Bandit Data-driven Optimization: AI for Social Good and Beyond

Anonymous Author(s)
Affiliation
Address
email

Abstract

The use of machine learning (ML) systems in real-world applications entails more than just a prediction algorithm. AI for social good (AI4SG) applications, and many real-world ML tasks in general, feature an iterative process that joins prediction, optimization, and data acquisition in a loop. We introduce bandit data-driven optimization, the first iterative prediction-prescription framework to combine the advantages of online bandit learning and offline predictive analytics. It offers a flexible setup to reason about unmodeled policy objectives and unforeseen consequences. We propose PROOF, the first algorithm for this framework and show that it achieves no-regret. PROOF achieves superior performance over existing baseline in numerical simulations.

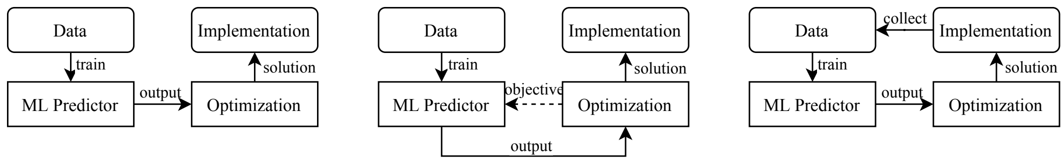
1 Introduction

The success of modern ML often does not directly translate into a perfect solution to a real-world AI4SG problem [17], or more generally, data-driven policy-making. One obvious reason is that supervised learning focuses on prediction, yet real-world problems, by and large, need prescription. For example, rather than predict which households' water pipes are contaminated (labels) using construction and demographic information (features), municipal officials need to know how to schedule inspections and replacements (interventions) [2]. The common practice is a two-stage procedure, as shown in Figure 1a. After training an ML prediction model, the user makes prescriptive decisions based on some optimization problem or even simple heuristic which takes the prediction output as parameters. The training objective and the optimization objective are completely separate, which means it is hard to control the final prescription quality. In an emerging line of work on (one-shot) data-driven optimization, the learning problem is made aware of the downstream optimization objective through its loss function, and hence mitigating this issue [7, 11]. We illustrate this in Figure 1b.

However, this is still far from the complete picture of a real-world problem. Figure 1c shows a typical workflow in many AI4SG applications. After getting data from the collaborating organization, the researcher trains a predictive model and then, based on it, makes an intervention recommendation. The workflow does not stop here, though. After the organization implements the recommended intervention, it collects more data points. Using these additional data, the researcher then updates the predictive model and makes a new intervention recommendation to be implemented, so on and so forth, resulting in an iterative process. The guiding principles of the various components in this process are often not aligned. Without a rigorous, integrated framework to guide the procedure, this could lead to operation inefficiency, missed expectations, dampened initiatives, and new barriers of mistrust which are not meant to be.

Why is such an iterative process necessary? First, many "social good" domains do not have the luxury of millions of training examples. A small dataset leads to inaccurate predictions and hence

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.



(a) Two-stage prediction-prescription (b) One-shot data-driven optimization (c) Bandit data-driven optimization

Figure 1: Paradigms of how ML systems are used in realistic policy-making.

suboptimal decisions. Second, too often the initial dataset has some default intervention embedded, while the project's goal is to find the optimal intervention. It is necessary to at least try out some interventions and collect data under them. Third, the communication gap between researchers and practitioners makes it difficult to formulate the right objective at the beginning. Fourth, interventions may have unexpected consequences, thus the inherent impossibility of fully modeling the problem in one shot.

We propose the first iterative prediction-prescription framework, which we term as bandit data-driven optimization. Bandit data-driven optimization combines the relative advantages of both online bandit learning and offline predictive analytics. We achieve this with our algorithm PRedict-then-Optimize with Optimism in Face of uncertainty (PROOF). PROOF is a modular algorithm which can work with a variety of predictive models and optimization problems. Under specific settings, we formally analyze its performance and show that PROOF achieves no-regret. In addition, we propose a variant of PROOF which can handle the scenario where the intervention affects the data distribution, which also enjoys no-regret. Finally, we use numerical simulations to show that PROOF achieves superior performance than a pure bandit baseline.

We emphasized AI4SG as the motivation and application domain of bandit data-driven optimization, because we have experienced these problems first-hard in our applied work on AI4SG. However, none of the reasons above is exclusive to AI4SG. Many data science projects in the real world, whether explicitly for social good or not, fit these characterizations and can benefit from such a framework. In this sense, our work can be viewed as a general framework for data-driven policy-making.

2 Related Work

There are many paradigms in which ML is used in a real-world problem, yet the paradigm illustrated in Figure 1c is certainly a common one. To our knowledge, there is surprisingly no existing work that rigorously studies this procedure. We propose bandit data-driven optimization as the first step towards formalizing and improving the way machine learning is used in AI4SG projects. That said, several research topics also concern themselves with similar problems in the real-life usage of ML systems. We introduce them below and compare each of them with bandit data-driven optimization.

First, (one-shot) data-driven optimization is an emerging line of research in the operations research literature. Given a dataset consisting of features $x_1, \ldots, x_n$ and labels $c_1, \ldots, c_n$, the task is to find the action w^* that maximizes the expected utility given some feature x, i.e. $w^* = \arg\max_w \mathbb{E}_{c|x}[p(c, w)]$. There are two major approaches. The first one comes from the stochastic programming perspective [7, 6]. For example, Bertsimas and Kallus fuse the prediction and optimization by transforming the optimization objective into a weighted combination of cost, where the weight, determined by some ML algorithm, represents the similarity between the target feature x and each of the feature x_i's in the dataset. Another popular approach is generally referred to as the predict-then-optimize framework [13, 11, 14]. The idea is to fit an ML predictor f from the feature x to label c, and then use the prediction $c = f(x)$ in the optimization problem. To connect the training and optimization, typically the loss function in the ML problem is modified to reflect the downstream optimization objective. Compared to bandit data-driven optimization, this entire literature assumes that the optimization objective is known a priori and does not consider multi-period settings. This limits its applicability in reality.

Speaking of multi-period settings, contextual bandit is a well-studied online decision-making model which is very relevant to our framework [5, 15]. At each time step t of the contextual bandit, we receive feature x_t, pick an action w_t, and receive reward whose expectation $q(x_t, w_t)$ is some

Procedure 1: BANDIT DATA-DRIVEN OPTIMIZATION

1 Receive initial dataset $\mathcal{D} = \{(x_i^0, c_i^0; w_i^0)_{i=1,\ldots,n}\}$ from distribution D on (X, C).
2 **for** $t = 1, 2, \ldots, T$ **do**
3 Using all the available data $\mathcal{D}$, train ML prediction model $f_t : X \rightarrow C$.
4 Given n feature samples $\{x_i^t\} \sim D_x$, choose interventions $w^t = \{w_i^t\}$ for each individual i.
5 Receive n labels $\{c_i^t\} \sim D(w_i^t)_{c|x_i^t}$. Add $\{(x_i^t, c_i^t; w_i^t)_{i=1,\ldots,n}\}$ to the dataset $\mathcal{D}$.
6 Get cost $u_t = u(C^t, w^t) = \sum_i p(C_i^t, w_i^t) + q(w_i^t) + \eta$, where $\eta \sim N(0, \sigma^2)$.

unknown function of x_t and w_t. In fact, contextual bandit is even more general than bandit data-driven optimization, because we could simply ignore the label c, forego the training of a predictive model, and let the bandit algorithm pick an action. However, by doing so, we effectively give up all the valuable information in the historical data, finding the optimal action in a purely online fashion. Although contextual bandit algorithms often achieve no-regret and have been used for high-frequency decision-making [16], they are often impractical in AI4SG applications. It would hardly be acceptable to any AI4SG stakeholders that our algorithm only guarantees good results after using it for, say, 10 years, while the algorithm chooses not to use the dataset already available. That being said, bandit provides a proper setting for sequential decision making under uncertainty and bandit algorithms like LinUCB [9, 8, 1] play a central role in designing algorithms for bandit data-driven optimization.

Also related to our problem is offline policy learning [19, 10, 4], which shares similar goals with bandit data-driven optimization. It aims at finding the best policy $\pi(w|x)$, which is a distribution over actions for each given feature, using only historical action records. Its main advantage over its online counterpart, i.e. contextual bandit, is that it does not need even a single online trial, and hence is much easier to convince the stakeholder to adopt. However, this advantage comes with the assumption that the historical data has many different actions attempted. This assumption often fails to hold, at least in the AI4SG projects we have worked on. Furthermore, most of this literature focus on the binary action setting and it, similar to contextual bandits, does not explicitly use the feature/label dataset.

3 Bandit Data-driven Optimization

We describe the formal setup of bandit data-driven optimization in Procedure 1. On Line 1, we receive an initial dataset $\mathcal{D}$ of size n, consisting of features x and labels c. Each feature vector x_i^0 is drawn i.i.d. from an unknown distribution D_x. Each label vector c_i^0 is independently drawn from an unknown conditional distribution $D(w_i^0)_{c|x_i^0}$. The conditional distribution is indexed by intervention w, which means different interventions could potentially lead to different data distributions. In this case, w_i^0 is the default intervention which has been in-place with the data collector. On Line 3, we use all the data collected so far to train a machine learning model f_t, a mapping from features $X \subseteq \mathbb{R}^m$ to labels $C \subseteq \mathbb{R}^d$. On Line 4, we draw a new sample of features $\{x_i^t\}_{i=1}^n$. Then, we select an intervention $w_i^t \in W$ for each individual i. We have a known loss function $p(c, w)$ which represents our modeling effort and domain knowledge. We want to minimize its expectation given the features x_i^t. On Line 5, we commit to intervention w^t and then get the labels $\{c_i^t\}_{i=1}^n$ corresponding to the features drawn earlier. Subsequently, on Line 6, we incur a cost u_t. The cost consists of our known optimization objective $p(\cdot)$ and an unknown cost $q(\cdot)$ that represents all the unmodeled policy objectives and the unintended consequences. It comes with a random noise.

A few remarks are in order. First, a common question is why do we need an iterative procedure at all? From the real-world application perspective, data are often hard to acquire in AI4SG domains, so frequently we have no choice but to collect data as we roll out the intervention programs. Even when we seem to have enough data, it is still wise to keep collecting data and updating our model, because the data distribution might change over time, and continuous engagement is an important factor in AI4SG projects. The iterative procedure is also essential for exploration purpose. Bandit data-driven optimization contains two types of exploration-exploitation trade-offs. The historical data often contains only a single default intervention. If we do not explore and implement some other interventions, we will never learn how good (or bad) they are. The second exploration arises when dealing with the unknown cost component $q(\cdot)$, which is essentially a bandit problem in itself. More discussion on $q(\cdot)$ follows below.

Second, this form of loss – a known part $p(\cdot)$ and an unknown part $q(\cdot)$ – is a realistic compromise of two extremes. AI4SG researchers often spend a considerable amount of time communicating with domain experts to understand the problem. It would go against this honest effort to eliminate $p(\cdot)$ and model the process as a pure bandit problem. On the other hand, even when we take our completed work to the deployment stage, there will still be unmodeled objectives. Thus, it would be too arrogant to eliminate $q(\cdot)$ and to pretend that anything that does not go according to the plan is noise. The unknown $q(\cdot)$ is also our conscious acknowledgement that any intervention recommended by AI4SG projects may have unintended consequences. We believe that having both $p(\cdot)$ and $q(\cdot)$ is a faithful first step towards capturing the nature of AI4SG work.

Let us use two AI4SG examples to illustrate how bandit data-driven optimization captures real-world ML policy-making workflows.

Food Rescue A food rescue (FR) organization receives food donations from restaurants and grocery stores and connects them to low-resource community organizations. Dispatchers at FR would post the donor and recipient information on their mobile app, and some volunteer would claim the rescue and pick up and deliver the donations. To ensure that each rescue gets claimed, the dispatcher sets mobile app push notification time for each rescue, which is the intervention w, to alert the volunteers of available rescues. This decision is dependent on how likely a rescue will be claimed. Thus, we can develop a machine learning model which uses features x of a rescue, e.g. donor/recipient location, weather, and time of day, to predict the probability that a rescue will be claimed in 10 minutes, 30 minutes, etc. (label c). The optimization objective $p(c, w)$ is to minimize the expected wait time, while still guaranteeing not a lot of push notifications are sent. After we select a w for a rescue, we observe the label and this data point will be used for training before the next rescue trip comes along. The cost to the FR may include factors other than the rescue claim rate, e.g. gain/loss of registered volunteers as a result of push notification scheme. Of course we know this now, but the $q(\cdot)$ cost could capture similar factors which we do not know yet. For more background about the food rescue operation, the reader may refer to the work of Shi et al. [18].

Anti-poaching Wildlife poaching is a pressing problem in many parts of the world. Given that poachers are often experienced and shrewd, wildlife rangers need to patrol the vast area of a wildlife park intelligently. Indeed, this battle between rangers and poachers have received considerable interest from the computer science community. This is typically studied as a Stackelberg security game, which is essentially an optimization problem to find the best ranger patrolling strategy (intervention w). To solve the optimization problem, we need to know the poacher's behavior pattern, for which we will develop an ML model. The ML model takes as input the location, geographic feature, animal density, etc., of a small patch of land (feature x) and predicts the likelihood that the poacher will poach at this location (label c). Given this likelihood, we can select the optimal ranger patrolling strategy w which minimizes the likelihood of successful poaching. Every time a poacher or their belonging is caught, we can add a new data point to the dataset. The cost to the rangers may include factors other than the poached animals themselves, such as the crime rate in the region, as the enforcement of anti-poaching might lead to the increase of other crimes. $q(\cdot)$ could capture similar factors which we do not know yet. For more background about the game-theoretic work on anti-poaching, the reader may refer to the work of Fang et al. [12].

4 PRedict-then-Optimize with Optimism in Face of uncertainty (PROOF)

We propose the first algorithm for the bandit data-driven optimization, PRedict-then-Optimize with Optimism in Face of uncertainty (PROOF), which is shown in Algorithm 2.

Unless otherwise specified, we work with the following setting. The data points are drawn from $X \times C$ where $X \subseteq \mathbb{R}^m$ and $C \subseteq \mathbb{R}^d$. We assume all $x \in X$ has l^2-norm bounded by constant K_X, and the label space C has l^1-diameter K_C. The action space W could be either discrete or continuous, but is bounded inside the unit l^2-ball in $\mathbb{R}^d$. We specify the data distribution by an arbitrary marginal distribution D_x on X and a conditional distribution such that $c = f(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, for some function f. Thus, $\mathbb{E}_{c|x}[c] = f(x)$. Of course, f is unknown to us and needs to be learned. To begin with, we assume $f \in \mathcal{F}$ comes from the class of all linear functions with $f(x) = Fx$, and we use ordinary least squares regression as the learning algorithm. We will relax this assumption towards the end of Section 4. The known cost $p(c, w) = c^\dagger w$ is the inner product of label c and

Algorithm 2: PROOF: PREDICT-THEN-OPTIMIZE WITH OPTIMISM IN FACE OF UNCERTAINTY

1 **Initialize:**
2   Find a barycentric spanner $b_1, \ldots, b_d$ for W
3   Set $A_i^1 = \sum_{j=1}^d b_j b_j^\dagger$ and $\hat{\mu}_i^1 = 0$ for all i
4 Receive initial dataset $\mathcal{D} = \{(x_i^0, c_i^0; w_i^0)_{i=1,\ldots,n}\}$.
5 **for** $t = 1, 2, \ldots, T$ **do**
6   Using all the available data $\mathcal{D}$, train ML prediction model $f_t : X \to C$.
7   Given n feature samples $\{x_i^t\} \sim D_x$, get predictions $\hat{c}_i^t = f_t(x_i^t)$.
8   Set confidence ball radius $\beta^t = \max\left(128 d \log t \log(nt^2/\gamma), \left(\frac{8}{3}\log\left(\frac{nt^2}{\gamma}\right)\right)^2\right)$
9   **for** $i = 1, 2, \ldots, n$ **do**
10   Set CB $B_i^t = \{\nu : ||\nu - \hat{\mu}_i^t||_{2,A_i^t} \leq \sqrt{\beta^t}\}$.
11   Choose intervention w_i^t where $w_i^t = \arg\min_{w \in W} \min_{\nu \in B_i^t} (\hat{c}_i^t + \nu)^\dagger w$.
12   Get label $c_i^t \sim D_{c|x_i^t}$. Add $(x_i^t, c_i^t; w_i^t)$ to $\mathcal{D}$.
13   Get cost $u_i^t = (c_i^t)^\dagger w_i^t + \mu^\dagger w_i^t + \eta_i$, where $\eta_i \sim N(0, \sigma^2)$. In particular, let u_{oi}^t be the first term and let u_{bi}^t be the sum of the second and third term.
14   Update $A_i^{t+1} = A_i^t + w_i^t (w_i^t)^\dagger$
15   Update $\hat{\mu}_i^{t+1} = (A_i^{t+1})^{-1} \sum_{\tau=1}^t u_{bi}^t w_i^t$

action w.[1] The unknown cost is $q(w) = \mu^\dagger w$, where μ is an unknown but fixed vector. Furthermore, for exposition purpose we will start by assuming that the intervention w does not affect the data distribution. We will later remove this assumption and present the algorithm for the general case.

PROOF is a delicate integration of the celebrated Optimism in Face of Uncertainty (OFU) framework [9] and the predict-then-optimize framework. It is clear that the unknown cost component $q(\cdot) + \eta$ forms a linear bandit. For this bandit component, we run an OFU algorithm for each individual i with the same unknown loss vector μ. The OFU component for each individual i maintains a confidence ball (B_i^t) which is independent of the predict-optimize framework. The predict-then-optimize framework produces an estimated optimization objective (represented by $\hat{c}^t$) that has nothing to do with the OFU. The two components are integrated together on Line 11 of Algorithm 2, where we compute the intervention for the current round taking into consideration the essence of both frameworks.

Below, we justify why this algorithm achieves no-regret. First, we state a theorem by Dani et al. [9], which states that the confidence ball captures the true loss vector μ with high probability. Our only modification is a trivial union bound so that the result holds for all the n bandits simultaneously.

Lemma 1 (Adapted from Theorem 5 by Dani et al. [9])**.** *Let* $\gamma > 0$*,*

$$\mathbb{P}(\forall t, \forall i, \mu \in B_i^t) \geq 1 - \gamma.$$

This lemma was proved for the original OFU algorithm. However, the lemma itself does not depend on the way we choose w^t at each time step, as long as we computed $\hat{\mu}$ and A in the way we did (same as OFU) on Lines 14-15. That is, it does not matter that we have an additional $\hat{c}_i^t$ in the optimization on Line 11, compared to the original OFU. We state another useful result below.[2]

Lemma 2. *Suppose we use the ordinary least squares regression as the ML algorithm. The prediction error is*

$$\mathbb{E}_{X,\epsilon}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2\right] = O\left(\sqrt{\frac{dm}{nt}}\right).$$

Theorem 3. *Assuming we use ordinary least squares regression as the ML algorithm, PROOF has regret* $\tilde{O}\left(n\sqrt{dmT}\right)$ *with probability* $1 - \delta$*.*

[1] To avoid confusion, in this paper we use superscript † to denote matrix and vector transpose.

[2] Please refer to the full paper for complete proofs and additional technical results. https://www.dropbox.com/s/92nk7zolycva90h/BDO_anon.pdf?dl=0

Proof. Let $w_{i*}^t = \arg\min_w (\mathbb{E}_{c_i^t|x_i^t}[c_i^t] + \mu)^\dagger w$. w_{i*}^t is the optimal action for individual i at time t, and is the benchmark in our regret computation.

Fix i, fix t. Let $\tilde{\nu} = \arg\min_{\nu \in B_i^t} (\hat{c}_i^t + \nu)^\dagger w_i^t$. Because of Line 11, we have

$$(\hat{c}_i^t + \tilde{\nu})^\dagger w_i^t = \min_{\nu \in B_i^t, w \in W} (\hat{c}_i^t + \nu)^\dagger w \leq (\mathbb{E}_{c_i^t|x_i^t}[c_i^t] + \mu)^\dagger w_{i*}^t + (\hat{c}_i^t)^\dagger w_{i*}^t - \mathbb{E}_{c_i^t|x_i^t}[c_i^t]^\dagger w_{i*}^t.$$

The inequality above used the fact that $\mu \in B_i^t$, by Lemma 1. Thus, we get the per-round regret

$$\begin{aligned}(\mathbb{E}_{c_i^t|x_i^t}[c_i^t] + \mu)^\dagger (w_i^t - w_{i*}^t) &\leq (\mathbb{E}_{c_i^t|x_i^t}[c_i^t] + \mu)^\dagger w_i^t - (\hat{c}_i^t + \tilde{\nu})^\dagger w_i^t + (\hat{c}_i^t)^\dagger w_{i*}^t - \mathbb{E}_{c_i^t|x_i^t}[c_i^t]^\dagger w_{i*}^t \\ &= (\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t)^\dagger (w_i^t - w_{i*}^t) + (\mu - \tilde{\nu})^\dagger w_i^t\end{aligned}$$

We can view the second term is the per-round regret for the bandit part. By Theorem 6 in Dani et al. [9], we have

$$\sum_{t=1}^T ((\mu - \tilde{\nu})^\dagger w_i^t)^2 \leq 8m\beta^T \log T$$

Using the Cauchy-Schwarz, we get

$$\sum_{t=1}^T (\mu - \tilde{\nu})^\dagger w_i^t \leq \sqrt{8mT\beta^T \log T}$$

Thus, the regret of PROOF is

$$\begin{aligned}&\mathbb{E}\left[\sum_{t=1}^T \sum_{i=1}^n (\mathbb{E}_{c_i^t|x_i^t}[c_i^t] + \mu)^\dagger (w_i^t - w_{i*}^t)\right] \leq \mathbb{E}\left[\sum_{t=1}^T \sum_{i=1}^n (\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t)^\dagger (w_i^t - w_{i*}^t)\right] + n\sqrt{8mT\beta_T \log T} \\ &= O\left(\sum_{t=1}^T \sum_{i=1}^n \mathbb{E}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2\right] + n\sqrt{8mT\beta_T \log T}\right)\end{aligned}$$

The last step above used Cauchy-Schwartz and the bounded action space assumption. Using Lemma 2, we can conclude the total regret is

$$R_T = O\left(\sum_{t=1}^T \sum_{i=1}^n \sqrt{\frac{dm}{nt}} + n\sqrt{8mT\beta_T \log T}\right) = O\left(n\sqrt{dmT}\right)$$

The last step (bounding $\sum_{t=1}^T t^{-1/2}$) is by an upper bound on the generalized harmonic numbers, which can be found in Theorem 3.2 (b) in the text by Apostol [3]. □

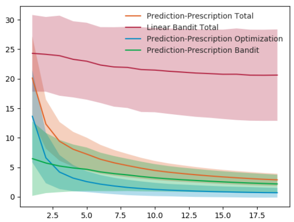
We now consider the more general setting where the intervention could affect the label distribution. We make the assumption that there are finitely many possible actions. In the full paper, we show that an adaptation of PROOF achieves no-regret, and we also consider continuous action space.

Theorem 4. *Assuming an OLS regression ML problem, an adaptation of PROOF has regret* $\tilde{O}\left(n(d|W|)^{1/3} m^{1/2} T^{2/3}\right)$ *with probability* $1 - \delta$.

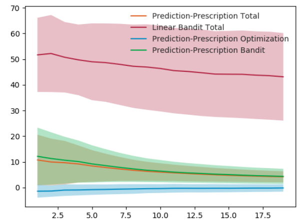
5 Numerical Simulations

In this section, we implement the PROOF algorithm described in Section 4 and show its performance on a simulated dataset. Recall that we train an ML predictor $\hat{f} : X \to C$ where $X \subseteq \mathbb{R}^m$ and $C \subseteq \mathbb{R}^d$. For a data point (x, c), we want to solve the optimization problem $w^* = \min_{w \in W} (c + \mu)^\dagger w$, had we known c and μ, where μ is the unknown bandit parameter. In Section 4, we showed that our PROOF algorithm can achieve no-regret when the true ML predictor is a linear mapping.

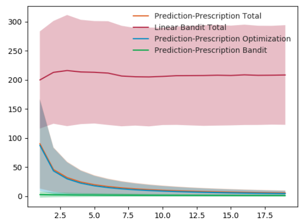
We start our numerical experiments with a small-scale setting. We take feature dimension $m = 20$ and label dimension $d = 5$. At every round we get $n = 20$ data points. Following the tradition in the bandit literature, we assume the bandit reward is bounded in $[-1, 1]$ and assume the feasible region W is the unit l_2-ball, as it's not the absolute magnitude, but the relative magnitude between the bandit and the optimization rewards, that matters. For the true linear map F, i.e. $c = Fx + \epsilon$, we upper



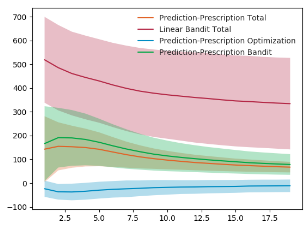
(a) $n = 20, m = 20, d = 5, K_F = 10, \sigma^2 = 0.1$



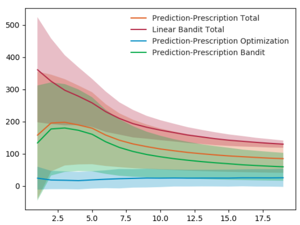
(b) $n = 40, m = 20, d = 5, K_F = 10, \sigma^2 = 0.1$



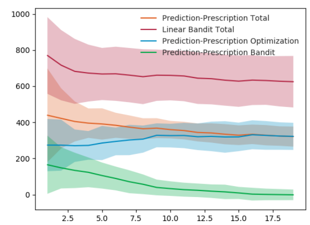
(c) $n = 20, m = 20, d = 5, K_F = 100, \sigma^2 = 0.1$



(d) $n = 500, m = 50, d = 5, K_F = 10, \sigma^2 = 0.1$



(e) $n = 500, m = 50, d = 5, K_F = 1, \sigma^2 = 0.1$



(f) $n = 500, m = 50, d = 5, K_F = 10, \sigma^2 = 0.5$

Figure 2: Numerical simulation results of PROOF compared against vanilla linear bandit. All results are averaged over 10 runs with shaded areas representing the standard deviation.

bound its l_1 matrix norm at 10. We sample the noise $\epsilon \sim \mathcal{N}(0, \sigma^2 I_d)$ from a normal distribution where $\sigma^2 = 0.1$. For the bandit noise, we take $\eta \sim N(0, 10^{-4})$. We use ordinary least squares regression on all the data points collected so far for the learning part at each iteration of bandit data-driven optimization. The optimization at each iteration of PROOF (Line 11 in Algorithm 2) is a non-convex bilinear optimization problem. We use the non-convex solver IPOPT to compute a heuristic solution to this problem. To compute the regret we also need to find the best action given the true reward parameters. This is a convex problem with linear objective and convex quadratic constraints, and thus we use Gurobi to solve it. All experiment results are the average over 10 runs.

In Alg. 2, there is an explicit expression for the confidence ball's radius β^t. That is for establishing the theoretical regret bound. However, in experiments, we found that, following that formula, the radius would be too large ($\sim 10^4$). This makes the algorithm unable to select meaningful action in the early rounds, even when the algorithm's reward estimate $\hat{\mu}$ is already quite accurate. In practice, it is common to select a small value of β, so that the algorithm can quickly concentrate on the correct region of interest. We thus set $\beta^t = 1$.

Under this setting, the problem in theory might be solved simply as a linear bandit problem. The expected cost for a fixed action w is

$$\mathbb{E}_{c,\eta}[(c+\mu)^\dagger w + \eta] = \mathbb{E}[x^\dagger F^\dagger w] + \mu^\dagger w = \mathbb{E}[x^\dagger] F^\dagger w + \mu^\dagger w = \mu^\dagger w,$$

because when we generated x, the distribution has zero mean. This fits in the setting of [9] and thus we could feed the total cost in bandit data-driven optimization to their OFU algorithm. Although vanilla OFU completely ignores the predict-then-optimize procedure, its regret bound is still the same as our PROOF in terms of the order of T. This brings back the question that we have been repeating since the beginning of this paper: if linear (contextual) bandit is a more general framework, why should we care about predict-then-optimize at all?

We have answered this question with the nature of AI4SG projects and real-world applications of ML systems. Here, we can also answer this question using numerical experiments. We show the average regret of PROOF as the orange curve in Figure 2, and that of OFU in red. We can decompose the average regret of PROOF into the regret of the optimization component and the regret of the bandit component. The former is simply the algorithm's optimization cost minus the (overall) best intervention's optimization cost. The latter is defined similarly. Both of them need not be positive, but they sum up to the average regret of PROOF. This decomposition provides a rough illustration of how PROOF makes progress on both ends.

Under the experiment setting introduced above, Figure 2a shows that PROOF can quickly reduce the average regret in both optimization and bandit components. On the other hand, the performance of vanilla OFU is much more underwhelming. A typical bandit regret bound ignores many constant factors. We think this performance discrepancy is primaily due to the large variance in the implicit context x and c, which could be much better captured by training a predictive model. In fact, PROOF also has much smaller variance in its performance than vanilla OFU consistently.

We now tweak the problem parameters a bit and see how the performance changes. If we change the number of data points per iteration from $n = 20$ to $n = 40$, we observe in Figure 2b that the regret of the optimization component becomes very small even at the beginning. This is because have more data to learn from. If we change the upper bound of the norm of the linear mapping matrix F from 10 to 100, we observe in Figure 2c that the optimization regret dominates the total regret. This is also reasonable because the optimization cost now has much larger magnitude than the bandit cost. Also, the vanilla OFU's performance becomes quite disastrous, because now its "bandit" cost, which is the total cost for PROOF, has even larger magnitude and variance.

Admittedly, the setup above serves more as a proof of concept. In the second set of experiments, we scale up the experiments and show that PROOF still has better performance than the vanilla OFU baseline even when the problem parameters are not as friendly. Suppose we receive $n = 500$ data points at every time step, and each data point has $m = 50$ features. Keeping all other parameters the same, we observe in Fig. 2d that PROOF still significantly outperforms OFU. Note that in this case, we seem to have enough data points for the prediction task, and thus the bandit regret dominates the total regret. In Fig. 2e, we change the norm of the linear mapping matrix F from 10 to 1. This implies the optimization cost is now less important than the bandit cost. Indeed, this change reduces the variance for the OFU algorithm and thus it is doing much better than in previous experiments. However, our PROOF algorithm still outperforms OFU. Finally, in Fig. 2f, we increase the noise in the label distribution from $\epsilon \sim \mathcal{N}(0, 0.1I_d)$ to $\epsilon \sim \mathcal{N}(0, 0.5I_d)$. This poses much more challenge to PROOF. As the data become more noisy, as expected, the optimization regret no longer stays close to zero as in the previous two experiments. Nevertheless, PROOF still manages to reduces the total regret at a faster rate than OFU.

Declarations

We believe that the positive social impact of our work has been thoroughly explained throughout this paper. We would like to stress here one potential negative impact that our particular work might bring. Our bandit data-driven optimization framework has an explicit structure to account for unintended social consequences of data-driven policy-making while the policy is being rolled out in the real world. This is our humble acknowledgment as AI researchers of the impossibility to consider all social factors in the initial formulation of an AI4SG problem. However, our work should not be misused as an excuse for not making the best effort to scope the unintended consequence ahead of time.

This work has not been published or first made available before January 1, 2017, and is original work.

References

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320, 2011.

[2] Jacob Abernethy, Alex Chojnacki, Arya Farahi, Eric Schwartz, and Jared Webb. Activeremediation: The search for lead pipes in flint, michigan. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 5–14, 2018.

[3] Tom M Apostol. Introduction to analytic number theory. 1966.

[4] Susan Athey and Stefan Wager. Efficient policy learning. *arXiv preprint arXiv:1702.02896*, 2017.

[5] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[6] Gah-Yi Ban and Cynthia Rudin. The big data newsvendor: Practical insights from machine learning. *Operations Research*, 67(1):90–108, 2019.

[7] Dimitris Bertsimas and Nathan Kallus. From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044, 2020.

[8] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214, 2011.

[9] Varsha Dani, Thomas P Hayes, and Sham M Kakade. Stochastic linear optimization under bandit feedback. In *Conference on Learning Theory*, 2008.

[10] Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601*, 2011.

[11] Adam N Elmachtoub and Paul Grigas. Smart" predict, then optimize". *arXiv preprint arXiv:1710.08005*, 2017.

[12] Fei Fang, Peter Stone, and Milind Tambe. When security games go green: designing defender strategies to prevent poaching and illegal fishing. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 2589–2595, 2015.

[13] Nam Ho-Nguyen and Fatma Kılınç-Karzan. Risk guarantees for end-to-end prediction and optimization processes. 2019.

[14] Yi-hao Kao, Benjamin V Roy, and Xiang Yan. Directed regression. In *Advances in Neural Information Processing Systems*, pages 889–897, 2009.

[15] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[16] Lihong Li, Wei Chu, John Langford, and Robert E 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.

[17] Zheyuan Ryan Shi, Claire Wang, and Fei Fang. Artificial intelligence for social good: A survey. *arXiv preprint arXiv:2001.01818*, 2020.

[18] Zheyuan Ryan Shi, Yiwen Yuan, Kimberly Lo, Leah Lizarondo, and Fei Fang. Improving efficiency of volunteer-based food rescue operations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(8):13369–13375, 2020.

[19] Adith Swaminathan and Thorsten Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *The Journal of Machine Learning Research*, 16(1):1731–1755, 2015.