

Personalized Ad Recommendation Systems for Life-Time Value Optimization with Guarantees

Georgios Theodorou
Adobe Research
theochar@adobe.com

Philip S. Thomas
UMassAmherst
and Adobe Research
phithoma@adobe.com

Mohammad Ghavamzadeh
Adobe Research and INRIA
ghavamza@adobe.com

Abstract

In this paper, we propose a framework for using reinforcement learning (RL) algorithms to learn good policies for personalized ad recommendation (PAR) systems. The RL algorithms take into account the long-term effect of an action, and thus, could be more suitable than myopic techniques like supervised learning and contextual bandit, for modern PAR systems in which the number of returning visitors is rapidly growing. However, while myopic techniques have been well-studied in PAR systems, the RL approach is still in its infancy, mainly due to two fundamental challenges: how to compute a good RL strategy and how to evaluate a solution using historical data to ensure its “safety” before deployment. In this paper, we propose to use a family of off-policy evaluation techniques with statistical guarantees to tackle both these challenges. We apply these methods to a real PAR problem, both for evaluating the final performance and for optimizing the parameters of the RL algorithm. Our results show that a RL algorithm equipped with these off-policy evaluation techniques outperforms the myopic approaches. Our results also give fundamental insights on the difference between the click through rate (CTR) and life-time value (LTV) metrics for evaluating the performance of a PAR algorithm.

1 Introduction

In personalized ad recommendation (PAR) systems, the goal is to learn a strategy (from the historical data) that for each user of the website, selects an ad with the highest probability of click by that user. Almost all such systems these days use supervised learning or contextual bandit algorithms (especially contextual bandits that take into account the important problem of *exploration*). These algorithms assume that the visits to the website are i.i.d. and do not discriminate between a visit and a visitor, i.e., each visit is considered as a new visitor that has been sampled i.i.d. from the population of the website’s visitors. As a result, these algorithms are myopic and do not try to optimize the long-term effect of the ads on the users. Click through rate (CTR) is a suitable metric

to evaluate the performance of such greedy algorithms. Despite their success, these methods are becoming insufficient as users incline to establish longer and longer-term relationship with their websites (by going back to them) these days. This increase in *returning visitors* further violates the main assumption underlying the supervised learning and bandit algorithms, i.e., there is no difference between a visit and a visitor. This is the main motivation for the new class of solutions that we propose in this paper.

Reinforcement learning (RL) algorithms that aim to optimize the long-term performance of the system (often formulated as the expected sum of rewards/costs) seem to be suitable candidates for PAR systems. The nature of these algorithms allows them to take into account all the available knowledge about the user in order to select an offer that maximizes the total number of times she will click over multiple visits, also known as the user’s life-time value (LTV). Unlike myopic approaches, RL algorithms differentiate between a visit and a visitor, and consider all the visits of a user (in chronological order) as a system trajectory. Thus, they model the visitors, and not their visits, as i.i.d. samples from the population of the users of the website. This means that although we may evaluate the performance of the RL algorithms using CTR, this is not the metric that they optimize, and thus, it would be more appropriate to evaluate them based on the expected total number of clicks per user (over the user’s trajectory), a metric we call LTV. This long-term approach to PAR allows us to make decisions that are better than the short-sighted decisions made by the greedy algorithms, decisions such as to propose an offer that might be considered as a loss to the company in the short term, but has an effect on the user that brings her back to spend more money in the future.

Despite these desirable properties, there are two major obstacles hindering the widespread application of the RL technology to PAR: **1)** how to compute a good LTV policy in a scalable way and **2)** how to evaluate the performance of a policy returned by a RL algorithm without deploying it (using only the historical data that has been generated by one or more other policies). The second problem, also known as *off-policy evaluation*, is of extreme importance not only in ad recommendation systems, but in many other domains such as health care and finance. It may also help us with the first problem, in selecting the right representation (features) for the RL algorithm and in optimizing its parameters, which

in turn will help us to have a more scalable algorithm and to generate better policies. Unfortunately, unlike the bandit algorithms for which there exist several *biased* and *unbiased* off-policy evaluation techniques (e.g., Li *et al.* [2010]; Strehl *et al.* [2010]; Langford *et al.* [2011]), there are not many applied, yet theoretically founded, methods to guarantee that a RL policy performs well in the real system without having a chance to deploy/execute it.

One approach to tackle this problem would be to first build a model of the system (a simulator) and then use it to evaluate the performance of RL policies [Theocharous and Hallak, 2013]. The drawback of this *model-based* approach is that accurate simulators, especially for PAR systems, are notoriously hard to learn. In this paper, we use our recently proposed *model-free* approach that computes a lower-bound on the expected return of a policy using a concentration inequality [Thomas *et al.*, 2015a] to tackle the off-policy evaluation problem. We also use two approximate techniques for computing this lower-bound (instead of the concentration inequality), one based on Student’s *t*-test [Venables and Ripley, 2002] and another based on bootstrap sampling [Efron, 1987]. This off-policy evaluation method takes historical data from existing policies, a baseline performance, a confidence level, and the new policy, as input, and outputs “yes” if the performance of the new policy is better than the baseline with the given confidence. This high confidence off-policy evaluation technique plays a crucial role in several aspects of building a successful RL-based PAR system. Firstly, it allows us to select a champion in a set of policies without the need to deploy them. Secondly, it can be used to select a good set of features for the RL algorithm, which in turn helps to scale it up. Thirdly, it can be used to tune the RL algorithm, e.g. many batch RL algorithms, such as fitted Q-iteration (FQI) [Ernst *et al.*, 2005], do not have a monotonically improving performance along their iterations, thus, an off-policy evaluation framework can be used to keep track of the best performing strategy along the iterations of these algorithms.

In general, using RL to develop LTV marketing algorithms is still in its infancy. Related work has experimented with toy examples and has appeared mostly in marketing venues (e.g., Pfeifer and Carraway [2000]; Jonker *et al.* [2004]; Tirenni *et al.* [2007]). An approach directly related to ours first appeared in Pednault *et al.* [2002], where the authors used public data of an email charity campaign, batch RL algorithms, and heuristic simulators for evaluation, and showed that RL policies produce better results than myopic’s. Silver *et al.* [2013] recently proposed an on-line RL system that learns concurrently from multiple customers. The system was trained and tested on a simulator and does not offer any performance guarantees. Unlike previous work, we deal with real data in which we are faced with the challenges of learning a RL policy in high-dimension and off-policy evaluation of these policies, with guarantees.

In the rest of the paper, we first summarize the three methods that we use to compute a lower-bound on the performance of a RL policy. We then describe the difference between CTR and LTV metrics for policy evaluation in PAR systems, and the fact that CTR could lead to misleading results when we have a large number of returning visitors. We then present

practical algorithms for myopic and LTV optimization that combine various powerful ingredients such as the robustness of random-forest regression, feature selection, and off-policy evaluation for parameter optimization. Finally, we finish with experimental results that clearly demonstrate the issues raised in the rest of the paper, such as LTV vs. myopic optimization, CTR vs. LTV performance measures, and the merits of using high-confidence off-policy evaluation techniques in learning and evaluating RL policies.

2 Preliminaries

We model the system as a *Markov decision process* (MDP) [Sutton and Barto, 1998]. We denote by s_t the feature vector describing a user’s t^{th} visit to the website and by a_t the t^{th} ad shown to the user, and refer to them as a *state* and an *action*. We call r_t the *reward*, which is 1 if the user clicks on the ad a_t and 0, otherwise. We assume that the users visit at most T times and set T according to the users in our data set. We write $\tau := \{s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_T, a_T, r_T\}$ to denote the history of interactions with one user, and we call τ a *trajectory*. The *return* of a trajectory is the discounted sum of rewards, $R(\tau) := \sum_{t=1}^T \gamma^{t-1} r_t$, where $\gamma \in [0, 1]$ is a discount factor.

A *policy* π is used to determine the probability of showing each ad. Let $\pi(a|s)$ denote the probability of taking action a in state s , regardless of the time step t . The goal is to find a policy that maximizes the expected total number of clicks per user: $\rho(\pi) := \mathbb{E}[R(\tau)|\pi]$. Our historical data is a set of trajectories, one per user. Formally, \mathcal{D} is the historical data containing n trajectories $\{\tau_i\}_{i=1}^n$, each labeled with the *behavior policy* π_i that produced it. We are also given an *evaluation policy* π_e that was produced by a RL algorithm, the performance of which we would like to evaluate.

3 Background: Off-Policy Evaluation with Probabilistic Guarantees

We recently proposed *High confidence off-policy evaluation* (HCOPE), a family of methods that use the historical data \mathcal{D} in order to find a lower-bound on the performance of the evaluation policy π_e with confidence $1 - \delta$ [Thomas *et al.*, 2015a]. In this paper, we use three different approaches to HCOPE, all of which are based on importance sampling. The *importance sampling estimator*

$$\hat{\rho}(\pi_e|\tau_i, \pi_i) := \underbrace{R(\tau_i)}_{\text{return}} \prod_{t=1}^T \underbrace{\frac{\pi_e(a_t^{\tau_i}|s_t^{\tau_i})}{\pi_i(a_t^{\tau_i}|s_t^{\tau_i})}}_{\text{importance weight}}, \quad (1)$$

is an unbiased estimator of $\rho(\pi)$ if τ_i is generated using policy π_i [Precup *et al.*, 2000]. We call $\hat{\rho}(\pi_e|\tau_i, \pi_i)$ an *importance weighted return*. Although the importance sampling estimator is conceptually easier to understand, in most of our experiments we use the *per-step importance sampling estimator*

$$\hat{\rho}(\pi_e|\tau_i, \pi_i) := \sum_{t=1}^T \gamma^{t-1} r_t \left(\prod_{j=1}^t \frac{\pi_e(a_j^{\tau_i}|s_j^{\tau_i})}{\pi_i(a_j^{\tau_i}|s_j^{\tau_i})} \right), \quad (2)$$

where the term in the parenthesis is the importance weight for the reward generated at time t . This estimator has a lower variance than (1), while it is still unbiased.

For brevity, we describe the approaches to HCOPE in terms of a set of non-negative independent random variables, $\mathbf{X} = \{X_i\}_{i=1}^n$ (note that the importance weighted returns are non-negative because the rewards are never negative). For our application, we will use $X_i = \hat{\rho}(\pi_e|\tau_i, \pi_i)$, where $\hat{\rho}(\pi_e|\tau_i, \pi_i)$ is computed either by (1) or (2). The three approaches that we will use are:

1. Concentration Inequality: Here we use the concentration inequality (CI) in Thomas *et al.* [2015a] and call it the *CI approach*. We write $\rho_{-}^{\text{CI}}(\mathbf{X}, \delta)$ to denote the $1 - \delta$ confidence lower-bound produced by their method. The benefit of this concentration inequality is that its lower-bound is a true lower-bound, i.e., it makes no false assumption or approximation, and so we refer to it as *safe*.

2. Student’s t -test: One way to tighten the lower-bound produced by the CI approach is to introduce a false but reasonable assumption. Specifically, we leverage the central limit theorem, which says that $\hat{X} := \frac{1}{n} \sum_{i=1}^n X_i$ is approximately normally distributed if n is large. Under the assumption that \hat{X} is normally distributed, we may apply the one-tailed Student’s t -test to produce $\rho_{-}^{\text{TT}}(\mathbf{X}, \delta)$, a $1 - \delta$ confidence lower-bound on $\mathbb{E}[\hat{X}]$, which in our application is a $1 - \delta$ confidence lower-bound on $\rho(\pi_e)$. Unlike the other two approaches, this approach, which we call it *TT*, requires little space to be formally defined, and so we present its formal specification:

$$\hat{X} := \frac{1}{n} \sum_{i=1}^n X_i, \quad \sigma := \sqrt{\frac{1}{n-1} \sum_{i=1}^n (X_i - \hat{X})^2},$$

$$\rho_{-}^{\text{TT}}(\mathbf{X}, \delta) := \hat{X} - \frac{\sigma}{\sqrt{n}} t_{1-\delta, n-1},$$

where $t_{1-\delta, \nu}$ denotes the inverse of the cumulative distribution function of the Student’s t distribution with ν degrees of freedom, evaluated at probability $1 - \delta$ (i.e., function `tinv(1 - δ , ν)` in MATLAB).

Because $\hat{\rho}_{-}^{\text{TT}}$ is based on a false (albeit reasonable) assumption, we refer to it as *semi-safe*. Although the *TT approach* produces tighter lower-bounds than the CI’s, it still tends to be overly conservative for our application, as discussed in Thomas *et al.* [2015b].

3. Bias Corrected and Accelerated Bootstrap: One way to correct for the overly-conservative nature of TT is to use bootstrapping to estimate the true distribution of \hat{X} , and to then assume that this estimate is the true distribution of \hat{X} . The most popular such approach is *Bias Corrected and accelerated* (BCa) bootstrap [Efron, 1987]. We write $\rho_{-}^{\text{BCa}}(\mathbf{X}, \delta)$ to denote the lower-bound produced by BCa, whose pseudocode can be found in Thomas *et al.* [2015b].

Although only semi-safe, the BCa approach produces lower-bounds on $\rho(\pi_e)$ that are actually less than $\rho(\pi_e)$ approximately $1 - \delta$ percent of the time, as opposed to the TT and CI approaches, which produce lower-bounds on $\rho(\pi_e)$ that are less than $\rho(\pi_e)$ much more than $1 - \delta$ percent of the time. Although BCa is only semi-safe (it can produce an error rate above $1 - \delta$), it has been considered reliable

enough to be used for many applications, particularly in medical fields [Champliss *et al.*, 2003; Folsom *et al.*, 2003].

4 CTR versus LTV

Any personalized ad recommendation (PAR) policy could be evaluated for its greedy/myopic or long-term performance. For greedy performance, click through rate (CTR) is a reasonable metric, while life-time value (LTV) seems to be the right choice for long-term performance. These two metrics are formally defined as

$$\text{CTR} = \frac{\text{Total \# of Clicks}}{\text{Total \# of Visits}} \times 100,$$

$$\text{LTV} = \frac{\text{Total \# of Clicks}}{\text{Total \# of Visitors}} \times 100.$$

CTR is a well-established metric in digital advertising and can be estimated from historical data (off-policy) in unbiased (inverse propensity scoring; Li *et al.* [2010]) and biased (see e.g., Strehl *et al.* [2010]) ways. In this paper, we extend our recently proposed practical approach for LTV estimation [Thomas *et al.*, 2015a], by replacing the concentration inequality with both t -test and BCa, and apply them for the first time to real online advertisement data. The main reason that we use LTV is that CTR is not a good metric for evaluating long-term performance and could lead to misleading conclusions. Imagine a greedy advertising strategy at a website that directly displays an ad related to the final product that a user could buy. For example, it could be the BMW website and an ad that offers a discount to the user if she buys a car. Users who are presented such an offer would either take it right away or move away from the website. Now imagine another marketing strategy that aims to transition the user down a sales funnel before presenting her the discount. For example, at the BMW website one could be first presented with an attractive finance offer and a great service department deal before the final discount being presented. Such a long-term strategy would incur more interactions with the customer and would eventually produce more clicks per customer and more purchases. The crucial insight here is that the policy can change the number of times that a user will be shown an advertisement—the length of a trajectory depends on the actions that are chosen. A visualization of this concept is presented in Figure 1.

5 Ad Recommendation Algorithms

For greedy optimization, we used a random forest (RF) algorithm [Breiman, 2001] to learn a mapping from features to actions. RF is a state-of-the-art ensemble learning method for regression and classification, which is relatively robust to overfitting and is often used in industry for big data problems. The system is trained using a RF for each of the offers/actions to predict the immediate reward. During execution, we use an ϵ -greedy strategy, where we choose the offer whose RF has the highest predicted value with probability $1 - \epsilon$, and the rest of the offers, each with probability $\epsilon/(|A| - 1)$ (see Algorithm 1).

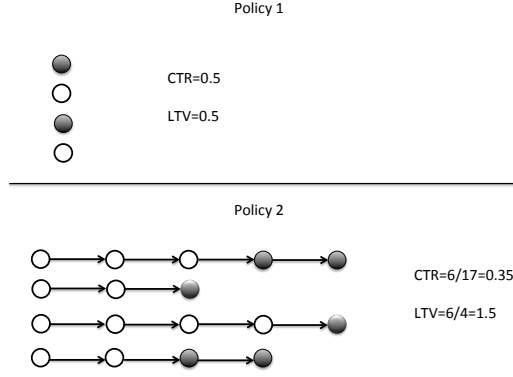


Figure 1: The circles indicate user visits. The black circles indicate clicks. Policy 1 is greedy and users do not return. Policy 2 optimizes for the long-run, users come back multiple times, and click towards the end. Even though Policy 2 has a lower CTR than Policy 1, it results in more revenue, as captured by the higher LTV. Hence, LTV is potentially a better metric than CTR for evaluating ad recommendation policies.

Algorithm 1 GREEDYOPTIMIZATION($\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{test}}, \delta, \epsilon$) : compute a greedy strategy using $\mathbf{X}_{\text{train}}$, and predict the $1 - \delta$ lower bound on the test data \mathbf{X}_{test} and the value function.

- 1: $y = \mathbf{X}_{\text{train}}(\text{reward})$
- 2: $x = \mathbf{X}_{\text{train}}(\text{features})$
- 3: $\bar{x} = \text{informationGain}(x, y)$ {feature selection}
- 4: $\text{rf}_a = \text{randomForest}(\bar{x}, y)$ {for each action}
- 5: $\pi_e = \text{epsilonGreedy}(\text{rf}, \mathbf{X}_{\text{test}})$
- 6: $\pi_b = \text{randomPolicy}$
- 7: $W = \hat{\rho}(\pi_e | \mathbf{X}_{\text{test}}, \pi_b)$ {importance weighted returns}
- 8: **return** $(\rho_{-}^{\dagger}(W, \delta), \text{rf})$ {bound and random forest}

For LTV optimization, we used a state-of-the-art RL algorithm, called FQI [Ernst *et al.*, 2005], with RF function approximator, which allows us to handle high-dimensional continuous and discrete variables. When an arbitrary function approximator is used in the FQI algorithm, it does not converge monotonically, but rather oscillates during training iterations. To alleviate the oscillation problem of FQI and for better feature selection, we used our high confidence off-policy evaluation (HCOPE) framework within the training loop. The loop keeps track of the best FQI result according to a validation data set (see Algorithm 2).

Both algorithms are described graphically in Figure 2. For both algorithms we start with three data sets an $\mathbf{X}_{\text{train}}$, \mathbf{X}_{val} and \mathbf{X}_{test} . Each one is made of complete user trajectories. A user only appears in one of those files. The \mathbf{X}_{val} and \mathbf{X}_{test} contain users that have been served by the random policy. The greedy approach proceeds by first doing feature selection on the $\mathbf{X}_{\text{train}}$, training a random forest, turning the policy into ϵ -greedy on the \mathbf{X}_{test} and then evaluating that policy using the off-policy evaluation techniques. The LTV approach starts

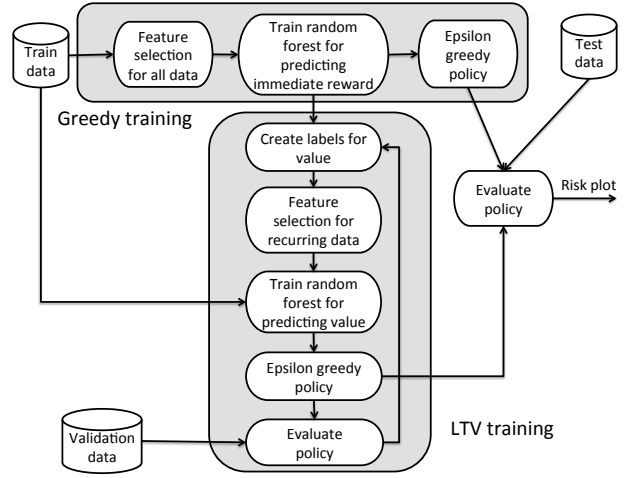


Figure 2: This figure shows the flow for training Greedy and LTV strategies.

from the random forest model of the greedy approach. It then computes labels as shown in step 6 of the LTV optimization algorithm 2. It does feature selection, trains a random forest model, and then turns the policy into ϵ -greedy on the \mathbf{X}_{val} data set. The policy is tested using the importance weighted returns Equation 2. LTV optimization loops over a fixed number of iterations and keeps track of the best performing policy, which is finally evaluation on the \mathbf{X}_{test} . The final outputs are “risk plots”, which are graphs that show the lower-bound of the expected sum of discounted reward of the policy for different confidence values.

Algorithm 2 LTVOPTIMIZATION($\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{val}}, \mathbf{X}_{\text{test}}, \delta, K, \gamma, \epsilon$) : compute a LTV strategy using $\mathbf{X}_{\text{train}}$, and predict the $1 - \delta$ lower bound on the test data \mathbf{X}_{test}

- 1: $\pi_b = \text{randomPolicy}$
- 2: $Q = \text{RF.GREEDY}(\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{test}}, \delta)$ {start with greedy value function}
- 3: **for** $i = 1$ **to** K **do**
- 4: $r = \mathbf{X}_{\text{train}}(\text{reward})$ {use recurrent visits}
- 5: $x = \mathbf{X}_{\text{train}}(\text{features})$
- 6: $y = r_t + \gamma \max_{a \in A} Q_a(x_{t+1})$
- 7: $\bar{x} = \text{informationGain}(x, y)$ {feature selection}
- 8: $Q_a = \text{randomForest}(\bar{x}, y)$ {for each action}
- 9: $\pi_e = \text{epsilonGreedy}(Q, \mathbf{X}_{\text{val}})$
- 10: $W = \hat{\rho}(\pi_e | \mathbf{X}_{\text{val}}, \pi_b)$ {importance weighted returns}
- 11: $\text{currBound} = \rho_{-}^{\dagger}(W, \delta)$
- 12: **if** $\text{currBound} > \text{prevBound}$ **then**
- 13: $\text{prevBound} = \text{currBound}$
- 14: $Q_{\text{best}} = Q$
- 15: **end if**
- 16: **end for**
- 17: $\pi_e = \text{epsilonGreedy}(Q_{\text{best}}, \mathbf{X}_{\text{test}})$
- 18: $W = \hat{\rho}(\pi_e | \mathbf{X}_{\text{test}}, \pi_b)$
- 19: **return** $\rho_{-}^{\dagger}(W, \delta)$ {lower bound}

6 Experiments

For our experiments we used 2 data sets from the banking industry. On the bank website when customers visit, they are shown one of a finite number of offers. The reward is 1 when a user clicks on the offer and 0, otherwise. We extracted/created features, in the categories shown in Table 1. For data set 1, we collected data from a particular campaign of a bank for a month that had 7 offers and approximately 200,000 interactions. About 20,000 of the interactions were produced by a random strategy. For data set 2 we collected data from a different bank for a campaign that had 12 offers and 4,000,000 interactions, out of which 250,000 were produced by a random strategy. When a user visits the bank website for the first time, she is assigned either to a random strategy or a targeting strategy for the rest of the campaign life-time. We splitted the random strategy data into a test set and a validation set. We used the targeting data for training to optimize the greedy and LTV strategies described in Algorithms 1 and 2. We used aggressive feature selection for the greedy strategy and selected 20% of the features. For LTV, the feature selection had to be even more aggressive due to the fact that the number of recurring visits is approximately 5%. We used information gain for the feature selection module [Tiejun *et al.*, 2012]. With our algorithms we produce performance results both for the CTR and LTV metrics. To produce results for CTR we assumed that each visit is a unique visitor.

<i>Cum action</i>	There is one variable for each offer, which counts the number of times each offer was shown
<i>Visit time recency</i>	Time since last visit
<i>Cum success</i>	Sum of previous reward
<i>Visit</i>	The number of visits so far
<i>Success recency</i>	The last time there was positive reward
<i>Longitude</i>	Geographic location [Degrees]
<i>Latitude</i>	Geographic location [Degrees]
<i>Day of week</i>	Any of the 7 days
<i>User hour</i>	Any of the 24 hours
<i>Local hour</i>	Any of the 24 hours
<i>User hour type</i>	Any of weekday-free, weekday-busy, weekend
<i>Operating system</i>	Any of unknown, windows, mac, linux
<i>Interests</i>	There are finite number of interests for each company. Each interest is a variable that gets a score according to the content of areas visited within the company websites
<i>Demographics</i>	There are many variables in this category such as age, income, home value...

Table 1: Features

We performed various experiments to understand the different elements and parameters of our algorithms. For all ex-

periments we set $\gamma = 0.9$ and $\epsilon = 0.1$.

Experiment 1: How do LTV and CTR compare? For this experiment we show that every strategy has both a CTR and LTV metric as shown in Figure 3. In general the LTV metric gives higher numbers than the CTR metric. Estimating the LTV metric however gets harder as the trajectories get longer and as the mismatch with the behavior policy gets larger. In this experiment the policy we evaluated was the random policy which is the same as the behavior policy, and in effect we eliminated the importance weighted factor.

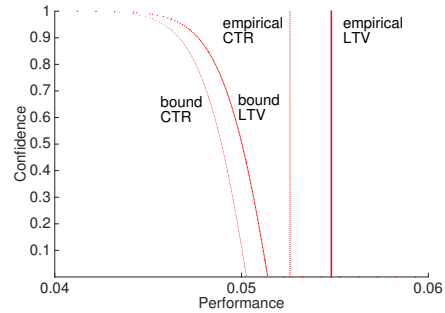


Figure 3: This figure shows the bounds and empirical importance weighted returns for the random strategy. It shows that every strategy has both a CTR and LTV metric. This was done for data set 1.

Experiment 2: How do the three bounds differ? In this experiment we compared the 3 different lower-bound estimation methods, as shown in Figure 4. We observed that the bound for the t -test is tighter than that for CI, but it makes the false assumption that importance weighted returns are normally distributed. We observed that the bound for BCa has higher confidence than the t -test approach for the same performance. The BCa bound does not make a Gaussian assumption, but still makes the false assumption that the distribution of future empirical returns will be the same as what has been observed in the past.

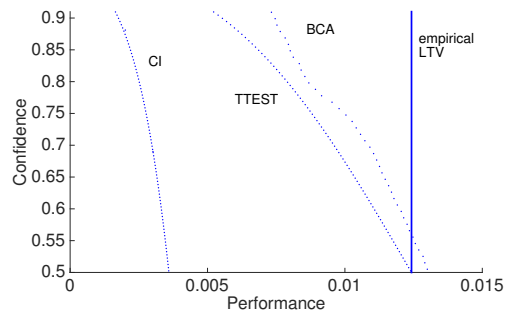


Figure 4: This figure shows comparison between the 3 different bounds. It was done for data set 2.

Experiment 3: When should each of the two optimization algorithms be used? In this experiment we observed that the GREEDYOPTIMIZATION algorithm performs the best under the CTR metric and the LTVOPTIMIZATION algorithm performs the best under the LTV metric as expected, see Figures 5 and 6. The same claim holds for data set 2.

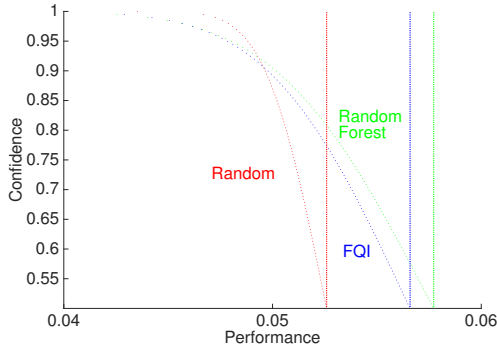


Figure 5: This figure compares the CTR bounds of the Greedy versus the LTV optimization It was done for data set 1, but similar graphs exist for data set 2.

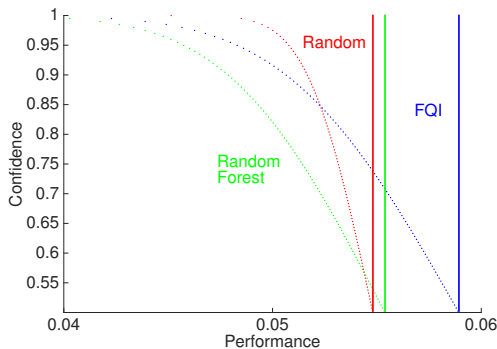


Figure 6: This figure compare the LTV bounds of the Greedy versus the LTV optimization It was done for data set 1, but similar graphs exist for data set 2.

Experiment 4: What is the effect of ϵ ? One of the limitations of our algorithm is that it requires stochastic policies. The closer the new policy is to the behavior policy the easier to estimate the performance. Therefore, we approximate our policies with ϵ -greedy and use the random data for the behavior policy. The larger the ϵ , the easier is to get a more accurate performance of a new policy, but at the same time we would be estimating the performance of a sub-optimal policy, which has moved closer to the random policy, see Figure 7. Therefore, when using this the bounds to compare two policies, such as Greedy vs. LTV, one should use the same ϵ .

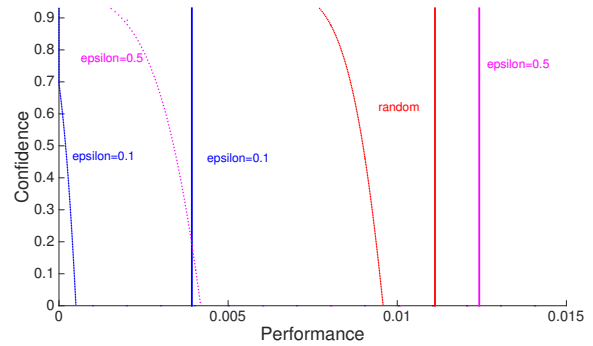


Figure 7: The figure shows that as epsilon gets larger the policy moves towards the random policy. Random policies are easy to estimate their performance since they match the behavior policy exactly. Thus epsilon should be kept same when comparing two policies. This experiment was done on data set 2 and shows the bounds and empirical mean importance weighted returns (vertical line) for the LTV policy. The bound used here was the CI.

7 Summary and Conclusions

In this paper, we presented a framework for training and evaluating personal ad recommendation (PAR) strategies. This framework is mainly based on a family of high confidence off-policy evaluation (HCOPE) techniques that we have recently developed [Thomas *et al.*, 2015a,b]. Our main contribution is using these HCOPE techniques together with RL algorithms to learn and evaluate PAR strategies that optimize for customers' life-time value (LTV). However, these HCOPE techniques can also be used to evaluate the performance of a myopic strategy that optimizes for click through rate (CTR), and to provide high confidence bounds for it. We provided extensive experiments with data sets generated from real-world PAR campaigns to show the effectiveness of our proposed framework and to clarify some of the issues raised and discussed in the paper such as LTV vs. myopic optimization, CTR vs. LTV performance measures, and the merits of using high-confidence off-policy evaluation techniques in learning and evaluating RL policies.

Overall, we can summarize the main contributions of this work as follows: **1)** Unlike most existing work on PAR systems, we tackled the problem of LTV recommendation and showed how our approach leads to desirable results, i.e., we were able to produce good results for a real PAR campaign with a relatively small data set of historical data. **2)** We identified the relationship between CTR and LTV and empirically demonstrated why CTR may not be a good metric to measure the performance of a PAR system with many returning visitors. **3)** To the best of our knowledge, this is the first work that optimizes LTV of a real-world PAR system and provides guarantees on the performance of the learned strategy. **4)** We combined state-of-the-art ingredients such as HCOPE methods, the power and robustness of random-forest regression, and aggressive feature selection to devise algorithms that efficiently learn a PAR policy with either a good CTR or LTV performance measure.

References

- L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.
- L. Chambless, A. Folsom, A. Sharrett, P. Sorlie, D. Couper, M. Szklo, and F. Nieto. Coronary heart disease risk prediction in the Atherosclerosis Risk in Communities (ARIC) study. *Journal of Clinical Epidemiology*, 56(9):880–890, 2003.
- B. Efron. Better bootstrap confidence intervals. *Journal of the American Statistical Association*, 82(397):171–185, 1987.
- D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.
- A. Folsom, L. Chambless, B. Duncan, A. Gilbert, and J. Pankow. Prediction of coronary heart disease in middle-aged adults with diabetes. *Diabetes Care*, 26(10):2777–2784, 2003.
- J. Jonker, N. Piersma, and D. Van den Poel. Joint optimization of customer segmentation and marketing policy to maximize long-term profitability. *Expert Systems with Applications*, 27(2):159 – 168, 2004.
- J. Langford, L. Li, and M. Dudk. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1097–1104, 2011.
- L. Li, W. Chu, J. Langford, and R. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670, 2010.
- E. Pednault, N. Abe, and B. Zadrozny. Sequential cost-sensitive decision making with reinforcement learning. In *Proceedings of the eighth international conference on Knowledge discovery and data mining*, pages 259–268, 2002.
- P. Pfeifer and R. Carraway. Modeling customer relationships as markov chains. *Journal of interactive marketing*, pages 43–55, 2000.
- D. Precup, R. S. Sutton, and S. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the 17th International Conference on Machine Learning*, pages 759–766, 2000.
- D. Silver, L. Newnham, D. Barker, S. Weller, and J. McFall. Concurrent reinforcement learning from customer interactions. In *In 30th International Conference on Machine Learning*, 2013.
- A. Strehl, J. Langford, L. Li, and S. Kakade. Learning from logged implicit exploration data. In *Proceedings of Neural Information Processing Systems 24*, pages 2217–2225, 2010.
- R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
- G. Theodorou and A. Hallak. Lifetime value marketing using reinforcement learning. In *The 1st Multidisciplinary Conference on Reinforcement Learning and Decision Making*, 2013.
- P. S. Thomas, G. Theodorou, and M. Ghavamzadeh. High confidence off-policy evaluation. In *Proceedings of the Twenty-Ninth Conference on Artificial Intelligence*, 2015.
- P. S. Thomas, G. Theodorou, and M. Ghavamzadeh. High confidence policy improvement. In *Proceedings of the International Conference on Machine Learning*, 2015.
- C. Tiejun, W. Yanli, and B. Stephen H. Fselector. *Bioinformatics*, 28(21):2851–2852, November 2012.
- G. Tirenni, A. Labbi, C. Berrospi, A. Elisseeff, T. Bhoose, K. Pauro, and S. Poyhonen. The 2005 ISMS Practice Prize Winner Customer-Equity and Lifetime Management (CELM) Finnair Case Study. *Marketing Science*, 26:553–565, 2007.
- W. Venables and B. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002.