared risk factors in cardiovascular disease and cancer. *Circulation*, 133(11):1104–1114.

[76] Krishna, V. (2009). *Auction theory*. Academic press.

[77] Lai, H.-C., Shih, W.-Y., Huang, J.-L., and Chen, Y.-C. (2016). Predicting traffic of online advertising in real-time bidding systems from perspective of demand-side platforms. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3491–3498. IEEE.

[78] Lange, S., Gabel, T., and Riedmiller, M. (2012). Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer.

[79] Laptev, N., Yosinski, J., Li, L. E., and Smyl, S. (2017). Time-series extreme event forecasting with neural networks at uber. In *International Conference on Machine Learning*.

[80] Laub, P. J., Taimre, T., and Pollett, P. K. (2015). Hawkes processes. Papers, arXiv.org.

[81] Lee, C., Zame, W. R., Yoon, J., and van der Schaar, M. (2018). Deephit: A deep learning approach to survival analysis with competing risks. AAAI.

[82] Lee, E. W., Wei, L., Amato, D. A., and Leurgans, S. (1992). Cox-type regression analysis for large numbers of small groups of correlated failure time observations. In *Survival analysis: state of the art*, pages 237–247. Springer.

[83] Lee, K.-C., Jalali, A., and Dasdan, A. (2013). Real time bid optimization with smooth budget delivery in online advertising. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*.

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

[85] Lewis, E., Mohler, G., Brantingham, P. J., and Bertozzi, A. L. (2012). Self-exciting point process models of civilian deaths in iraq. *Security Journal*, 25(3):244–264.

[86] Li, S., Gao, X., Bao, W., and Chen, G. (2017). Fm-hawkes: A hawkes process based approach for modeling online activity correlations. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1119–1128. ACM.

[87] Li, X. and Guan, D. (2014). Programmatic buying bidding strategies with win rate and winning price estimation in real time mobile advertising. In *Advances in Knowledge Discovery and Data Mining*. Springer.

[88] Liao, H., Peng, L., Liu, Z., and Shen, X. (2014). ipinyou global rtb bidding algorithm competition dataset. In *Proceedings of 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1–6. ACM.

[89] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

[90] Lin, C.-C., Chuang, K.-T., Wu, W. C.-H., and Chen, M.-S. (2016). 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.

[91] Linden, G., Smith, B., and York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, (1):76–80.

[92] Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*. Elsevier.

[93] Maehara, T., Narita, A., Baba, J., and Kawabata, T. (2018). Optimal bidding strategy for brand advertising. In *IJCAI*, pages 424–432.

[94] Mahdian, M. and Tomak, K. (2007). Pay-per-action model for online advertising. In *Proceedings of the 1st International Workshop on Data Mining and Audience Intelligence for Advertising*, ADKDD '07, pages 1–6, New York, NY, USA. ACM.

[95] Manzoor, E. and Akoglu, L. (2017). Rush!: Targeted time-limited coupons via purchase forecasts. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1923–1931. ACM.

[96] McMahan, H. B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., Golovin, D., et al. (2013). Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

[97] Menon, A. K., Chitrapura, K.-P., Garg, S., Agarwal, D., and Kota, N. (2011). Response prediction using collaborative filtering with hierarchies and side-information. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 141–149. ACM.

[98] Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

[99] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[100] Miller Jr, R. G. (2011). *Survival analysis*, volume 66. John Wiley & Sons.

[101] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

[102] Myerson, R. B. (2013). *Game theory*. Harvard university press.

[103] Nash, J. F. et al. (1950). Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49.

[104] Newman, M. E. (2005). Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351.

[105] Nguyen, T. T., Nguyen, N. D., and Nahavandi, S. (2018). Deep reinforcement learning for multi-agent systems: a review of challenges, solutions and applications. *arXiv preprint arXiv:1812.11794*.

[106] Oentaryo, R., Lim, E.-P., Finegold, M., Lo, D., Zhu, F., Phua, C., Cheu, E.-Y., Yap, G.-E., Sim, K., Nguyen, M. N., et al. (2014a). Detecting click fraud in online advertising: a data mining approach. *The Journal of Machine Learning Research*, 15(1):99–140.

[107] Oentaryo, R. J., Lim, E.-P., Low, J.-W., Lo, D., and Finegold, M. (2014b). Predicting response in mobile advertising with hierarchical importance-aware factorization machine. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 123–132. ACM.

[108] Ogata, Y. (1988). Statistical models for earthquake occurrences and residual analysis for point processes. *Journal of the American Statistical association*, 83(401):9–27.

[109] Pan, J., Xu, J., Ruiz, A. L., Zhao, W., Pan, S., Sun, Y., and Lu, Q. (2018). Field-weighted factorization machines for click-through rate prediction in display advertising. In *Proceedings of the 2018 World Wide Web Conference*, pages 1349–1357. International World Wide Web Conferences Steering Committee.

[110] Peng, R. D. (2002). Multi-dimensional point process models in r. *Department of Statistics, UCLA*.

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

[112] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

[113] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI.

[114] Rapoport, A., Chammah, A. M., and Orwant, C. J. (1965). *Prisoner's dilemma: A study in conflict and cooperation*, volume 165. University of Michigan press.

[115] Rasmussen, J. G. (2018). Lecture notes: Temporal point processes and the conditional intensity function.

[116] Reisinger, J. and Driscoll, M. (2010). Pricing externalities in real-time bidding markets. In *NIPS Workshop: Machine Learning in Online Advertising*.

[117] Ren, K., Qin, J., Zheng, L., Yang, Z., Zhang, W., Qiu, L., and Yu, Y. (2019a). Deep recurrent survival analysis.

[118] Ren, K., Qin, J., Zheng, L., Yang, Z., Zhang, W., and Yu, Y. (2019b). Deep landscape forecasting for real-time bidding advertising. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 363–372. ACM.

[119] Ren, K., Zhang, W., Chang, K., Rong, Y., Yu, Y., and Wang, J. (2017). Bidding machine: Learning to bid for directly optimizing profits in display advertising. *IEEE Transactions on Knowledge and Data Engineering*, 30.

[120] Ren, K., Zhang, W., Rong, Y., Zhang, H., Yu, Y., and Wang, J. (2016). User response learning for directly optimizing campaign performance in display advertising. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 679–688. ACM.

[121] Rendle, S. (2010). Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000. IEEE.

[122] Richardson, M., Dominowska, E., and Ragno, R. (2007). Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web (WWW)*.

[123] Roondiwala, M., Patel, H., and Varma, S. (2017). Predicting stock prices using lstm. *International Journal of Science and Research (IJSR)*, 6(4):1754–1756.

[124] Rosales, R., Cheng, H., and Manavoglu, E. (2012). Post-click conversion modeling and analysis for non-guaranteed delivery display advertising. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM.

[125] Sarwar, B. M., Karypis, G., Konstan, J. A., Riedl, J., et al. (2001). Item-based collaborative filtering recommendation algorithms. *Www*, 1:285–295.

[126] Schmitter, T. and Rosen, J. (2005). Contextual advertising system. US Patent App. 10/836,820.

[127] Schwartz, E. M., Bradlow, E., and Fader, P. (2015). Customer acquisition via display advertising using multi-armed bandit experiments. *Ross School of Business Paper*.

[128] Shapley, L. S. (1953). Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100.

[129] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354.

[130] Stone-Gross, B., Stevens, R., Zarras, A., Kemmerer, R., Kruegel, C., and Vigna, G. (2011). Understanding fraudulent activities in online ad exchanges. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 279–294. ACM.

[131] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

[132] Tallis, M. and Yadav, P. (2018). Reacting to variations in product demand: An application for conversion rate (cr) prediction in sponsored search. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1856–1864. IEEE.

[133] Trofimov, I., Kornetova, A., and Topinskiy, V. (2012). 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*.

[134] Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

[135] Vasile, F. and Lefortier, D. (2016). Cost-sensitive learning for bidding in online advertising auctions. *CoRR*.

[136] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*.

[137] Wang, H., Zhang, F., Zhao, M., Li, W., Xie, X., and Guo, M. (2019a). Multi-task feature learning for knowledge graph enhanced recommendation. In *WWW*.

[138] Wang, H., Zhao, M., Xie, X., Li, W., and Guo, M. (2019b). Knowledge graph convolutional networks for recommender systems. In *The World Wide Web Conference*, WWW '19, pages 3307–3313, New York, NY, USA. ACM.

[139] Wang, J.-H. and Leu, J.-Y. (1996). Stock market trend prediction using arima-based neural networks. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 4, pages 2160–2165. IEEE.

[140] Wang, P., Li, Y., and Reddy, C. K. (2019c). Machine learning for survival analysis: A survey. *ACM Computing Surveys (CSUR)*, 51(6):110.

[141] Wang, R., Fu, B., Fu, G., and Wang, M. (2017). Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, page 12. ACM.

[142] Wang, Y., Ren, K., Zhang, W., Wang, J., and Yu, Y. (2016). Functional bid landscape forecasting for display advertising. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*.

[143] Wen, Y., Yang, Y., Luo, R., Wang, J., and Pan, W. (2019). Probabilistic recursive reasoning for multi-agent reinforcement learning. In *International Conference on Learning Representations*.

[144] Wu, D., Chen, X., Yang, X., Wang, H., Tan, Q., Zhang, X., Xu, J., and Gai, K. (2018a). Budget constrained bidding by model-free reinforcement learning in display advertising. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1443–1451. ACM.

[145] Wu, W., Yeh, M.-Y., and Chen, M.-S. (2018b). Deep censored learning of the winning price in the real time bidding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2526–2535. ACM.

[146] Wu, W. C.-H., Yeh, M.-Y., and Chen, M.-S. (2015). 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.

[147] Xu, H., Gao, B., Yang, D., and Liu, T.-Y. (2013). Predicting advertiser bidding behaviors in sponsored search by rationality modeling. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1433–1444. ACM.

[148] Xu, J., Lee, K.-c., Li, W., Qi, H., and Lu, Q. (2015). Smart pacing for effective online ad campaign optimization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

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

[150] Xu, L., Duan, J. A., and Whinston, A. (2014). Path to purchase: A mutually exciting point process model for online advertising and conversion. *Management Science*, 60(6):1392–1412.

[151] Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., and J.Wang (2018). Mean field multi-agent reinforcement learning. *arXiv preprint arXiv:1802.05438*.

[152] Yuan, S., Wang, J., Chen, B., Mason, P., and Seljan, S. (2014). An empirical study of reserve price optimisation in real-time bidding. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1897–1906. ACM.

[153] Yuan, S., Wang, J., and Zhao, X. (2013). Real-time bidding for online advertising: measurement and analysis. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, page 3. ACM.

[154] Zhai, S., Chang, K.-h., Zhang, R., and Zhang, Z. (2016). Attention based recurrent neural networks for online advertising. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 141–142. International World Wide Web Conferences Steering Committee.

[155] Zhang, H., Zhang, W., Rong, Y., Ren, K., Li, W., and Wang, J. (2017). Managing risk of bidding in display advertising. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 581–590. ACM.

[156] Zhang, L. and Guan, Y. (2008). Detecting click fraud in pay-per-click streams of online advertising networks. In *Distributed Computing Systems, 2008. ICDCS'08. The 28th International Conference on*, pages 77–84. IEEE.

[157] Zhang, W., Chen, L., and Wang, J. (2016a). Implicit look-alike modelling in display ads:transfer collaborative filtering to ctr estimation. In *Proceedings of the 38th European Conference on Information Retrieval*.

[158] Zhang, W., Du, T., and Wang, J. (2016b). Deep learning over multi-field categorical data. In *European Conference on Information Retrieval (ECIR)*.

[159] Zhang, W. and Wang, J. (2015). Statistical arbitrage mining for display advertising. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1465–1474. ACM.

[160] Zhang, W., Yuan, S., and Wang, J. (2014a). Optimal real-time bidding for display advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*.

[161] Zhang, W., Yuan, S., and Wang, J. (2014b). Real-time bidding benchmarking with ipinyou dataset. *CoRR*.

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

[163] Zhang, W., Zhang, Y., Gao, B., Yu, Y., Yuan, X., and Liu, T.-Y. (2012). Joint optimization of bid and budget allocation in sponsored search. In *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1177–1185. ACM.

[164] Zhang, W., Zhou, T., Wang, J., and Xu, J. (2016c). Bid-aware gradient descent for unbiased learning with censored data in display advertising. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

[165] Zhang, Y., Dai, H., Xu, C., Feng, J., Wang, T., Bian, J., Wang, B., and Liu, T. (2014d). Sequential click prediction for sponsored search with recurrent neural networks. *CoRR*, abs/1404.5772.

[166] Zhao, J., Qiu, G., Guan, Z., Zhao, W., and He, X. (2018). Deep reinforcement learning for sponsored search real-time bidding. *arXiv preprint arXiv:1803.00259*.

[167] Zhao, Q., Erdogdu, M. A., He, H. Y., Rajaraman, A., and Leskovec, J. (2015). Seismic: A self-exciting point process model for predicting tweet popularity. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1513–1522. ACM.

[168] Zhou, G., Zhu, X., Song, C., Fan, Y., Zhu, H., Ma, X., Yan, Y., Jin, J., Li, H., and Gai, K. (2018). Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1059–1068. ACM.

[169] Zhu, W.-Y., Shih, W.-Y., Lee, Y.-H., Peng, W.-C., and Huang, J.-L. (2017). A gamma-based regression for winning price estimation in real-time bidding advertising. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1610–1619. IEEE.