Ad Campaign Optimization - A Targeted Contextual Bandit Approach

Master Thesis
Ana Dimitrova
November 7, 2018

Advisors: Prof. Dr. T. Hofmann, Dr. H. Vanchinathan
Department of Computer Science, ETH Zürich
Abstract

Internet advertising is a fast growing business that has proven to be fundamental for the modern digital economy. There has been a significant effort to find the best match between a given user, a given context and a list of available ads because it provides the highest possible revenue.

The main stakeholders in online advertising are the web-publishers, who integrate ad campaigns alongside their online content, and the advertisers, who provide adverts to be displayed beside the publisher’s content. Both sides aim to achieve the highest possible user engagement while maximizing their long term profits. Often, as part of the contractual agreement between the two parties, the publisher guarantees a minimal impression count. When that agreed minimum impression count is not met, a monetary penalty is applied.

This master thesis addresses the challenge of identifying the most appropriate ad unit to display alongside a given piece of web-based content at the best time for individual users within the contract established between a publisher and an advertiser.
Contents

List of Figures

1 Introduction
  1.1 Online Advertising
     1.1.1 Advertiser Perspective
     1.1.2 Publisher Perspective
  1.2 Behavioural Targeting
  1.3 Optimizing Ad Campaigns
  1.4 Thesis Contributions
  1.5 Thesis Outline

2 Related Work
  2.1 Contextual Multi-Arm Bandits
     2.1.1 Problem Setting
     2.1.2 Algorithms
  2.2 Budgeted Multi-Arm Bandits
  2.3 Evaluation Methodology - Exploration Scavenging

3 Problem Formulation
  3.1 Setup
  3.2 Mathematical Formulation
  3.3 Optimization Criteria
  3.4 Algorithmic Interpretation in Our Case
  3.5 Model Constraints
  3.6 Setting Assumptions and Relaxations

4 Experiment Data Set
  4.1 Data Source and Format
     4.1.1 The Ad Activity Log
## Contents

4.1.2 The User Embeddings ........................................... 16
4.2 Experiment Log Metadata ........................................ 16
4.3 Data Limitations .................................................. 18

5 Single Campaign Investigations .................................. 19
5.1 The Single Campaign Approach ................................. 19
  5.1.1 Experiment Execution Flow ................................ 19
  5.1.2 Contextual Multi-Arm Bandits Algorithms ............... 20
  5.1.3 Evaluation Metrics .......................................... 21
5.2 Experiments with Synthetic Data Sets ....................... 21
  5.2.1 Setup ....................................................... 21
  5.2.2 Results .................................................... 22
5.3 Experiments with Real World Data Set ....................... 23
5.4 Conclusions ....................................................... 24

6 Multi-Campaign Approach ........................................... 27
6.1 The Targeted Bandits Approach ................................ 27
6.2 Experiments with Synthetic Data Sets ....................... 29
  6.2.1 Setup ....................................................... 29
  6.2.2 Results .................................................... 31

7 Experimental Results ............................................... 35
7.1 Setup .......................................................... 35
7.2 Feature Impact .................................................. 35
  7.2.1 Target Budget Split ...................................... 36
  7.2.2 Audience Normalization ................................. 37
  7.2.3 Implementation Specific Add-Ons ....................... 38
7.3 LinUCB vs T_LinUCB ............................................ 38
7.4 Conclusion ....................................................... 41

8 Conclusion .......................................................... 43

Bibliography .......................................................... 45
List of Figures

4.1 Impression count and CTR per ad campaign .......................... 16
4.2 Grouping of users by number of seen campaigns ...................... 17
4.3 Ad clicks vs views for the users that have clicked at least one ad 17

5.1 CTR gains per algorithm type in various synthetic settings ... 22
5.2 Recommendation list size effect on CTR gains and TPR in synthetic setting ................................................................. 22
5.3 CTR and MSE for Different Algorithms with Fixed Recommendation Size: 20% ................................................................. 23
5.4 CTR and TPR Ration with Different Recommendation Sizes ... 24

6.1 Simulation results ................................................................. 32
6.2 Performance of T_LinUCB with various added noise ................. 32
6.3 The effect of the target budget size in both lower and upper bound simulation settings w/o clickers ................................. 33

7.1 Performance of LinUCB vs T_LinUCB (baseline) ..................... 36
7.2 Feature performance .......................................................... 37
7.3 Implementation specific add-ons performance ....................... 38
7.4 LinUCB vs T_LinUCB ............................................................. 39
7.5 CTR Per Campaigns ............................................................. 40
7.6 Impression Distribution Per Campaign ................................. 40
Chapter 1

Introduction

In this chapter we introduce the main topic of the thesis. First, we describe the field of online advertising. Then, we briefly cover prior work in the area of ad contextual recommender systems and budgeting. We continue with the presentation of the main challenges and contributions of the thesis. Finally, we give outline of how the thesis is structured in the remaining chapters.

1.1 Online Advertising

Online advertising is a fast growing business with universal economical impact. In 2016, internet advertising in the United States surpassed cable and broadcast TV in revenue [IAB, 2018]. It is believed that soon it will surpass traditional advertising media worldwide [Yuan et al., 2012]. The popular consensus as to what drives this shift is attributed to the fact that web-based ads can and are often targeted at a filtered set of interested users.

The key stakeholders of online advertising make use of personal and user activity information in order to tailor adverts to their demographic and taste. They aim to maximize the number of click-throughs received for each advert and subsequently the revenue obtained. Moreover, users online activity while observing adverts is recorded and is later used as a feedback for further improvement of the ad recommender system.

1.1.1 Advertiser Perspective

Advertisers use ads to generate demand from potential customers for a product or service. Additionally, ads can be used to improve brand awareness, which is why advertisers pay per ad impression [Yuan et al., 2012]. Therefore, advertisers are trying to display their ads to users that would have
the most interest in them i.e. generate larger revenue by purchasing their products or services.

1.1.2 Publisher Perspective

Web publishers are online platforms for distributing digital content to users. They aim to increase their user engagement by providing appealing information. Online publishers often obtain a significant amount of their revenue through the displaying of ads. That is why it is important to strike a balance between the quality of content and displaying ads.

Revenue Maximization

There are a diverse set of pricing models for ad placement including cost-per-mille (CPM), cost-per-click (CPC), cost-per-action (CPA), etc. [Yuan et al., 2012]. Typically CPC or CPA are much higher than the CPM. This incentivizes the publishers to display an ad to the users that are most likely to interact with it since that will increase their earnings.

This has given rise to a significant technical investment in ad recommendation systems that aims to maximize publishers’ revenue while simultaneously maintaining user engagement.

Contract Fulfillment and Budgeting

Sometimes, as part of the ad campaign contract between an advertiser and a publisher, the latter is obliged to display an agreed minimum number of impressions. In the event that the publisher is unable to fulfill that requirement, a financial penalty often applies. Therefore it is imperative for the publisher to achieve this agreed upon impression count [Yuan et al., 2012]. The lack of truthful knowledge of web traffic a priori poses challenges in estimating the number of available impressions. Moreover, simply displaying the ads in a way that fulfills a contract would not necessarily provide the highest possible revenue.

Throughout the remainder of this thesis the aforementioned minimum guaranteed impression count will be referred to as the target budget.

1.2 Behavioural Targeting

Both advertisers and publishers have the incentive to display ad campaigns to the most suitable set of users. One way to do so is by tailoring the ads to the content displayed on the web page or search terms which are often referred to as contextual advertising. Behaviour targeting on the other hand is centred around the historical activity and actions of specific users. A user can be uniquely identified and tracked on the pages of a publisher’s web
1.3 Optimizing Ad Campaigns

Web-based advertising can easily be viewed as a recommendation system, where a publisher offers a list of pre-selected ads to its users. Historically there has been a large number of recommender systems developed, aiming to provide meaningful recommendations at a targeted individual level. Some examples include recommender systems for movies, music, books, article, and other products. Traditional approaches for solving this problem are collaborative and content-based filtering. They leverage user’s interest as demonstrated by their prior activity and information about the recommended items.

Online advertising, though a recommender system problem, has to adapt to fast paced changing content, constant influx of new users, popularity and temporal changes. These reasons make the existing recommender system solutions sub optimal, giving rise to the reformulation of the problem as a feature-based exploration/exploitation [Langford and Zhang, 2007]. This contextual bandit approach is now considered state of the art solution for ad recommendation systems.

1.4 Thesis Contributions

The main contribution of this thesis is to iteratively learn to maximize the performance of a group of display advertising campaigns each with its own impressions delivery guarantee constraint.

We first investigate algorithms similar to the ones studied in Langford and Zhang [2007], Li et al. [2010], Agrawal and Goyal [2012], Vanchinathan et al. [2014] and Zhou [2015] to the scenario of ad recommendations. Later we integrate them with the concept of budgets introduced formally in Badanidiyuru et al. [2013]. In our work we are not interested in the traditional maximal budget limits, ones that should not be exceeded, but instead with minimal target budgets, ones that have to be reached. This is an important distinction which aims to provide a guarantee of ad recommender systems that they would reach their contract agreements while simultaneously maximizing their revenue.

We performed an offline evaluation of our findings on both simulated data.
1. Introduction

and real ad logs provided by 1PlusX \(^1\). The assessment is performed in two stages. First, all contextual multi-arm bandits algorithms are compared on single campaign data in order to find the most suitable algorithm, given the user embeddings and logs at hand. Later, we combine the best performing algorithm with our concept of targeted budgets and explore its performance on multiple simultaneously running campaigns.

1.5 Thesis Outline

The thesis will be structured in the following way. In chapter 2 we present prior relevant research. Then in chapter 3 we provide a problem formulation for ad recommender systems in combination with applied contract constraints. In chapter 4 we explore in detail the real data set that will be used to conduct our experiments. Then in chapter 5 we investigate how the existing contextual multi-arm bandit algorithms fit our data set in the single campaign case. After that in chapter 6 we consider the complete scenario of targeted contextual multi-arm bandits. We present our results on synthetic and real data in chapter 7. We then close with conclusions and potential future work in chapter 8.

\(^1\)https://www.1plusx.com/
Chapter 2

Related Work

In this chapter we introduce formally the contextual multi-arm bandit setting and cover its relevant algorithms. Then, we describe the notion of bandits with knapsacks. We conclude with the existing literature on the ways to evaluate contextual multi-arm bandit algorithms.

2.1 Contextual Multi-Arm Bandits

Here we describe formally the contextual multi-arm bandits setting and review relevant algorithms.

2.1.1 Problem Setting

We are given a set of $n$ arms. When an arm $a$ is pulled a reward $r_a \in [0, 1]$ is observed. All rewards for the arms are drawn from an unknown distribution $P$ with parameters specific to each individual arm. We define the contextual multi-arm bandit problem as a continuous game over time $T$. At each iteration $t$ a new context $c_t \in \mathbb{R}^d$ is observed, then a player pulls an arm $a_t$ and finally a reward is revealed. Note that no additional reward information is given for the arms which were not chosen. The goal of the player is to maximize the expected gains from playing the game: $\max \mathbb{E}\left[\sum_{t=1}^{T} r_{a_t}\right]$.

An algorithm $A$ for the contextual multi-arm bandit problem must choose at every time $t$ the optimal arm $a_t$ given a context $c_t$ and the observation history $(a_1, c_1, r_1), (a_2, c_2, r_2), \ldots, (a_{t-1}, c_{t-1}, r_{t-1})$. The goal of the algorithm is to minimize the expected regret defined as the optimal rewards that could have been observed at each iteration $a_t^*$ minus the rewards obtained by following the algorithm:

$$\mathbb{E}[\text{regret}] = \mathbb{E}\left[\sum_{t=1}^{T} r_{a_t^*}\right] - \mathbb{E}\left[\sum_{t=1}^{T} r_{a_t}\right].$$
2. Related Work

2.1.2 Algorithms

All multi-arm bandit algorithms try to balance between exploration and exploitation. During the exploitation phase, an algorithm tries to capitalize gain based on the rewards obtained from previous iterations. A strategy that only performs exploitation could lead to intermediate short term gains, however in the long term it may perform poorly since it can only act according to an incomplete view of the available options. At the same time in the exploration phase an algorithm would try to extend its view of the potential action rewards. Performing only exploration would clearly lead to suboptimal reward gains since we would never take advantage of the obtained information.

ε-Greedy

ε-Greedy is a simple, widely used multi-arm bandits algorithm. At each iteration with probability $1 - \varepsilon$ the algorithm selects the arm with highest empirical mean and with probability $\varepsilon$ it chooses an arm uniformly at random. This simplistic algorithm suffers from two main shortcomings. First during its exploitation phase an arm is chosen greedily without taking into account potential long-term effects. And secondly, the algorithm’s unguided exploration strategy disregards any of the already collected information per arm.

UCB

Upper confidence bound (UCB) algorithms family is a widely used bandit strategy proposed first in Auer et al. [2002]. In comparison to the unguided exploration of ε-greedy algorithms UCB doesn’t have a clear separation between its exploration and exploitation phases. Instead it chooses which arm to pull based on an upper confidence bound for its rewards. This ensures that an arm can not remain completely unexplored but also that if an arm has a very low mean and not too large variance it would not be chosen.

LinUCB

LinUCB [Li et al., 2010] is an UCB algorithm that extends to the contextual bandit cases. In the setting that LinUCB considers, each arm $a$ is associated with a feature vector $x_{t,a} \in \mathbb{R}^d$. Moreover, algorithm assumes a linear dependency between the expected observed rewards and the context per arm:

$$
E[r_{t,a}|x_{t,a}] = x_{t,a}^\top \theta^*
$$
2.1. Contextual Multi-Arm Bandits

where $\theta^*$ is the true unknown coefficient vector. This model is called disjoint since there are no shared parameters between the different arm feature vectors.

Now, let $D_t \in \mathbb{R}^{t \times d}$ and $y_t \in \mathbb{R}^t$ be the historical data up to time $t$, where the $i^{th}$ row of the matrix $D_t$ is the feature vector of the arm pulled at time $t$ and $y_i$ is the corresponding response value. Then by applying ridge regression we can evaluate the closed-form approximation for $\theta^*$ at time $t$:

$$\hat{\theta}_t = (D_t^T D_t + I_d)^{-1} D_t y_t$$

where $I_d$ is the identity matrix in $\mathbb{R}^d$.

**Theorem 2.1** If the reward components in $y_{t,a}$ are independently conditioned on their corresponding rows in $D_t$. Then for any $\delta > 0$ and $x_{t,a} \in \mathbb{R}^d$, and a constant $\alpha = 1 + \sqrt{\ln 2/\delta/2}$ it can be proven that with probability at least $1 - \delta$:

$$|x_{t,a}^T \hat{\theta} - \mathbb{E}[r_{t,a} | x_{t,a}]| \leq \alpha \sqrt{x_{t,a}^T (D_{t,a}^T D_{t,a} + I_d)^{-1} x_{t,a}}$$

Using the inequality above we get a reasonably tight upper confidence bound for the expected payoff per arm. As a typical UCB algorithm LinUCB pulls the arm with the highest upper confidence interval:

$$a_t = \arg\max_{a \in \{a_1,..,a_N\}} (x_{t,a}^T \hat{\theta}_a + \alpha \sqrt{x_{t,a}^T A_{t,a}^{-1} x_{t,a}})$$

where $A_{t,a} = D_{t,a}^T D_{t,a} + I_d$.

The time and space complexity of LinUCB make it very appealing, since it has $O(N)$ and at most $O(d^3)$ computational complexity, where $N$ is the number of arms and $d$ is the feature vector dimensionality.

**Thompson Sampling**

Thompson Sampling [Thompson, 1933] is a old heuristic for multi-arm bandits based on Bayesian ideas. It only recently generated a significant interest due to several studies that show good empirical results while using it (Graepel et al. [2010], Chapelle and Li [2011], Kaufmann et al. [2012]).

**Thompson Sampling for Contextual Bandits with Linear Payoffs**

Agrawal and Goyal [2012] proposed an approach on how to use Thompson sampling in the contextual setting. Similarly to LinUCB Li et al. [2010] the authors assume that for an arm $a$ at time $t$ and context $c_{t,a} \in \mathbb{R}^d$ we observe linear payoffs: $r_{t,a} = c_{t,a}^T \theta^*$, where $\theta^*$ is again the unknown true coefficient vector.
In order to model this dependency in the context of Thompson sampling the authors presume Gaussian likelihood for the rewards $r_{t,a} \sim \mathcal{N}(c_{t,a}^T \theta, \nu^2)$, and Gaussian prior for $\theta \sim \mathcal{N}(\hat{\theta}_t, \nu B^{-1}_t)$, where $\nu$ is a constant and $B_t$ and $\hat{\theta}_t$ are defined as follows:

$$B_t = I_d + \sum_{i=1}^{t-1} c_{t,a_i} c_{t,a_i}^T$$

$$\hat{\theta}_t = B_t^{-1} \left( \sum_{i=1}^{t-1} c_{t,a_i} r_{t,a_i} \right)$$

Then to compute the model posterior distribution at time $t+1$, we can use the likelihood and prior at time $t$ i.e. $P(\hat{\theta}|r_{t,i}) \propto P(r_{t,i}|\hat{\theta}) P(\hat{\theta})$.

Subsequently to choose the best arm we simply sample $\hat{\theta}_t$ from the posterior distribution $\mathcal{N}(\hat{\theta}_t, \nu^2 B^{-1}_t)$ and play the arm that maximizes the reward estimate:

$$a_t = \arg\max_{a \in \{a_1,..,a_N\}} c_{t,a}^T \hat{\theta}_t$$

In the same paper [Agrawal and Goyal, 2012] the authors are the first to prove a regret bound for Thompson sampling in contextual multi-arm bandit setting of $O\left(\frac{d^2 \sqrt{T}}{\nu} + \epsilon\right)$ where $d$ is the dimensionality of the context and all $\epsilon \in (0,1)$. They do so by assuming that $(r_{t,i} - c_{t,a_i}^T \theta)$ is R-Sub-Gaussian.

**Gaussian Processes**

Gaussian processes (GP) are a common choice for function modeling [Rasmussen and Williams, 2005]. A GP uses a collection of random variables, one for each observed data point, such that any subset of these random variables has a multivariate normal distribution. One of the main advantages of GP comes from using kernel functions to measure the similarity between points. For each data point $d \in \mathcal{D}$ a Gaussian process $GP(\mu(d), k(d,d'))$ is fully defined by a covariance function $k(d,d') = E[(f(d) - \mu(d))(f(d') - \mu(d'))]$ and a mean function $\mu(d) = E[f(d)]$, where $f$ is the modelled function.

On the other hand, there are simple formulae for the derivation of the mean and covariance functions of the posterior distribution for GP. If we have a vector of noisy observations $y_T = (y_1, y_2, .., y_T) : y_i = f(d_i) + \epsilon_i$, where $\epsilon_i$ are i.i.d. normally distributed noise with mean 0 and variance $\sigma^2$, then it can be proven that the posterior distribution over $f$ at time $T$ is also a GP with mean and covariance specified below:

$$\mu_T(d) = k_T(d)^T(K_T + \sigma^2 I)^{-1} y_T$$

$$k_T(d,d') = k(d,d') - k_T(d)^T(K_T + \sigma^2 I)^{-1} k_T(d')$$
where \(k_T(d) = [k(d_1, d), k(d_2, d), ..., k(d_T, d)]\) and \(K_T\) is the positive semi-definite kernel matrix \([k(d, d')]_{d, d' \in D}\).

Gaussian Processes have been used to model the non-linear nature of reward functions. Srinivas et al. [2009] propose an upper confidence bound algorithm that uses GP to approximate the reward function. Later Vanchinathan et al. [2014] uses GP to model similarities between users and items when their numbers are large.

Even though GP provides strong theoretical guarantees using it can be computationally infeasible due to the need to invert the covariance matrix, resulting in \(O(n^3)\) training time and \(O(n^2)\) memory, where \(n\) is the number of data points.

2.2 Budgeted Multi-Arm Bandits

Badanidiyuru et al. [2013] propose a generalization of the multi-arm bandits problem - "bandits with knapsacks". The premise for the work is that budgets and time constraints are often applied in practice in the application domains of bandit problems.

Building up on the contextual multi-arm bandit definition from the beginning of this chapter we will now describe the bandits with knapsacks setting. Besides the list of arms we were give, we also are presented with a set of \(d\) resources. For each resource \(i\) there is a predefined budget \(B_i\) representing the maximal amount of budget that can be consumed. When an arm is pulled, we observe both a reward and a vector of the consumed values from each one of the \(d\) resources. In this setting the goal of a player would be to maximize their reward before any of the resource budgets is exceeded.

The authors argue that achieving a sub-linear regret for this setting is much harder than when no limitations are applied and propose two general solution approaches. The first approach called \textsc{BalancedExploration} aims to choose the best from an exhaustive list of potential strategies. The idea is to explore as much as possible while avoiding obviously sub-optimal solutions. The second and much more relevant to this thesis algorithm called \textsc{PrimalDualBwk} follows the idea of greedily selecting the arm with the greatest estimated reward per unit of resource consumption. The main challenge with this approach is that there is no clear estimator for unit of resource consumption, therefore the authors propose using and UCB and LCB estimators for the rewards and resource consumption vectors respectively. The algorithm stops when any of the resources has been exhausted.

Later Agrawal and Devanur [2015] consider budgets with knapsacks in the contextual bandit setting. Similarly to previous contextual approaches for Thompson Sampling [Agrawal and Goyal, 2012] and UCB [Li et al., 2010]
the authors assume linear dependency between the context and observed rewards.

2.3 Evaluation Methodology - Exploration Scavenging

Popular offline machine learning evaluation techniques, where one splits existing data into train and test data sets, have been proven inapplicable in the contextual multi-arm bandit setting. The reasoning is that an arm’s context can change with every iteration and a train/test approach is incapable of capturing that change.

Langford et al. [2008] propose a principled method called “exploration scavenging” for evaluating contextual multi-arm bandit algorithms. The technique can be used as an accurate estimator of the value of new policies as long as they do not depend on the current input and choose each action sufficiently often. The idea of the method is applied iteratively to offline data such that a new policy $\pi$ can be evaluated as follows:

1. Read the observed information: context $c$, displayed arm $a_i$, a click/no click information $y$ and a list of the potential arms: $L = \{a_1, a_2, ..., a_N\}$.
2. The policy $\pi(c, L)$ uses the context to choose arm $a_j \in L$.
3. If the displayed arm $a_i$ is the same as the chosen arm $a_j$, then we can use the click/no click information to update our model.
4. Otherwise, if the displayed arm $a_i \neq a_j$ then we skip that impression information.

This offline evaluation method will be used throughout the thesis as a primary evaluation strategy.
Chapter 3

Problem Formulation

In chapter 2 we saw classical contextual multi-arm bandit algorithms applicable in recommender systems. We also outlined the generalization of multi-arm bandits called "bandits with knapsacks". In this chapter we describe our problem setup and define the mathematical formulation on which the main contributions of this thesis are built.

3.1 Setup

Ad recommender systems can naturally be formulated as a multi-arm bandit problem where one maximizes click through rate (CTR). There are two main differences between the already described bandit settings and our particular setup specific to 1PlusX that need to be clarified before we can formally state our problem.

First, in the setting at hand we have available user embeddings, however there is no feature information for the ad campaigns. Moreover there are a large number of users \( n \) and a considerably smaller number of ads \( m \), i.e. \( n \ll m \). Therefore in order to better reason about the problem, we switch the usage of the items and users from the traditional recommendations setting. In all considered algorithms from chapter 2 a publisher site chooses a subset of the *items* to be displayed to a user. In our case a more appropriate interpretation is that a publisher site chooses a subset of the *users* to whom a particular ad will be displayed. This inversion is made to ensure that the available contextual information is maximized. From this setting it is now natural to consider that in order to solve the problem we have to split the users into disjoint subsets s.t. each user is assigned to a different ad campaign. This interpretation does not fundamentally change the problem at hand and the algorithms that can be used.

Second, in the "bandits with knapsacks" Badanidiyuru et al. [2013] setting
only maximal budget limitations are considered. In our case (see 1.1.2) however we want to ensure a minimal target budget is achieved. Therefore the goals of the two settings don’t overlap, however they are closely related.

We now continue with the mathematical formulation of the problem at hand.

3.2 Mathematical Formulation

We are given a total impression count $T$ often referred to as total time. There are $m$ different ad campaigns: $A = \{a_1, a_2, \ldots, a_m\}$ and $n$ users $K = \{k_1, k_2, \ldots, k_n\}$. Each campaign has a list of users $K_i = \{k_{i1}, k_{i2}, \ldots, k_{ip_i}\}$ to whom it can be displayed to, where $K = \bigcup_{i \in \{1, \ldots, m\}} K_i$ and for any $i, j$ s.t. $i \neq j$ : $K_i \cap K_j = \emptyset$.

At each iteration $t = 1, 2, \ldots$ we first observe a user $k_i$ and a context $c_t \in \mathbb{R}^d$, then display the ad $a_j$ s.t. $k_i \in K_j$ and finally we observe a reward $r_t(k_i, c_t, a_j)$.

In our setting each campaign has a target budget $B = \{B_1, B_2, \ldots, B_m\}$. The sum of the target budgets cannot exceed the total time $T$ i.e. $\sum_{i \in \{1, \ldots, m\}} B_i \leq T$. We define $B_{ti}$ to be the remaining target budget per campaign $a_i$ at time $t$. When a campaign is displayed to a user, the remaining target budget of that campaign is reduced by 1.

If at any time $t$: $\sum_{i \in \{1, \ldots, m\}} B_{ti} > T - t$ then the running algorithm stops. On the other hand if $B_{ti}$ reaches 0 then we don’t have to stop execution and we have no remaining obligation to display ad $a_i$.

We assume that the order in which users are observed, the campaigns that they were assigned to, the observed rewards and the consumed resources are i.i.d.

3.3 Optimization Criteria

Similarly to contextual bandits our optimization criteria is to maximize expected rewards or minimize the expected regret before the game ends:

\[
\begin{align*}
\text{minimize} \quad & \sum_{t} \mathbb{E}[r^*_t] - \sum_{t} \mathbb{E}[r_t(k_{ij}, c_t, a_{ji})] \\
\text{subject to} \quad & \sum_{j \in \{1, \ldots, m\}} B_{ji} \leq T - t, \ t = 1, \ldots, T.
\end{align*}
\]

where $r^*_t$ is the optimal reward at time $t$. 

12
3.4 Algorithmic Interpretation in Our Case

As discussed the target budget setting at hand is not the same as the one described in "bandits with knapsacks" Badanidiyuru et al. [2013]. However there are general similarities that will guide the solution approach where we will try to estimate the reward per unit of resource consumption (see 2.2).

Let \( q_t(k_i, c, a_j) \) be the consumed resource estimator, where \( k_i \) is the observed user, \( c \) is the provided context and \( a_j \) is an ad. On every iteration the user \( k_i \) consumes a single impression. At the same time we need to include the information of the remaining target budgets that need to be fulfilled. When a campaign has a large remaining target budget it has to be prioritized over a campaign with a small remaining target budget. That can be achieved by a scaling of the costs per campaign. An example cost function with such scaling is: 

\[
q_t(k_i, c, a_j) = \frac{1}{\log(B_{jt} + 1) + 1}. 
\]

Let \( f(k_i, c, a_j) \) be the evaluation function for estimating the CTR of user \( k_i \) observing campaign \( a_j \) with context \( c \). Then a user \( k_i \) is part of the recommendation list of users \( K_j \) for \( a_j \) iff:

\[
a_j = \arg\max_{a_i \in A} \left( \frac{f(k_i, c, a_1)}{q(k_i, c, a_1)} \cdot \frac{f(k_i, c, a_2)}{q(k_i, c, a_2)} \cdot \ldots \cdot \frac{f(k_i, c, a_m)}{q(k_i, c, a_m)} \right)
\]

In order to find \( f \) we can use any of the contextual bandit algorithms described in chapter 2.

3.5 Model Constraints

Our main constraint as described above is that while we optimize the CTR we need to ensure we fulfill the relevant target budget constraint.

Lets consider the case when the timeline \( T \) is indefinite. Then all target budgets would eventually, always be met converting the problem to the baseline contextual multi-arm bandits setting. Therefore from now on we assume that \( T \) is not indefinite.

Since we deal with real world data of large size an incremental update of the prediction model after each iteration would be infeasible from a computational standpoint. In order to account for this restriction, all proposed algorithms will perform incremental batch learning.

3.6 Setting Assumptions and Relaxations

Here we define few assumptions and relaxations necessary for the real world data to fit within the mathematical formulation above.
3. Problem Formulation

- First and foremost, we assume that all publishers care about the clicks and conversions of the displayed ad campaigns. Otherwise there will be no need to use algorithms to optimize CTR.

- We assume that all campaigns run for the same time slot and the same set of users. This means there is no filtering of users displayed to the different campaigns and that the campaigns are running at the same time. The requirements are crucial since there is no point in optimizing campaigns which are not concurrent between each other.

- We assume that only a single ad campaign is to be displayed to a user at a given time. Clearly this is not necessarily true since more than one ads can be displayed on the same web page. However this relaxation allows us to create disjoint recommendation lists and clearly state the user’s preference between different ad campaigns.

- For simplicity of setup we assume that all recommendation lists will be updated simultaneously. This guarantees that there is never a time when a user can be part of more than one recommendation list or that they are not part of any.

- Ad fatigue - the situation when consumers get overwhelmed by the abundance and repetition of ads they see, will not be modelled in this thesis. The reason is that we want to focus on discovering how best to pair a user and an ad based on the user’s interest.

- We assume that user embeddings do not change or adapt for the duration of an experiment. This is done for two main reasons. First it allows to remove bias potentially brought by any changes in the embeddings. And second, many of the contextual algorithms can not currently handle changes in the dimensionality of the underlying context.

- We assume that the list of potential ad campaigns that can be displayed to the users is available after any ad slot bidding is executed. Again a necessary condition to ensure we focus on finding best fit between a user and an ad.
Chapter 4

Experiment Data Set

In this chapter we cover the data used to run all the real world experiments in this thesis. We first describe the general nature of the data and then focus on the specifics of the ad campaigns used in our experiments. We complete by discussing the limitations to the data set at hand.

4.1 Data Source and Format

As mentioned in the introductory chapter, all of the real data used throughout this thesis has been provided by 1PlusX\(^1\). It has been obfuscated so that it complies with the General Data Protection Regulation Council of European Union [2016]. No identifiable user information is persisted. More specifically the user hash provided by 1PlusX is the result of a one way random hashing of the real user id.

In addition, all non-relative numbers reported in this thesis have been scaled by a random factor in order to protect business-sensitive information. The data encompasses a week of ad activity logs and the relevant user embeddings for that time period.

4.1.1 The Ad Activity Log

As briefly described in chapter 1 publishers collect usage data to improve their ad recommender systems. In this thesis we use a similar activity log to evaluate our algorithms offline. The ad activity log has the following format:

\[
\text{List} \left[ \text{UserHash}, \text{Timestamp}, \text{Click/NoClick}, \text{AdCampaignId} \right]
\]

From now on in this thesis each entry of the historical log will be referred to as an impression. For the purpose of our experiments we define an addi-

\(^1\)https://www.1plusx.com/
4. Experiment Data Set

The experimental concept of ad campaign audiences. An audience represents the impressions that an ad campaign is displayed to.

4.1.2 The User Embeddings

A user embedding is a multi-dimensional vector representation of a user’s historical activity information. During this thesis user embeddings are used as contextual information. Consequently the thesis goals include learning how to recommend ads to users given their feature vectors.

The way the user embeddings were created or the kind of features they encompass is unknown. However their format is:

\[ \text{List}[\text{UserHash}, \text{UserEmbeddingVector}] \]

where the UserHash can be matched to the one in the ad campaign activity log.

It is important to note that the user embeddings remain the same for the duration of the ad activity log. This is necessary since the contextual bandits algorithms considered in chapter 2 (see 2.1.2) can not handle any dimensional or rotational changes to the contextual information. Dealing with such changes is out of the scope of this thesis.

![Figure 4.1: Impression count and CTR per ad campaign](image)

4.2 Experiment Log Metadata

For the experiments were selected 5 ad campaigns and their ad logs for almost a full week of data. On figure 4.1 it can be observed that all the
campaigns have comparable audiences.

![Grouping of users by number of seen campaigns](image1)

**Figure 4.2:** Grouping of users by number of seen campaigns

Furthermore, some exploratory analysis into the users’ behavior for the duration of the available log showed that the majority of the users (about 80%) have seen only a single ad campaign (see figure 4.2). From the remaining 20% of the users less than 2% have viewed more than 2 campaigns.

![Ad clicks vs views for the users that have clicked at least one ad](image2)

**Figure 4.3:** Ad clicks vs views for the users that have clicked at least one ad

Now considering only the users that have performed at least one click it can be observed that about 99.7% have clicked on a single campaign (see figure 4.3. Therefore with high confidence it can be deduced that within the selected experiment data set there is an insignificant number of users that like to click on ads without having any particular interest in them.
4.3 Data Limitations

Finally, in order to better understand the setup we describe some limitations of the experiment data set.

**Ads Placement** There is no information about the adverts placement. It is possible that the same advert was placed in few very different positions on the web page. This has been known to affect significantly an advert’s CTR [Joachims et al., 2005].

**Ads Concurrency** In the experiment log there are no details about all the ads that could have potentially been displayed at a given time. It is possible that not all the 5 selected campaigns were running for the same time slot.

**Multiple Ad Campaigns Displayed Simultaneously** The logs do not provide information as to whether multiple campaigns were displayed at the same time. Even if that were the case then separate impression entries would be included in the log.

**No Web Content Information** There are no context details about the content of the web pages on which the ads were displayed. It is possible that there is a minimal to no overlap between the pages that two ads were presented on. Moreover, the different ad campaigns could have been shown on a wide range of content types for example sports, news, weather etc..

**User Filtering** There is no information if any preliminary filtering was applied to the different ad campaign user sets. Therefore, it is unclear if any filtering has affected the amount of users overlapping between the ad campaigns. Additionally, some users who were displayed one campaign might have never been seen another campaign, since they were filtered out.

**Target Budget Penalties** Lastly, we lack data about the different target budget penalties that were applied on the different ad campaign contracts.

Many of the aforementioned limitations are addressed in the "Setting Relaxations and Assumptions" discussed in chapter 3 (see 3.6).
Chapter 5

Single Campaign Investigations

Chapter 5 focuses on the effectiveness of the existing contextual bandit algorithms described in chapter 2 in a single campaign setting. We evaluate their performance over both simulated and real data sets.

5.1 The Single Campaign Approach

The goal of performing single campaign experiments is to evaluate the different contextual algorithms without any budgetary restrictions. Additionally, such analysis aids in getting a thorough understanding of the specifics of the experimental data sets.

5.1.1 Experiment Execution Flow

In the single campaign setting a contextual bandit algorithm chooses which ad to display to a user based on the idea described in chapter 3. More specifically, each ad campaign has a pre-selected list of recommended users. A particular ad is displayed to a user only if they are part of that ad’s user recommendation list.

A single experimental run follows the “exploration scavenging” procedure (see 2.3) described in chapter 2. In this single-campaign setting the information from an impression (see 4.1.1) is used only if the user is part of the recommendation list of the ad. All the remaining impressions from the log are discarded and not used for the training of the algorithm.

During each experiment, the log data is split into equally sized sequential batches based on the impression timestamp. In the beginning of an experimental run, the ad campaign recommendation list contains all of the users that could be potentially observed. Once a single batch data is filtered using “exploration scavenging” the algorithm is updated and a new recommenda-
5. Single Campaign Investigations

A recommendation list is provided. For all following batches the operation is repeated with the most recently created recommendation list.

Note that the first batch of data is used as a warm up. Since no users are filtered at the start, the algorithm consumes all the impressions from this phase. For all remaining batches a log entry is used only if the user in the entry is part of the recommendation list provided by the algorithm at the end of the previous period.

Such a batching process is necessary to enable processing of large amounts of data in a feasible time frame for this thesis. Note that this batching algorithm is identical to the regular bandits algorithms when the batch size is one.

5.1.2 Contextual Multi-Arm Bandits Algorithms

The algorithms that are explored in this ad recommendation setting can be grouped into two categories.

The first group is one of linear estimators where algorithms such as $\epsilon$-Greedy with Linear Regression, LinUCB and Thompson Sampling with linear payoffs are considered. The main parameters used to tune these approaches balance the exploration-exploitation trade-off.

The second group contains non-linear estimators such as Gaussian Processes (GP) and Neural Networks (NN). In the case of GP all of the users are clustered in order to reduce the computational time of fitting a GP model. Similar approach has been used in the past to provide article recommendations by Vanchinathan et al. [2014]. Additionally, the Matern kernel is used to estimate similarities between two users which is a commonly used kernel used with GP [Srinivas et al., 2009]. Similarly to the linear algorithms exploration-exploitation trade-off parameters are fine-tuned. However, further adjustments are done through changing the number of user clusters and tuning the kernel parameters. Last but not least, a NN with a single hidden layer is used as a CTR estimator. Here the main tuning was done on the learning rate and number of nodes in the hidden layer.

For more background on the algorithms refer to chapter 2.

From now on all the presented results both in the simulated and real world data sets are obtained using the best fitting parameters found for each of the algorithms. Additionally, in each experiment setting a baseline result is established in order to give a perspective about the achieved gains by the different algorithms. This baseline result is computed by running a bandit algorithm with random ad assignment on the same data sets.
5.1.3 Evaluation Metrics

In all of the experiments the created models are evaluated using both metrics related to revenue and profit and ones that assess the accuracy of prediction. For the first we use CTR which is a standard profit estimator in many bandit algorithms [Vanchinathan et al., 2014, Zhou, 2015]. The second type of metric is necessary to be able to estimate the accuracy of the predicted click probability and provide more reliable forecasting. Some of the metrics used include mean squared error (MSE), true positive rate (TPR) and normalized cross-entropy (NE). More specifically, the TPR is used to evaluate the amount of clicks that are missed using various recommendation list sizes and the normalized cross-entropy is the predictive log loss normalized by the entropy of the background CTR. Normalized cross-entropy has been used in Li et al. [2015] and He et al. [2014] as main evaluation metric.

5.2 Experiments with Synthetic Data Sets

The goal of using synthetic data is to investigate how the algorithms behave in varied settings.

5.2.1 Setup

The simulated data generation is achieved by transforming the reward information of already existing ad log for any selected ad campaign. The process of creating a synthetic log consists of a few steps.

First, a linear regression model \( f \) is trained on the full data set of a single ad campaign as a hindsight CTR estimator: \( E[r] = \hat{f}(c) \), where \( c \) is the current context and \( r \) is the reward.

Then, a transformation \( g \) is applied over the CTR estimator \( g(\hat{f}(c)) \). The explored transformations are linear, non-linear (sigmoid and low degree polynomials), with or without added error.

Subsequently, a copy of of the ad log data is created where iteratively the reward information on every impression entry is updated based on the current context \( c_i \) of the impression \( l \), the described above transformation and desired overall reward \( d \):

\[
 r_l = \begin{cases} 
 1, & g(\hat{f}(c_i)) \leq d \\
 0, & \text{otherwise} 
\end{cases}
\]

These alterations to the copied log result in controlled changes to the overall CTR and the click/no click underlying distribution.
5. Single Campaign Investigations

5.2.2 Results

On figure 5.1 one can observe what are the CTR gains between the baseline and the used algorithms in the settings described above. All linear algorithms perform very similarly obtaining about 5x improvement rate on CTR in all experiments. However, both NN and GP show significantly smaller gains especially in the non-linear case.

![Figure 5.1: CTR gains per algorithm type in various synthetic settings](image)

Subsequently, figure 5.2 displays how the user recommendation list size affects the CTR gains in the linear setting using LinUCB. It is clear that the
gains grow exponentially with the decrease of the recommendation size. In this case it is important to consider what is the true positive rate of the algorithm and how many clicks will not be observed. That is necessary since there is a danger that with a very limited recommendation list size a user type that is interested in the ad might never be detected. In other words the recommendation size is also a parameter that balances exploration-exploitation.

5.3 Experiments with Real World Data Set

Similarly to the simulated experiments the algorithms are compared on a real data set. The results obtained in the two settings are very comparable. On figure 5.3 it can be observed that the non-linear algorithms present significantly worse CTR than the linear ones. Moreover, all representatives of the latter perform very much the same throughout the entire experiment time progression (see table 5.1). This should not be surprising since the underlying assumptions for the three algorithms are very similar. However it should be mentioned that Thompson Sampling’s performance is a lot more susceptible to changes in the exploration-exploitation parameter, while EGreedy and LinUCB display more stable outcomes.

Looking at figure 5.4 one can see that even though the performance gains in this setting are much smaller than the synthetic one, however they are still substantial. Here specifically the experiment is run on multiple different ad campaigns and CTR gains are sustained for all of them.

Figure 5.3: CTR and MSE for Different Algorithms with Fixed Recommendation Size: 20%
5. Single Campaign Investigations

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CTR</th>
<th>Relative CTR Gains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.5205075</td>
<td>—</td>
</tr>
<tr>
<td>GP (Clustered)</td>
<td>0.9512467</td>
<td>182.7 %</td>
</tr>
<tr>
<td>NN</td>
<td>0.9736409</td>
<td>187.1 %</td>
</tr>
<tr>
<td>TS (Lin)</td>
<td>1.4241693</td>
<td>273.6 %</td>
</tr>
<tr>
<td>EGreedy (Lin)</td>
<td>1.4247989</td>
<td>273.7 %</td>
</tr>
<tr>
<td>LinUCB (Disjoint)</td>
<td>1.4259891</td>
<td>274 %</td>
</tr>
</tbody>
</table>

Table 5.1: CTR and relative CTR gains achieved by the explored algorithms

Finally, experiments with various batch sizes showed that the time between updates can have significant impact on the short term performance of an algorithm. The reason is that a smaller batch size naturally results in a quicker learning process for the models. However this affect is diminished in the long term once the models are fully converged.

5.4 Conclusions

In this chapter we established that the contextual bandits algorithms can successfully be applied on 1PlusX real and simulated data sets. Here we describe few main takeaways:

**Linear vs Non-linear Algorithms**  Both non-linear algorithms that were considered in this case showed unsatisfactory results compared to the linear ones in varied settings. Our assumption is that the very small CTR is the core reason why such bad results are observed. One could further investi-
gate regularization or improvements of these methods however that work is out of scope of this thesis.

**The User Recommendation List Size**  As discussed in both the simulated and real data settings the recommendation size has a large impact on the CTR. It can be used as parameter to manage exploration-exploitation however one has to be careful about the amount of information that could be missed in case the recommendation size is too small.

In conclusion, based on the results presented in this chapter all further experiments in the multi-campaign setting are conducted with LinUCB as baseline contextual bandits algorithm.
In this chapter we describe an approach for handling target budgets in the contextual multi-arm bandit setting. We then evaluate the proposed algorithm with synthetic data sets. Later, in chapter 7 we present its performance in the real world setting.

6.1 The Targeted Bandits Approach

In chapter 5 we explored contextual bandits algorithms. We now focus on algorithms that can handle adding target budgetary restrictions to the contextual bandit setting. These strategies can be used as an independent extension to existing bandits algorithms.

We first summarize the general contextual bandits algorithms. Given a user $u_i$, context $c$ and a list of ads $\{a_1, \ldots, a_m\}$ an algorithm $A$ uses its expected reward estimator $f(u_i, c, a_j)$ to evaluate which of the $m$ available ads is most appropriate for that user. The interpretation of expected reward is specific to each algorithm, however the rest of the setup is effectively the same for all contextual multi-arm bandits algorithms.

As described in chapter 3 our approach to handle target budgets consists of providing a consumed resource estimator $q_t(u_i, c, a)$ which is used as a scaling function for the expected reward estimator $f$ of any contextual multi-arm bandits algorithm. For time $t$ we propose that $q_t(u_i, c, a_j) = 1/(\log(B_{ji} + 1) + 1)^\varepsilon$ where $B_{ji}$ is the remaining budget for ad campaign $a_j$. Such normalization allows for prioritizing campaigns with higher remaining impression budget than others.

We now go over a few challenges with the above described approach and propose relevant solutions.
6. Multi-Campaign Approach

Large Target Campaign Budgets Variance  One of the main problems with the approach described above is the potential large difference in campaigns’ target budgets. If an ad campaign \( a_j \) has a very large target impression budget compared to the other campaigns \( B_j \gg B_i \) for any \( a_i \) (that could be the case with a simply larger campaign or a freshly started one) then the algorithm would almost always prioritize \( a_j \). This means that sometimes the remaining campaigns and their goodness of fit are completely disregarded resulting in effectively random ad assignment. In addition to leading to low immediate CTR, it results in a minimal exploration of the remaining campaigns.

We propose that instead of relying entirely on a single final target budget per ad campaign, we split it into multiple checkpoint budgets (for example on a daily or hourly basis). This way the exhaustion of the impressions in a target budget is regularized within the entire time frame and a single campaign does not overtake the campaign assignment. We further refer to this feature as target budget split.

Different Campaign Audience Sizes  Another challenge is inherent from the fact that campaigns can have varying audience sizes (see chapter 4). That could happen due to a predefined filtering of the users or of the publisher sites that the ad campaign is to be displayed to. Consequently, the expected number of viewers per campaign can be very different resulting in the following problematic scenario: There are two campaigns, campaign \( a_i \) with large target budget \( B_i \) and large expected audience \( A_i \) and another campaign \( a_j \) with small target budget \( B_j \) and small expected audience \( A_j \) s.t: \( B_i \gg B_j \) and \( A_i \gg A_j \). Similarly to the previous section we would almost always recommend \( a_i \). However, besides resulting in a reduced CTR, there are additional problems that surface in this case. More specifically, in the initial phase before the remaining target budget of campaign \( a_i \) becomes comparable with the smaller campaign one \( a_j \) (\( B_i \sim B_j \)), the first campaign \( a_i \) will continuously and overwhelmingly be displayed to users. This means that the small campaign \( a_j \) would only be displayed to a greatly reduced percent of its total potential users \( A_j^* \). This could easily result in the remaining audience being smaller than the desired target budget \( A_j^* < B_j \) resulting in a failure to comply with the contractual obligations.

Clearly such behavior is suboptimal. Therefore, we propose normalizing the consumed resource estimator by the expected audience size for each campaign:

\[
q_t^N(u_i, c, a_j) = \frac{q_t(u_i, c, a_j)}{\mathbb{E}[A_j^*]}
\]

where \( \mathbb{E}[A_j^*] \) is the expected audience size at time \( t \) for ad campaign \( a_j \). Using such normalization strategy can remedy this behavior by representing
the importance of a specific impression being assigned to a particular ad campaign. In order to calculate the normalization values one can estimate the audience sizes of ad campaigns based on their historical performance. However, the details of specific approximations strategies are out of scope for this thesis.

**Batch Implementation Challenges**  Similarly to chapter 5 we assume batch execution flow of the experiments in the multi-campaign setting. As a reminder, this kind of execution allows updates to the recommended list of users per campaign only in the end of the batch. However, if a campaign budget is exhausted early within the batch time frame, then the same campaign would continue to be shown to users. This could be problematic especially when the batch size is large. Therefore we propose two optimization strategies:

- On demand update - If the target budget of a campaign is exhausted, then trigger an on-demand update to the recommendation lists of the ads.
- Crop percent - An alternative approach is necessary if an on-demand update is not possible. We propose to evaluate how close a campaign is to reaching its target budget. When a target budget reaches a critically low number we additionally penalize its expected reward estimation to minimize its usage.

Finally, for the remainder of the thesis we will be referring to the combination of our targeted algorithm and LinUCB as $T_{\text{LinUCB}}$.

### 6.2 Experiments with Synthetic Data Sets

Similarly to the single-campaign experiments we evaluate the performance of the targeted contextual bandits algorithm on synthetic data sets. We first describe how the simulation data sets are generated and then present the obtained results for the targeted contextual bandit algorithm we have created.

#### 6.2.1 Setup

The activity logs from all available ad campaigns are used for the creation of the simulated data (see chapter 4). The logs are combined and sorted by timestamp. For impression $i$ in the combined log let $s(i)$ be the source ad campaign where the impression originated from.

The overall idea of the synthetic data generation is to have **full knowledge of the reward when a user is shown any of the ad campaigns**. In order to achieve this, for each impression the synthetic log has the full click information about each one of the ad campaigns. Note the contrast with real data where for
an impression there is information only for the reward obtained by showing a single campaign. This difference results in the lack of need to use "exploration-scavenging" (see 2.3). As a reminder, this technique uses impression information only when the ad predicted by the tested model and the displayed ad are the same. All remaining impression data is discarded. However, in this multi-campaign simulated environment there is full knowledge of the reward gained by displaying an specific ad to a user.

If $r_{ai}(ci)$ is the reward obtained from showing ad campaign $ai$ in the context $ci$ of impression $l$, $r_{ai}(ci)$ exists only if $s(l) = ai$. Then in order to create a simulated environment we have to provide a generative function for $\tilde{r}_{ai}(c)$ for all $ai \in \{a_1, \ldots, a_5\}$.

We propose two approaches to achieve this.

Lower Bound Simulations The first approach provides no additional bias when generating click/no click information for the non-observed campaigns. Since every impression $l$ in the combined log originates from an ad campaign $s(l)$, there is knowledge about the reward in that context. The reward information for the other ad campaigns is generated based on random sampling and their estimated CTR:

$$\tilde{r}_{ai}(ci) = \begin{cases} r_{ai}(ci), & \text{if } s(l) = ai \\ 1, & y^* \leq \tilde{f}_{ai}(ci), \text{ where } y^* \sim \text{Uniform}(0, 1) \\ 0, & \text{otherwise} \end{cases}$$

where $ci$ is the context of impression $l$. We call this simulation type lower bound, since the amount of true non-random information presented to the algorithms is minimal and the results obtained can not be better than the real world data results.

Hindsight Simulations In the second simulation type we aim to create an ideal environment where one can evaluate the performance of the bandit algorithms. In contrast to the lower bound simulation here all ad campaign rewards are synthetic. We use the hindsight information from the available ad campaigns. For each ad campaign $ai \in \{a_1, \ldots, a_5\}$ we first learn linear regression models $\tilde{f}_{ai}$ based on their activity logs s.t. $\mathbb{E}[r_{ai}] = \tilde{f}_{ai}(c)$, where $c$ is the context and $\mathbb{E}[r_{ai}]$ is the expected reward for campaign $ai$. Then a generative function $\tilde{r}_{ai}(c)$ is used for each campaign and context to produce the click/no click information. It is based on the estimated expected reward $\tilde{f}_{ai}(c)$ and a transformation $g$: $\tilde{r}_{ai}(c) = g(\tilde{f}_{ai}(c))$. Three separate assumptions are made for $g$:

1. no error
2. linear model class for the rewards
6.2. Experiments with Synthetic Data Sets

3. the data is fully characterized by the embeddings at hand

Each one of them is relaxed until a performance similar to the one observed in the real world data is achieved resulting in the following simulation scenarios:

- All the assumptions are applied and the expected reward is randomly sampled from a normal distribution.
  \[ g(\tilde{f}_{a_i}(c)) \sim \mathcal{N}(\tilde{f}_{a_i}(c), \sigma^*) \]

- Assumptions 2 and 3 are applied. The expected reward is sampled from a normal distribution with error.
  \[ g(\tilde{f}_{a_i}(c)) \sim \mathcal{N}(\tilde{f}_{a_i}(c), \sigma^*) + C \cdot \tilde{f}_{a_i}(c) \cdot \mathcal{N}(0, 1) \]

- Only assumption 3 remains active. The expected reward is the result from a non-linearity \( h \) with error.
  \[ g(\tilde{f}_{a_i}(c)) \sim h(\tilde{f}_{a_i}(c)) + C \cdot \tilde{f}_{a_i}(c) \cdot \mathcal{N}(0, 1) \]

- No active assumptions remain. A random sample from Chi-squared distribution is added to model the additional click information not captured by the user embeddings.
  \[ g(\tilde{f}_{a_i}(c)) \sim h(\tilde{f}_{a_i}(c)) + \tilde{f}_{a_i}(c)(C_1 \cdot \chi^2(df) + C_2 \cdot \mathcal{N}(0, 1)) \]

Where \( C, C_1 \) and \( C_2 \) are constants, \( \sigma \) is appropriately scaled standard deviation and \( h \) is either a linear or non-linear generative function. Then for each impression \( l \), campaign \( a_i \) and desired CTR \( D \) the generated click/no click becomes:

\[
\tilde{r}_{a_i}(c_l) = \begin{cases} 
1, & \text{if } g(\tilde{f}_{a_i}(c_l)) \leq D, \\
0, & \text{otherwise}
\end{cases}
\]

6.2.2 Results

We now present the results from the assessment of the discussed target budgets algorithms in the synthetic environments described above.

We start by evaluating which of the hindsight simulations performs most similarly to the real world data at hand. The simulations are executed with the basic LinUCB without budgeting restrictions in order to avoid potential bias introduced by the addition of target budgets. On figure 6.1a one can observe that the behavior of the simulation using a linear generation function
6. Multi-Campaign Approach

(a) Hindsight Simulation Comparison of LinUCB
(b) LinUCB vs T_LinUCB on best fitting simulation data

Figure 6.1: Simulation results

Figure 6.2: Performance of T_LinUCB with various added noise
6.2. Experiments with Synthetic Data Sets

with chi squared synthetic clicks and added error is closest to the real data results. Then on figure 6.1 we compare the performance of LinUCB with T_LinUCB. The budgeted algorithm performs very similarly to the baseline where the relative CTR loss is approximately 2.08%. We further display how the amount of information not captured by the user embeddings affects the CTR gains. We simulate this behavior by generating additionally added value with chi squared distribution. On the right side of figure 6.2 is plotted the added value density and on the left we can observe the impact to the T_LinUCB performance. Naturally the more random information is added to the basic linear model the worse the algorithm performs.

Figure 6.3: The effect of the target budget size in both lower and upper bound simulation settings w/o clickers

Finally, we consider how the size of the target budgets affects the performance of the algorithm. Each campaign has a predefined maximum target budget (MTB). On figure 6.3 it can be observed how reducing the target budget with respect to the MTB results in larger CTR increases. The reason is that the algorithm does not have to prioritize the ad campaigns based on budgetary restrictions and is unobstructed to recommend to the users the best fitting to their needs campaign. This behavior is consistent in both the hindsight and lower bound simulations.
Experimental Results

In this chapter we evaluate how the proposed target contextual multi-arm bandits algorithm performs with the real world data set. We consider the impact of the features proposed in chapter 6. Finally, we conclude by comparing our model with the existing non-targeted algorithm.

7.1 Setup

In the experiments with real data in the multi-campaign setting we use the data set described in chapter 4. The ad activity log for the 5 campaigns is combined and sorted according to timestamp.

In the beginning of each experimental run the targeted contextual bandit algorithm is provided target budgets for the ad campaigns as metadata during its initialization. Afterwards, the familiar "exploration scavenging" procedure (see 2.3) is used to decide which impressions can be used to train our model and which are going to be discarded.

The target budgets $B_i$ for each ad campaign $i$ are computed based on their original impression count. Furthermore, the budget sizes are adjusted to account for the loss of impressions caused by using "exploration scavenging".

7.2 Feature Impact

We first present the performance of the targeted contextual multi-arm bandit algorithm in its basic form without including any additional improvements. We remind the reader that throughout all the multi-campaign experiments the underlying contextual bandit algorithm is LinUCB [Li et al., 2010]. In addition to measuring the differences between the targeted algorithms and the baseline LinUCB model, we compare them to the performance of a random ad assignment model. Moreover, we define CTR gains as the ratio between
7. Experimental Results

a contextual bandits model (with or without budgets) and the random ad assignment model. Furthermore, for any of the presented results below we use values relative to the CTR gains and not to the CTR itself.

![Graph showing performance comparison]

Figure 7.1: Performance of LinUCB vs T_LinUCB (baseline)

On the left side of figure 7.1 we can see the performance of T_LinUCB compared to its unconstrained baseline. Unsurprisingly there is notable loss of precision. However on the right side of figure 7.1 we can observe that even in its basic form our algorithm provides a significant improvement in CTR relative to the random ad campaign assignment model, with an almost doubling in CTR (see table 7.1).

We now proceed with discussing the performance of the proposed in chapter 6 add-ons to T_LinUCB.

7.2.1 Target Budget Split

This first add-on attempts to deal with inordinately large campaigns. In our data set (see chapter 4) one of the campaigns has significantly larger (about 2x) target budget compared to the other campaigns. Therefore we would expect that replacing the all time target budgets with daily ones would have a positive impact on the CTR gains (see 6.1). We can observe the obtained results on figure 7.2a. The target split performs slightly worse than the baseline T_LinUCB algorithm in the beginning of the experimental run. However, towards the end of the experiment that pattern reverses and the algorithm completes with total relative CTR gain of about 5% (see table 7.1).
7.2. Feature Impact

(a) Target split

(b) Normalization

Figure 7.2: Feature performance

<table>
<thead>
<tr>
<th>Relative CTR Gains</th>
<th>T_LinUCB</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_LinUCB</td>
<td>—</td>
<td>196.9 %</td>
</tr>
<tr>
<td>T_LinUCB /w Daily Target Split (DTS)</td>
<td>5.4 %</td>
<td>207.4 %</td>
</tr>
<tr>
<td>T_LinUCB /w Normalization</td>
<td>20.5 %</td>
<td>237.2 %</td>
</tr>
<tr>
<td>T_LinUCB /w On-Demand Update</td>
<td>6.2 %</td>
<td>209 %</td>
</tr>
<tr>
<td>T_LinUCB /w Crop Percent</td>
<td>&lt;0.1 %</td>
<td>196.9 %</td>
</tr>
</tbody>
</table>

Table 7.1: Relative CTR gains between baseline T_LinUCB and Random CTR and the proposed T_LinUCB with add-ons.

7.2.2 Audience Normalization

The second add-on is very important for our experiments since the audiences for each of the ad campaigns vary significantly. Thus, it is not surprising that the inclusion of this feature results in a pronounced improvement of our basic T_LinUCB algorithm of more than 20% (see table 7.1). Similarly to the target split feature we start noticing the CTR improvement towards the end of the experimental run (see figure 7.2b). We believe the reason for this is that by using the normalization we maintain a larger variety of ad campaigns active towards the end and the underlying contextual bandits algorithm can choose more freely which the best fitting campaign is.

This add-on provides a great improvement to the baseline T_LinUCB algorithm. However we believe that this advancement is caused by the availability of good audience estimator. Therefore, we believe that a less reliable audience model would result in reduced impact to the CTR.
7. Experimental Results

7.2.3 Implementation Specific Add-Ons

We now present the results for the two features suggested as an improvement to the problems introduced by using a batch implementation of T_LinUCB. It should be noted that they are unnecessary in the case of a non-batch system.

In order to showcase the best results for the two methods, we present them in combination with the Daily Target Split add-on (DTS). The reason is that with daily budgets it would be much more likely that a target budget is exhausted within a batch than if there were a single total target budget. On figure 7.3 we can see that the on-demand update significantly improves the CTR of the algorithm with about 5.3% relative to the baseline. At the same time the crop percent feature executes considerably worse than T_LinUCB /w DTS reducing its CTR with 4.2% (see table 7.2). Although, this on-demand alternative has transpired as unsuccessful we believe that it can be researched further and its performance can be potentially improved if its parameters are adjusted to match the expected impressions within each batch time span.

7.3 LinUCB vs T_LinUCB

We now summarize the results from the best version of T_LinUCB. Optimal performance for the targeted contextual bandits context is achieved by T_LinUCB in combination with audience normalization, daily target split and on-demand update. The results can be seen on figure 7.4. Our algorithm has practically identical performance to its unconstrained baseline.
7.3. LinUCB vs T_LinUCB

<table>
<thead>
<tr>
<th>Relative CTR Gains</th>
<th>T_LinUCB /w DTS</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_LinUCB /w DTS</td>
<td>—</td>
<td>207.4%</td>
</tr>
<tr>
<td>T_LinUCB /w DTS</td>
<td>5.3%</td>
<td>218.5%</td>
</tr>
<tr>
<td>/w On-Demand Update</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>T_LinUCB /w DTS /w Crop Percent</td>
<td>- 4.2%</td>
<td>198.6%</td>
</tr>
</tbody>
</table>

Table 7.2: Relative CTR gains between T_LinUCB /w DTS, Random CTR and the implementation add-ons.

![Graph of LinUCB vs T_LinUCB](image1)

Figure 7.4: LinUCB vs T_LinUCB

<table>
<thead>
<tr>
<th>Relative CTR Gains</th>
<th>LinUCB</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinUCB</td>
<td>—</td>
<td>238.1%</td>
</tr>
<tr>
<td>T_LinUCB /w Add-ons</td>
<td>- 0.2%</td>
<td>237.7%</td>
</tr>
</tbody>
</table>

Table 7.3: Relative CTR gains between LinUCB, Random CTR and the best T_LinUCB implementation.

algorithm LinUCB losing less than 1% of the CTR (see table 7.3). The majority of the improvement from the basic T_LinUCB model is achieved by including the audience normalization feature.

Finally, we present the break down of impression count (figure 7.6) and CTR (figure 7.5) between the different campaigns. Given the overall CTR performance of the two algorithms presented above it is foreseeable that individual breakdown for each ad campaign is very similar. In fact, it is interesting that the impression counts per ad campaigns practically overlap for LinUCB and T_LinUCB. The reason is that the optimal solution found by LinUCB in fact complies with the budget constraints that were applied it this experiment. However, it is important to note that when that is not the
7. Experimental Results

case the CTR loss is very likely going to decrease more substantially.

Figure 7.5: CTR Per Campaigns

Figure 7.6: Impression Distribution Per Campaign
7.4 Conclusion

Overall, we can conclude that the proposed T_LinUCB method provides satisfactory results even when used without additional heuristics. Moreover, the experiments above showcase how add-on improvements like audience normalization, on-demand update and target split can close the performance gap between the basic targeted bandits strategy and the underlying contextual bandits algorithm. Therefore, we believe that the proposed general strategy can be an useful technique for handling targeted budgets in ads recommender systems.
Chapter 8

Conclusion

The primary contribution of this thesis is an algorithm which iteratively learns to maximize the performance of a group of display advertising campaigns each with its own impression delivery guarantee constraint. The introduced algorithm is an extension to existing contextual multi-arm bandits algorithms which now allows handling of target budgetary constraints. We proposed additional optimizations which resulted in significant CTR improvements on top of the baseline targeted contextual algorithm. We evaluated the proposed method on both synthetic and real world data sets and showed that in practice it can perform on parity with the unconstrained baseline algorithms.

Some of the challenges of this thesis included evaluating the best fitting contextual multi-arm bandits algorithms to the real world data set at hand. We were successful in establishing significant improvements with the linear contextual bandits algorithms. However, we were unable to provide satisfactory results with the non-linear algorithms like Gaussian Processes or Neural Networks. Even though these models could further be improved we believe that their poor performance is not detrimental to the way we handle target budgetary restrictions since the proposed extension is agnostic to the type of the underlying contextual multi-arm bandit algorithm.

Additionally, we were able to evaluate the target budget algorithms only on a limited data set. As future work one can confirm our findings in larger data sets with more ad campaigns, longer experimental run time and various campaign start and stop times. Furthermore, the real world data set that was used was not completely randomized and with no prior ad placement information which could have had impact on our findings.

Another potential future improvement includes expanding the proposed algorithm to simultaneously handle multiple target budget constraints per ad campaign. Besides being a nice generalization to our setting such extension
would allow for handling diverse business needs, such as including minimal number of clicks per ad campaign in addition to allowing only minimal impression count. Moreover, one can pursue combining our work and the "budgets with knapsacks" [Badanidiyuru et al., 2013] to create a model for both minimum and maximum constrains fulfillment.

Currently, our algorithm makes two important assumptions. First, that all ad campaigns have the same penalties and second, that the target budgets can always be met. However, in reality that is not necessarily the case. It could happen that the agreed upon impression count is unreachable within a given time frame. Therefore, it would be interesting to consider how the monetary size of the contractual penalty can affect the overall strategy of the algorithms so that revenue is still maximized. That might include deliberately failing a contract with considerably smaller penalty in order to uphold other contracts with financial sanctions.

In conclusion, in this thesis we presented an algorithm that handles contractual obligations between advertisers and publishers while simultaneously optimizing revenue. We provided an agnostic extension on existing contextual multi-arm bandits algorithms and displayed its effectiveness in experiments with both synthetic and real world data sets.


Thore Graepel, Joaquin Quiñonero Candela, Thomas Borchert, and Ralf Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft’s bing search engine. In Proceedings of the


Bibliography


Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

AD CAMPAIGN OPTIMIZATION - A TARGETED CONTEXTUAL BANDIT APPROACH

Authorised by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):

DIMITROVA

First name(s):

ANA

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date

ZURICH, 5.11.2018

Signature(s)

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.