Bandit Data-Driven Optimization for Crowdsourcing Food Rescue Platforms*

Zheyuan Ryan Shi[1, 2], Zhiwei Steven Wu[1], Rayid Ghani[1], Fei Fang[1]
[1]Carnegie Mellon University
[2]98Connect
{ryanshi, zstevenwu, rayid}@cmu.edu, feif@cs.cmu.edu

Abstract

Food waste and insecurity are two societal challenges that coexist in many parts of the world. A prominent force to combat these issues, food rescue platforms match food donations to organizations that serve underprivileged communities, and then rely on external volunteers to transport the food. Previous work has developed machine learning models for food rescue volunteer engagement. However, having long worked with domain practitioners to deploy AI tools to help with food rescues, we understand that there are four main pain points that keep such a machine learning model from being actually useful in practice: small data, data collected only under the default intervention, unmodeled objectives due to communication gap, and unforeseen consequences of the intervention. In this paper, we introduce bandit data-driven optimization which not only helps address these pain points in food rescue, but also is applicable to other nonprofit domains that share similar challenges. Bandit data-driven optimization combines the advantages of online bandit learning and offline predictive analytics in an integrated framework. We propose PROOF, a novel algorithm for this framework and formally prove that it has no-regret. We show that PROOF performs better than existing baseline on food rescue volunteer recommendation.

1 Introduction

Wasted food account for 25% of the US food consumption, while 12% of the US population struggle with food insecurity (Coleman-Jensen et al. 2020). With the end of COVID-19 pandemic nowhere in sight, the problem is becoming even more serious (Laborde et al. 2020). From New York to Colorado, from San Francisco to Sydney, food rescue platforms are fighting against food waste and insecurity in over 100 cities around the world. Their operation has proved to be effective (Wolfson and Greeno 2018). These platforms match food donations to low-resource communities. Volunteers transport the food from the donor to the recipient and claim upcoming rescues in real time on a smart phone app.

Relying on external volunteers brings great uncertainty to the food rescue operation, and previous works have tried to use machine learning (ML) models to address this uncertainty. For example, Shi, Lizarondo, and Fang (2021) use a recommender system to selectively send push notifications to "most likely" volunteers for an upcoming rescue. Such work shows promising metrics on historical data. However, as we worked with our food rescue partners and discussed with the domain experts, we realized that if one were to deploy such model and replicate the model at other food rescue organizations, a lot more challenges need to be addressed:

1. There may not be enough data to begin with. Food rescue does not have the luxury of millions of training examples, and many food rescues are relatively premature with data initiatives. A small dataset at the beginning leads to inaccurate predictions and hence suboptimal decisions, but this could improve as we collect more data.
2. Too often the initial dataset has some default intervention embedded, while the project aims to find the optimal intervention. For example, Shi et al. (2020) designed a food rescue notification scheme but existing data were all collected under a default suboptimal scheme. If one expects the data distribution to vary across interventions, one has to try out some interventions and collect data under them.
3. We may not perfectly know the correct objective function to optimize for. This is especially true considering the gap between AI researchers and domain practitioners. For sending push notifications to food rescue volunteers, one might obviously want to optimize for the hit rate, but it is less obvious how to formulate each volunteer's reaction to push notifications into the objective.
4. Interventions may have unexpected consequences. Food rescue is a growing domain where neither platforms nor researchers could fully map out the consequences beforehand, thus the inherent impossibility of fully modeling the problem in one shot. However, it is imperative that we proactively account for such limitations.

These challenges require us to handle prediction and optimization in an iterative paradigm. We propose *bandit data-driven optimization* as the first iterative prediction prescription framework that addresses these challenges in a unified and rigorous way. It combines the advantages of 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 various ML models and optimization problems. We analyze its regret convergence, and

*Please refer to https://arxiv.org/abs/2008.11707 for the complete version of this paper.

show that PROOF performs well in the food rescue problem.

1.1 Beyond Food Rescue

The four challenges listed above apply to a lot more non-profit domains beyond food rescue. The canonical problem setting is the scarce resource allocation in the non-profit context. The canonical workflow is that we first train an ML model, then take intervention based on some optimization problem. Using the new data collected under the intervention, we update the ML model and choose a new intervention, so on and so forth, leading to an iterative process.

For example, in game-theoretic anti-poaching, one trains an ML model using geospatial features to predict poacher activity, and then solves an optimization problem to find a patrol strategy (Nguyen et al. 2016; Fang et al. 2016). The patrol finds more poaching data points so we go back to update the ML model, starting another iteration of trial. In education programs, one trains an ML model to predict students' risk of dropping out, and then solves an optimization problem to allocate education resources to the students under budget and fairness constraints (Lakkaraju et al. 2015). After one round, one observes the students' performance and starts the next iteration of the program.

While this iterative workflow is not new, there lacks a rigorous framework to guide this procedure. Since the principles of the various steps are often not aligned, not having such a rigorous framework could lead to operation inefficiency, missed expectations, dampened initiatives, and barriers of mistrust. Bandit data-driven optimization is our first attempt to fill this gap. In this paper, we introduce the model and algorithm in an application-agnostic manner, and then explain how food rescue problems fit into this framework.

2 Related Work

2.1 Food Rescue

Food rescue has received attention from the AI community in recent years. Some well-studied aspects include matching donors and recipients (Aleksandrov et al. 2015; Prendergast 2016; Lundy et al. 2019), planning delivery routes (Nair et al. 2016), and forecasting food demand and supply (Phillips, Hoenigman, and Higbee 2011).

Among the few papers that study the volunteer engagement of food rescue operations, the most relevant to our work is (Shi, Lizarondo, and Fang 2021). They developed a recommender system to select a subset of volunteers to send push notifications to for each rescue trip. Our work builds on top of theirs by considering an iterative process. Manshadi and Rodilitz (2020) study volunteer notification in a similar setting and prove theoretical guarantees. However, we base our work on the former because it scales to real-world food rescue instances whereas the prediction model of the latter is more of a toy example. Furthermore, the former makes no modeling assumptions about volunteer behavior, hence easier to deploy and replicate to other food rescue platforms.

2.2 Bandit Data-driven Optimization

We propose bandit data-driven optimization to address the challenges we encountered in food rescue, because we found surprisingly no existing work that rigorously studies the iterative prediction-prescription procedure. We explain below how several lines of work with similar goals fail to address the challenges we face, and summarize them in Table **??**.

First, (one-shot) data-driven optimization aims to find the action w^* that maximizes the expected value of objective function $p(c, w)$ given some feature x, i.e. $w^* = \arg\max_w \mathbb{E}_{c|x}[p(c, w)]$ (Bertsimas and Kallus 2020; Ban and Rudin 2019). A popular approach is referred to as the predict-then-optimize framework (Elmachtoub and Grigas 2017; Kao, Roy, and Yan 2009). There, one learns an ML predictor f from data and then optimize $p(c, w)$ with the predicted label $c = f(x)$. This entire literature assumes that the optimization objective is known a priori, which is often too good to be true. It also does not consider sequential settings and hence cannot adapt to new data. Meanwhile, inverse optimization (Esfahani et al. 2018; Dong, Chen, and Zeng 2018) also does not apply to our problem, for the actions taken obviously have no definite relationship with the optimal action. Our work touches on optimization under uncertainty (Zheng et al. 2018; Chen et al. 2017; Balkanski, Rubinstein, and Singer 2016). They learn the parameters of an optimization problem. However, they do not learn the data distribution or use the feature/label dataset that is so common in real-world applications like food rescue.

Contextual bandit is a proper setting for sequential decision making (Lai and Robbins 1985), and algorithms like LinUCB (Dani, Hayes, and Kakade 2008; Chu et al. 2011) play a central role in designing PROOF. Recent advances further improve the convergence rate under specific settings (Bastani and Bayati 2020; Mintz et al. 2020). Bandit data-driven optimization reduces to contextual bandit if we skip training an ML model and directly pick an action. However, by doing so, we would effectively give up all the valuable historical data. Furthermore, although bandit algorithms have succeeded in millisecond-level decision-making (Li et al. 2010), they are impractical in applications like food rescue where one time step represents a week, if not a month. The resulting long convergence time would hardly be acceptable to any stakeholders. We prove the same regret bound as previous work in our more realistic setting, with the regret decreasing much faster empirically.

Also related is offline policy learning (Swaminathan and Joachims 2015; Dudík, Langford, and Li 2011; Athey and Wager 2017). It does not need any online trials, and hence is much easier to convince the stakeholder to adopt. However, it assumes the historical data has various actions attempted, which fails to hold in most public sector applications.

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_0, with features x_i^0 and label c_i^0 for data point i, and intervention in-place w_i^0 when the data point is collected. Each feature vector x_i^0 is drawn i.i.d. from an unknown distribution D_x. Each label $c_i^0 \in C$ is independently drawn from an unknown conditional distribution $D(w_i^0)_{c|x_i^0}$, which is parameterized by the intervention w_i^0, as different

Desired properties in food rescue applications	**Bandit data-driven optimization**	Data-driven optimization	Contextual bandit	Offline policy learning
No diverse past data needed	**Yes**	No	Yes	No
Explicit learning and optimization	**Yes**	Yes	No	No
No assumption on policy objective	**Yes**	No	Yes (but ignores domain knowledge)	Yes (but ignores domain knowledge)
Allows for iterative process	**Yes**	No	Yes	Yes
Finds optimal policy quickly	**Yes (compared to bandit)**	Yes (if diverse data available)	No	Yes (if diverse data available)

Table 1: A comparison of different models with respect to the desired properties in food rescue.

interventions could lead to different data distributions. In reality, w_i^0 is often identical across all i. On Line 3, we use all the data collected so far to train an ML model f_t, which is a mapping from features X to labels C. On Line 4, we get a new set of feature samples $\mathbf{x}^t = \{x_i^t\}_{i=1}^n$. Then, we select an intervention $w_i^t \in W$ for each individual i. On Line 5, we commit to interventions $\mathbf{w}^t = \{w_i^t\}$ and receive the labels $\mathbf{c}^t = \{c_i^t\}$. Each label is independently drawn from the distribution $D(w_i^t)_{c|x_i^t}$. Then, on Line 6, we incur a cost u_t.

We assume that the cost u_t is determined by a partially known function $u(\mathbf{c}^t, \mathbf{w}^t)$. The function consists of three terms. The first term $\sum_i p(c_i^t, w_i^t)$ is the known loss. $p(c, w)$ is a fully known function capturing the loss for choosing intervention w and getting label c. It represents our modeling effort and domain knowledge. The second term $\sum_i q(w_i^t)$ is the unknown loss. $q(w)$ is an unknown function representing all the unmodeled objectives and the unintended consequences of using the intervention w. The third term is random noise η. This form of loss – a known part $p(\cdot)$ and an unknown part $q(\cdot)$ – is a realistic compromise of two extremes. We spend a lot of time communicating with food rescue practitioners 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, there will be unmodeled objectives, however hard we try. It would be too arrogant to eliminate $q(\cdot)$ and pretend that anything not going according to the plan is noise. The unknown $q(\cdot)$ is our acknowledgement that any intervention may have unintended consequences.

Hence, the question is how to select the intervention $\mathbf{w}^t$. As is typical in the bandit literature, we define the optimal policy to be that given feature $\mathbf{x}$, pick action $\pi(\mathbf{x})$ such that

$$\pi(\mathbf{x}) = \arg\min_{\mathbf{w}} \mathbb{E}_{\mathbf{c},\eta|\mathbf{x}}[u(\mathbf{c}, \mathbf{w})],$$

where the expectation is taken over labels $\mathbf{c}$ and noise η conditioned on the features $\mathbf{x}$. The goal is to devise an algorithm to select interventions $\mathbf{w}^t$ to minimize the regret

$$R_T = \mathbb{E}_{x,c,\eta}\left[\sum_{t=1}^{T} \left(u(\mathbf{c}^t, \mathbf{w}^t) - u(\mathbf{c}^t, \pi(\mathbf{x}^t))\right)\right].$$

The label c can be a scalar or a vector. For the rest of the paper, we assume $C \in \mathbb{R}^d$ and $W \in \mathbb{R}^d$. W may be discrete or continuous but it is assumed to be bounded.

Procedure 1: BANDIT DATA-DRIVEN OPTIMIZATION

```
1 Receive initial dataset $\mathcal{D} = \{(x_i^0, c_i^0; w_i^0)_{i=1,\ldots,n_0}\}$ 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 $f_t : X \to C$.
4     Given $n$ feature samples $\{x_i^t\} \sim D_x$, choose interventions $\{w_i^t\}$ 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 $u_t = u(\mathbf{c}^t, \mathbf{w}^t) = \sum_i p(c_i^t, w_i^t) + \sum_i q(w_i^t) + \eta$, where $\eta \sim N(0, \sigma^2)$.
```

3.1 Food Rescue Volunteer Recommendation as Bandit Data-driven Optimization

A food rescue organization receives food donations from restaurants and grocery stores and matches them to low-resource community organizations. Once this matching is done, the food rescue dispatcher would post the donor and recipient information on their mobile app. The volunteers will then receive push notifications about the rescue. They could then claim it on the app and then complete the rescue.

Apparently, this is an ideal workflow. Yet occasionally, some rescue trips would stay unclaimed for a long time. Since unclaimed rescues would seriously discourage the donors and recipients from further participation, food rescue dispatchers want to prevent this as much as possible. The dispatcher may recommend each rescue to a subset of volunteers through push notifications, The selection of volunteers to notify is the intervention $w \in \{0, 1\}^d$ (with the j^{th} dimension representing whether to send notification to the j^{th} volunteer). This decision is dependent on how likely a rescue will be claimed by each volunteer. Thus, we develop a ML-based recommender system which uses the features of a rescue and the volunteers, e.g. donor/recipient location, weather, the volunteer's historical activities, etc. (feature x), to predict the probabilities that each volunteer will claim the rescue. After we select w for a rescue, we observe which volunteer actually claim the rescue, that is, the label c (with the j^{th} dimension representing whether the j^{th} volunteer claims the rescue). This data point will be added to our dataset and used for training later. The optimization objective $p(c,w)$ is that we want to send notifications to the volunteers who will claim it, while not sending too many notifications. Obviously, whether or not the rescue gets claimed after these push notifications matter to the food rescue organization. Yet, there is more to the cost to the food rescue, e.g. how each volunteer reacts to push notifications (will they get annoyed and leave?). The $q(\cdot)$ cost could capture such factors.

4 Algorithms and Regret Analysis

We propose a flexible algorithm for bandit data-driven optimization and establish a formal regret analysis.

The data points are drawn from $X \times C \subseteq \mathbb{R}^m \times \mathbb{R}^d$. 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 unknown function f. 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 4.2. The known cost $p(c,w) = c^\dagger w$ is the inner product of label c and 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. In Section 4.3, we will remove this assumption.

4.1 With Exactly Known Objectives

As a primer to our main results to be introduced in the following section, we first look into a special case where we know the optimization objective. That is, our cost only consists of $p(\cdot)$, with $q(\cdot) = 0$. This is not very realistic, but by studying it we will gain intuition for the general case.

At each iteration, this setting resembles the predict-then-optimize framework studied by Elmachtoub and Grigas (2017). Given a sample feature x, we need to solve the linear program with a known feasible region $W \subseteq \mathbb{R}^d$:

$$\begin{aligned} \min_{w} \quad & \mathbb{E}_{c\sim D_{c|x}}[p(c,w)|x] = \mathbb{E}_{c\sim D_{c|x}}[c|x]^\dagger w \\ s.t. \quad & w \in W \end{aligned}$$

We hope to learn a predictor $\hat{f} : X \to C$ from the given dataset, so that we can solve the following problem instead.

$$\begin{aligned} w^*(\hat{c}) := \arg\min_{w} \quad & \hat{c}^\dagger w \qquad \text{where} \quad \hat{c} = \hat{f}(x) \\ s.t. \quad & w \in W \end{aligned}$$

In this paper we assume that the problem has a unique optimal solution. Since the total cost is the same as the known optimization objective, intuitively we should simply commit to the action $w^*(\hat{c})$. By doing so, the expected regret we incur on this data point is $\mathbb{E}_x[r(x)]$, where

$$r(x) = \mathbb{E}_{c|x}[c]^\dagger (w^*(\hat{c}) - w^*(\mathbb{E}_{c|x}[c])).$$

[1]We use superscript † to denote matrix and vector transpose.

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

```
1  Initialize:
2      Find a barycentric spanner $b_1, \dots, b_d$ for $W$
3      Set $A_i^1 = \sum_{j=1}^d b_j b_j^\dagger$ and $\hat{\mu}_i^1 = 0$ for $i = 1, 2, \dots, n$.
4  Receive initial dataset $\mathcal{D} = \{(x_i^0, c_i^0; w_i^0)_{i=1,\dots,n_0}\}$ from
     distribution $D$ on $(X, C)$.
5  for $t = 1, 2, \dots, T$ do
6      Using all data in $\mathcal{D}$, train ML 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 $\beta^t = \max\left(128 d \log t \log \frac{nt^2}{\gamma}, \left(\frac{8}{3}\log\frac{nt^2}{\gamma}\right)^2\right)$
9      for $i = 1, 2, \dots, n$ do
10         Conf. ball $B_i^t = \{\nu : ||\nu - \hat{\mu}_i^t||_{2,A_i^t} \le \sqrt{\beta^t}\}$.
11         Choose intervention
             $w_i^t = \arg\min_{w\in W} \min_{\nu \in B_i^t} (\hat{c}_i^t + \nu)^\dagger w$.
12         Receive 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 $u_i^t = u(x_i^t, c_i^t, w_i^t) = (c_i^t)^\dagger w_i^t + \mu^\dagger w_i^t +$
14         $\eta_i$, where $\eta_i \sim N(0, \sigma^2)$. Let $u_{oi}^t$ be the 1st term
             and let $u_{bi}^t$ be the sum of the 2nd and 3rd term.
15         Update $A_i^{t+1} = A_i^t + w_i^t (w_i^t)^\dagger$
16         Update $\hat{\mu}_i^{t+1} = (A_i^{t+1})^{-1} \sum_{\tau=1}^t u_{bi}^t w_i^t$
```

Theorem 1 establishes that, indeed, this strategy leads to no-regret. This is not entirely trivial, because the optimization is based on the learned predictor yet the cost is based on the true distribution. The proof is instrumental to the subsequent results. All the proofs are in the full version on arXiv.

Theorem 1. *When the total cost is fully modeled, i.e. $q(\cdot) = 0$, simply following the predict-then-optimize optimal solution leads to regret $O(\sqrt{ndmT})$.*

4.2 PROOF: Predict-then-Optimize with Optimism in Face of Uncertainty

When there is no bandit uncertainty, as we showed just now one can simply follow the predict-then-optimize framework and no-regret is guaranteed. However, the unknown bandit cost is crucial to real-world food rescue and similar applications. We now describe the first algorithm for bandit data-driven optimization, PRedict-then-Optimize with Optimism in Face of uncertainty (PROOF), shown in Algorithm 2.

PROOF is an integration of the celebrated Optimism in Face of Uncertainty (OFU) framework 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 $\hat{c}^t$ independent of 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, Hayes, and Kakade (2008), which states that the confidence ball captures the true loss vector μ with high probability. The result was proved for the original OFU algorithm. However, since the result itself does not depend on the way we choose w^t, it still holds in bandit data-driven optimization. We adapt it by adding a union bound so that the result holds for all the n bandits simultaneously.

Lemma 2 (Adapted from Theorem 5 by Dani, Hayes, and Kakade (2008))**.** *Let* $\gamma > 0$*, then* $\mathbb{P}(\forall t, \forall i, \mu \in B_i^t) \geq 1-\gamma$.

The following key lemma decomposes the regret into two components: one involving the online bandit loss, the other concerning the offline supervised learning loss.

Lemma 3. *With probability* $1-\delta$*, Algorithm 2 has regret*

$$O\left(n\sqrt{8mT\beta^T \log T} + \sum_{t=1}^{T}\sum_{i=1}^{n}\mathbb{E}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2\right]\right).$$

Clearly, to characterize the regret, we need to bound $\mathbb{E}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2\right]$. In the case of linear regression, we have the following theorem.

Theorem 4. *Assuming we use ordinary least squares regression as the ML algorithm, Algorithm 2 has regret* $\tilde{O}\left(n\sqrt{dmT}\right)$ *with probability* $1-\delta$.

Theorem 4 assumes a linear regression problem with a specific learning algorithm – ordinary least squares linear regression. If our intent is for Algorithm 2 to be modular where one can use any learning algorithm, we could resort to sample complexity bounds. In the full version of the paper, we include a derivation of the regret bound from the sample complexity perspective. This approach allows us to extend the result in Theorem 4 to a more general setting.

4.3 When Interventions Affect Label Distribution

So far in this section, we have had the assumption that the action w does not affect the distribution D from which as sample (X, C). In many real-world scenarios this is not the case. For example, if the wildlife patrollers change their patrol routes, the poachers' poaching location would change accordingly and hence its distribution would be very different. Thus, it is valuable to study this more general setting where the intervention could affect the label distribution.

First, let us make the assumption that there are finitely many possible actions. We will consider the continuous action space later. Since there are finitely many actions, an intuitive idea is to train an ML predictor for each action separately. Because we do not impose any assumption on our initial dataset, which might only have a single action embedded, we clearly need to use exploration in the bandit algorithm and use the data points gathered along the way to train the predictor. It might seem very natural to fit this directly into the framework of PROOF as shown in Algorithm 2: simply maintain several predictors instead of one, and still choose the best action on Line 11. However, to train the predictor corresponding to each action, we need at least a certain number of data points to bound the prediction error. Yet, PROOF, and UCB-type algorithms in general, do not give a lower bound on how many times each action is tried. For example, Algorithm 2 might never try some action at all, and we would not be able to train a predictor for that action. To resolve this philosophical contradiction, we add a uniform exploration phase of length $\tilde{T}$ at the beginning, where at each round $1, 2, \ldots, \tilde{T}$, each action is taken on some examples. Other than this, we inherit all the setup for the analysis in Section 4.2. We describe the detailed procedure as Algorithm 3 in the full version of the paper.

We establish the following lemma which decomposes the regret into 3 parts: regret during uniform exploration, regret in UCB bandit, and regret through supervised learning.

Lemma 5. *With probability* $1-\delta$*, Algorithm 3 has regret*

$$O\Bigg(n\tilde{T} + n\sqrt{8mT\beta_T \log T} + \sum_{t=\tilde{T}+1}^{T}\sum_{i=1}^{n}\mathbb{E}\left[\left\|\mathbb{E}_{c_i^t|x_i^t, w_i^t}[c_i^t(w_i^t)] - \hat{c}_i^t(w_i^t)\right\|_2\right]\Bigg).$$

By combining Lemma 5 with previous results, we arrive at the regret of PROOF in this more general setting.

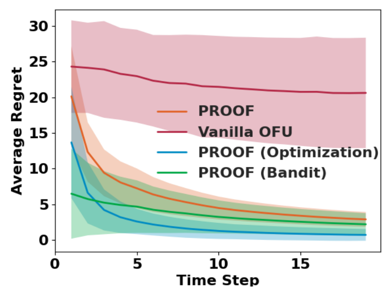
Theorem 6. *With finitely many actions and OLS as the ML algorithm, Algorithm 3 has regret* $\tilde{O}\left(n(d|W|)^{1/3}m^{1/2}T^{2/3}\right)$.

We now move on to the scenario where the action space W is continuous. In this case, we assume the true label of feature x under action w is $c = Fx + Gw + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. A small modification of Algorithm 3 will work in this scenario: instead of rotating over each action in the uniform exploration phase, we simply pick action w uniformly at random for each individual. Then, the regret of the algorithm is as follows.

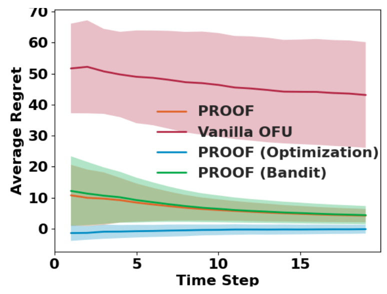
Theorem 7. *Suppose the action space is continuous and the label can be modeled as a linear function of the feature and action. Assuming OLS as the ML algorithm, Algorithm 3 has regret* $\tilde{O}\left(m^{1/3}d^{2/3}nT^{2/3}\right)$.

4.4 PROOF Is a Modular Algorithm

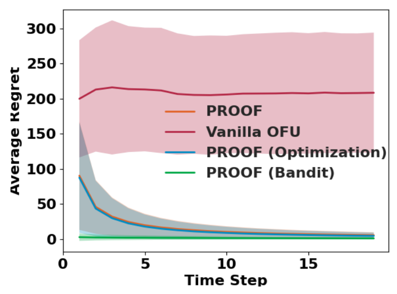
In practice, PROOF can be applied beyond the setting under which we proved the previous results. Rather than a fixed algorithm, it is designed to be modular so that we can plug in different learning algorithms and optimization problems. First, instead of linear regression, PROOF can accommodate any predictive model such as tree-based models and neural networks. Second, The nominal optimization problem need not be a linear optimization problem. The optimization problem may be continuous or discrete, convex or non-convex, as we do not concern ourselves with computational complexity in this paper. In Section 5.2, we demonstrate that even when we insert complex algorithms into the PROOF framework, thereby going beyond the setting where we established formal regret guarantees, PROOF still works well.



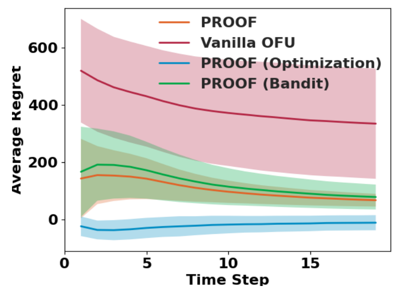
(a) Small scale base case



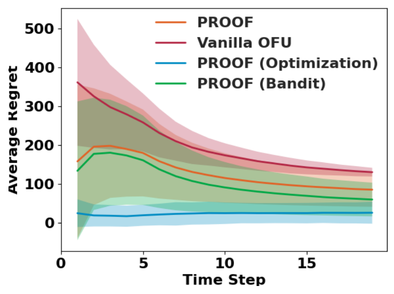
(b) Data per step increased from 20 to 40



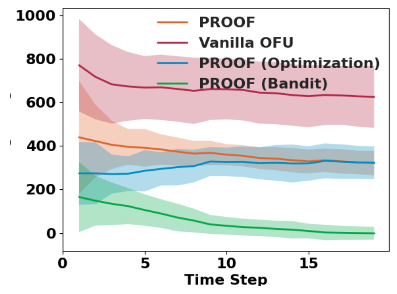
(c) Linear mapping norm multiplied by 10.



(d) Large scale base case.



(e) Linear mapping norm divided by 10.



(f) Data noise multiplied by 5.

Figure 1: Numerical simulation results of PROOF compared against vanilla linear bandit. All results are averaged over 10 runs with shaded areas representing the standard deviation.

5 Experimental Results

5.1 Numerical Simulations

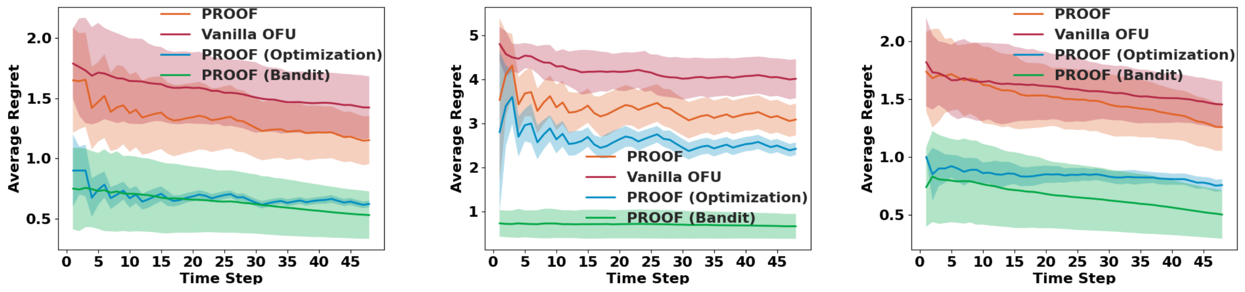
As the first step of validation, we implement PROOF in the setting described in Section 4.2 on a simulated dataset. We start with a small-scale experiment. Recall that we train an ML predictor $\hat{f} : X \to C$ where $X \subseteq \mathbb{R}^m$ and $C \subseteq \mathbb{R}^d$. We take feature dimension $m = 20$ and label dimension $d = 5$. At every round we get $n = 20$ data points. As is typical, we assume the bandit reward is bounded in $[-1, 1]$ and the feasible region W is the unit l_2-ball. For the true linear map F where $c = Fx + \epsilon$, we upper 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 $\sigma^2 = 0.1$. We take the bandit noise $\eta \sim N(0, 10^{-4})$. We use OLS at each time step. We solve the non-convex program on Line 11 in Algorithm 2 with IPOPT. We find the best action given the true reward parameters using Gurobi. We set $\beta^t = 1$ so that the algorithm can quickly concentrate on the region of interest.

The expected cost for a fixed action w is $\mathbb{E}_{c,\eta}[(c+\mu)^\dagger w + \eta] = \mathbb{E}[x^\dagger F^\dagger w] + \mu^\dagger w = \mu^\dagger w$, because when we generated x, the distribution has zero mean. This problem in theory might be solved as a linear bandit by feeding the total cost to OFU. Since the regret bound of OFU is the same as PROOF in the order of T, this brings back the point that we have been emphasizing since the beginning: if linear bandit is a more general model whose algorithms already solve our problem, why would we care about bandit data-driven optimization?

In Sections 1 through 3, we answered this question with the characteristics of the food rescue. Here, we answer this question using experiments. We show the average regret of PROOF as the orange curve in Figure 1, 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 (known) cost minus the best intervention's optimization cost. The latter is defined similarly. Neither needs to be positive. Figure 1a shows that PROOF quickly reduces the regret in both components, while the performance of vanilla OFU is much more underwhelming. This difference is because an offline predictive model captures the large variance in the implicit context x and c much better. In fact, PROOF consistently has much smaller variance than OFU.

We now tweak the parameters a bit. When we increase the number of data points per iteration from $n = 20$ to 40, Figure 1b shows that the regret of the optimization component becomes very small to start with, because we have more data to learn from. When we increase $||F||$ from 10 to 100, Figure 1c shows that the optimization regret dominates the total regret, as the optimization cost is now much larger than the bandit cost. Here, vanilla OFU suffers even more, because now its cost signal has even larger magnitude and variance.

We then scale up the experiments and show that PROOF still outperforms OFU even when the problem parameters are not as friendly. Suppose we get $n = 500$ data points every time and each data point has $m = 50$ features. Keeping all other parameters unchanged, Fig. 1d shows that PROOF still outperforms OFU by a lot. In Fig. 1e, we change $||F||$ from 10 to 1, making the optimization cost less important. This reduces the variance of OFU and it is doing better than previously. However, our PROOF still outperforms OFU. In



(a) Base case (b) Known cost $p(\cdot)$ multiplied by 4 (c) Initial dataset size decreased to 20

Figure 2: The experiment results on the real-world food rescue data of PROOF compared against vanilla linear bandit. All results are averaged over 10 runs with shaded areas representing the standard deviation.

Fig. 1f, we increase the label noise from $\epsilon \sim \mathcal{N}(0, 0.1I_d)$ to $\mathcal{N}(0, 0.5I_d)$. This poses more challenge to PROOF. But still, PROOF manages to keep its regret below OFU.

5.2 Food Rescue Volunteer Recommendation

Bandit data-driven optimization is motivated by the practical challenges in food rescue volunteer recommendation. After abstracting these challenges to a theoretical model, we now return to the food rescue problem. We have introduced the details of food rescue operations at the end of Section 3.

There are 100 volunteers. At each time step, we get a new rescue and decide a subset of 10 volunteers to whom we send push notifications. We represent this action with a binary vector $w \in \{0,1\}^{100}$ such that $w_i = 1$ if volunteer i is notified and 0 otherwise. Thus, the feasible action space W is $\{0,1\}^{100}$ with the constraint of $\sum_{i=1}^{100} w_i \leq 10$. The action w we take at each time step is backed by a content-based ML recommender system. The ML model receives a feature vector x which describes a particular rescue-volunteer pair, and outputs a label prediction $\hat{c}$ as the likelihood of this volunteer claiming this rescue.[2] We adapt this ML component from the one studied in (Shi, Lizarondo, and Fang 2021). Its feature selection, model architecture, and training techniques, are not trivial at all. Yet, since they are not the focus of this paper, we include all these details in Appendix D in the full version. The actual label c is a one-hot vector in $\{0,1\}^{100}$ indicating which volunteer actually claimed the rescue. The known cost $p(c,w) = c^\dagger w$ is 1 if we notify a volunteer who eventually claimed a rescue and 0 otherwise. To minimize it, we could negate the label c (and its prediction $\hat{c}$). The bandit cost $q(\cdot)$ is the same as before. We solve the optimization at each time step of PROOF with Gurobi after applying a standard linearization trick (Liberti, Cafieri, and Tarissan 2009). We also gradually decrease the confidence radius β.

Unlike the case in Section 5.1, OFU algorithm does not work here in principle. This is because, working with real-world data, we do not know the data distribution and it is almost certainly not zero-mean. In fact, this experiment has also gone beyond the setting for which we proved formal regret bound for PROOF, yet we would like to see how these two algorithms perform in such a real-world use case.

We assume an initial dataset of 300 rescues and run the algorithms for 50 time steps, each time step corresponding to one new rescue. As shown in Figure 2a, PROOF outperforms vanilla OFU by roughly 15%. The performance gain by PROOF can be contributed to its effective use of the available data, as the progress on bandit made by PROOF and vanilla OFU are quite similar. In Figure 2b, we scale up the known part of the cost by a factor of 4. Because the optimization is more emphasized, it is unsurprising to see that most of PROOF's progress depends on the recommender system itself. In this case, it has a larger performance margin over vanilla OFU. Finally, in Figure 2c we decrease the size of the initial dataset from 300 rescues to 20 rescues. We observe that PROOF still has an edge over vanilla OFU. The margin is minimal at the initial time steps, because we have much less initial information here. Yet still, as time goes by PROOF picks up more information in the feature/label dataset to expand its margin. In actual food rescue projects, the amount of initial data is typically more than this, more resembling Figure 2a, but Figure 2c assures us that PROOF still works in this more extreme case.

6 Conclusion

Volunteer engagement in food rescue is a challenging and impactful problem. Motivated by four practical pain points in food rescue, we proposed bandit data-driven optimization, designed the PROOF algorithm, and showed that it has no-regret. Finally, we show its better performance over bandit algorithm in simulations and the food rescue context. Our technological intervention only amplifies the existing initiative rather than create a new one, thus more likely to achieve deployment and sustainable impact (Toyama 2015).

This work has many potential applications beyond food rescue. We view it as our first attempt to bridge the last-mile gap between static ML models and their actual deployment in the real-world public-sector context.

[2]Here the label is 1-dimensional while our action space is 100-dimensional. This is easy to resolve. Each rescue-volunteer pair has m' features. While in practice we have $\hat{f}' : \mathbb{R}^{m'} \to \mathbb{R}$ and pass 100 feature vectors to it serially, one could think of a product model $\hat{f} = \prod_{i=1}^{100} \hat{f}'$ which takes the concatenation of 100 feature vectors and outputs a 100-dimensional vector.

Acknowledgments

We have learned so much from our collaborators at 412 Food Rescue and other nonprofit organizations and we thank them for what they are doing for our community. We thank Thomas G. Dietterich, Fatma Kılınç-Karzan and Kit T. Rodolfa for the inspiring discussions at the early stage of this work. This work was supported in part by NSF grants IIS-1850477 and IIS-2046640 (CAREER), a Siebel Scholarship and a Carnegie Mellon Presidential Fellowship. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies.

References

Aleksandrov, M.; Aziz, H.; Gaspers, S.; and Walsh, T. 2015. Online fair division: analysing a food bank problem. In *Proceedings of the 24th International Conference on Artificial Intelligence*, 2540–2546.

Athey, S.; and Wager, S. 2017. Efficient policy learning. *arXiv:1702.02896.*

Balkanski, E.; Rubinstein, A.; and Singer, Y. 2016. The Power of Optimization from Samples. In *NIPS*, 4017–4025.

Ban, G.-Y.; and Rudin, C. 2019. The big data newsvendor: Practical insights from machine learning. *Operations Research*, 67(1): 90–108.

Bastani, H.; and Bayati, M. 2020. Online decision making with high-dimensional covariates. *Operations Research.*

Bertsimas, D.; and Kallus, N. 2020. From predictive to prescriptive analytics. *Management Science*, 66(3): 1025–1044.

Chen, L.; Gupta, A.; Li, J.; Qiao, M.; and Wang, R. 2017. Nearly optimal sampling algorithms for combinatorial pure exploration. In *Conference on Learning Theory*, 482–534. PMLR.

Chu, W.; Li, L.; Reyzin, L.; and Schapire, R. 2011. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 208–214.

Coleman-Jensen, A.; Rabbitt, M. P.; Gregory, C. A.; and Singh, A. 2020. Household Food Security in the United States in 2019. *USDA-ERS Economic Research Report.*

Dani, V.; Hayes, T. P.; and Kakade, S. M. 2008. Stochastic linear optimization under bandit feedback. In *Conference on Learning Theory.*

Dong, C.; Chen, Y.; and Zeng, B. 2018. Generalized inverse optimization through online learning. In *NeurIPS.*

Dudík, M.; Langford, J.; and Li, L. 2011. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601.*

Elmachtoub, A. N.; and Grigas, P. 2017. Smart" predict, then optimize". *arXiv preprint arXiv:1710.08005.*

Esfahani, P. M.; Shafieezadeh-Abadeh, S.; Hanasusanto, G. A.; and Kuhn, D. 2018. Data-driven inverse optimization with imperfect information. *Mathematical Programming*, 167(1): 191–234.

Fang, F.; Nguyen, T. H.; Pickles, R.; Lam, W. Y.; Clements, G. R.; An, B.; Singh, A.; Tambe, M.; and Lemieux, A. 2016. Deploying PAWS: Field optimization of the protection assistant for wildlife security. In *Twenty-eighth IAAI conference.*

Kao, Y.-h.; Roy, B. V.; and Yan, X. 2009. Directed regression. In *Advances in Neural Information Processing Systems*, 889–897.

Laborde, D.; Martin, W.; Swinnen, J.; and Vos, R. 2020. COVID-19 risks to global food security. *Science*, 369(6503): 500–502.

Lai, T. L.; and Robbins, H. 1985. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1): 4–22.

Lakkaraju, H.; Aguiar, E.; Shan, C.; Miller, D.; Bhanpuri, N.; Ghani, R.; and Addison, K. L. 2015. A machine learning framework to identify students at risk of adverse academic outcomes. In *KDD*, 1909–1918.

Li, L.; Chu, W.; Langford, J.; and Schapire, R. E. 2010. A contextual-bandit approach to personalized news article recommendation. In *WWW*, 661–670.

Liberti, L.; Cafieri, S.; and Tarissan, F. 2009. Reformulations in mathematical programming: A computational approach. In *Foundations of Computational Intelligence Volume 3*, 153–234. Springer.

Lundy, T.; Wei, A.; Fu, H.; Kominers, S. D.; and Leyton-Brown, K. 2019. Allocation for social good: auditing mechanisms for utility maximization. In *ACM EC.*

Manshadi, V.; and Rodilitz, S. 2020. Online Policies for Efficient Volunteer Crowdsourcing. *arXiv preprint arXiv:2002.08474.*

Mintz, Y.; Aswani, A.; Kaminsky, P.; Flowers, E.; and Fukuoka, Y. 2020. Nonstationary bandits with habituation and recovery dynamics. *Operations Research*, 68(5): 1493–1516.

Nair, D. J.; Grzybowska, H.; Rey, D.; and Dixit, V. 2016. Food rescue and delivery: Heuristic algorithm for periodic unpaired pickup and delivery vehicle routing problem. *Transportation Research Record*, 2548(1): 81–89.

Nguyen, T. H.; Sinha, A.; Gholami, S.; Plumptre, A.; Joppa, L.; Tambe, M.; Driciru, M.; Wanyama, F.; Rwetsiba, A.; Critchlow, R.; et al. 2016. CAPTURE: A New Predictive Anti-Poaching Tool for Wildlife Protection. In *AAMAS*, 767–775.

Phillips, C.; Hoenigman, R.; and Higbee, B. 2011. Food redistribution as optimization. *arXiv preprint arXiv:1108.5768.*

Prendergast, C. 2016. The Allocation of Food to Food Banks. *EAI Endorsed Trans. Serious Games*, 3(10): e4.

Shi, Z. R.; Lizarondo, L.; and Fang, F. 2021. A Recommender System for Crowdsourcing Food Rescue Platforms. In *Proceedings of the Web Conference 2021*, 857–865.

Shi, Z. R.; Yuan, Y.; Lo, K.; Lizarondo, L.; and Fang, F. 2020. Improving Efficiency of Volunteer-Based Food Rescue Operations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(8): 13369–13375.

Swaminathan, A.; and Joachims, T. 2015. Batch learning from logged bandit feedback through counterfactual risk minimization. *JMLR*, 16(1): 1731–1755.

Toyama, K. 2015. *Geek heresy: Rescuing social change from the cult of technology*. PublicAffairs.

Wolfson, M. D.; and Greeno, C. 2018. Savoring surplus: effects of food rescue on recipients. *Journal of Hunger & Environmental Nutrition*.

Zheng, S.; Waggoner, B.; Liu, Y.; and Chen, Y. 2018. Active Information Acquisition for Linear Optimization. In *Uncertainty in artificial intelligence*.