# Boosted Off-Policy Learning

**Ben London**
Amazon

**Levi Lu**
Amazon

**Ted Sandler**
Groundlight.ai

**Thorsten Joachims**
Amazon

## Abstract

We propose the first boosting algorithm for off-policy learning from logged bandit feedback. Unlike existing boosting methods for supervised learning, our algorithm directly optimizes an estimate of the policy's expected reward. We analyze this algorithm and prove that the excess empirical risk decreases (possibly exponentially fast) with each round of boosting, provided a "weak" learning condition is satisfied by the base learner. We further show how to reduce the base learner to supervised learning, which opens up a broad range of readily available base learners with practical benefits, such as decision trees. Experiments indicate that our algorithm inherits many desirable properties of tree-based boosting algorithms (e.g., robustness to feature scaling and hyperparameter tuning), and that it can outperform off-policy learning with deep neural networks as well as methods that simply regress on the observed rewards.

## 1 INTRODUCTION

Boosting algorithms (Schapire & Freund, 2012) have been a go-to approach for supervised learning problems, where they have been highly successful across a wide range of applications. Not only have they been the winning method for many Kaggle competitions (Chen & Guestrin, 2016), a recent empirical study (Yang et al., 2020) shows that gradient-boosted trees (Friedman, 1999; Mason et al., 1999) are the most successful model across the OpenML (Vanschoren et al., 2013; Feurer et al., 2019) tasks, providing the best classification accuracy on 38.60% of the datasets. In contrast, their closest competitor, multi-layer perceptrons, achieve top performance on only 20.93% of the datasets. Beyond pure predictive accuracy, boosted ensembles are also considered robust to feature scaling and hyperparameter tuning, and generally easier to train than neural networks, which often makes them a more practical choice for real-world applications (Grinsztajn et al., 2022).

While boosting has enjoyed much success in supervised learning, a growing number of applications—such as search, recommendation and display advertising—fall outside the realm of supervised learning. Arguably, these applications are better formulated as contextual bandit problems, since the feedback (i.e., supervision) one observes depends on the actions that are taken. In practice, training often happens offline, using logged bandit feedback. For example, search ranking algorithms are usually trained on logged click data rather than in real time on live web traffic. This offline setting necessitates an *off-policy* approach to learning, since the data used for training is collected by a different policy than the one being trained. To date, many off-policy learning algorithms have been proposed (Strehl et al., 2010; Dudik et al., 2011; Bottou et al., 2013; Swaminathan & Joachims, 2015a,b; Joachims et al., 2018; Wu & Wang, 2018; Ma et al., 2019; Kallus, 2019; London & Sandler, 2019; Chen et al., 2019; Faury et al., 2020; Jeunen et al., 2020), yet none have explored a boosting approach.

In this paper, we derive the first boosting algorithm designed specifically for off-policy learning of contextual bandit policies. The algorithm, which we call *BOPL* (for *boosted off-policy learning*), sequentially constructs an *ensemble policy*—comprised of a linear combination of predictors (e.g., decision trees)—by directly optimizing an estimate of the policy's expected reward—which is, after all, the quantity of interest when the policy is deployed. This *policy optimization* approach stands in contrast to methods that indirectly derive a policy by regressing on the observed rewards, since better reward prediction (in terms of squared error) does not necessarily result in a better policy (Beygelzimer & Langford, 2009).

The BOPL algorithm is built upon a rigorous theoretical foundation. We first prove that its learning objective is *smooth* (i.e., has a Lipschitz gradient), and then use this property to derive a gradient boosting algorithm. Our smoothness analysis has the advantage of giving us a closed-form expression for the ensemble weight of any predictor, and an upper bound on the learning objective at each round of boosting. To address optimization issues, we de-

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. Copyright 2023 by the author(s).

rive a variant of BOPL, called *BOPL-S*, that optimizes a surrogate objective, which upper-bounds BOPL's objective and can be convex. Unlike some existing off-policy methods (Dudik et al., 2011; Bottou et al., 2013; Swaminathan & Joachims, 2015a,b; Joachims et al., 2018; London & Sandler, 2019), which require models to be differentiable in a fixed and enumerable set of parameters, BOPL only requires a *base learner* that, at each round, trains a predictor to approximate the gradient with respect to the current ensemble. We show how this base learning objective can be reduced to a weighted regression or binary classification problem, which can be solved by off-the-shelf supervised learning algorithms. Moreover, we prove an upper bound on the excess empirical risk (that is, the empirical suboptimality) that decreases with each round of boosting as long as the base learner can produce a nontrivial predictor (akin to a "weak" learning condition). Under certain conditions, the convergence rate can be exponentially decreasing in the number of rounds. When coupled with concentration and uniform convergence bounds, our excess empirical risk bound yields a bound on the excess population risk.

To evaluate the effectiveness of our approach, we conduct experiments on four public datasets. We find that BOPL outperforms boosted reward regression on three of these, thus illustrating that it can be advantageous to perform policy optimization, since it directly optimizes an estimate of the quantity we care about, the expected reward. We also find that boosted ensemble policies are competitive with neural network-based policies—while requiring less tuning, shorter training time and fewer computing resources—thereby demonstrating that boosted off-policy learning is a compelling alternative to deep off-policy learning.

## 2 RELATED WORK

Of the extensive literature on learning from logged bandit feedback (Strehl et al., 2010; Dudik et al., 2011; Bottou et al., 2013; Swaminathan & Joachims, 2015a,b; Wu & Wang, 2018; Ma et al., 2019; Kallus, 2019; London & Sandler, 2019; Chen et al., 2019; Faury et al., 2020; Jeunen et al., 2020), one particularly influential prior work, BanditNet (Joachims et al., 2018), can be viewed as the deep learning analog of our boosting approach. The authors argue that complex policy classes can overfit the logged propensities. To combat this phenomenon, they propose optimizing a self-normalized estimator, which they show is equivalent to optimizing an unnormalized estimator with translated rewards. For this reason, we explicitly allow rewards to be negative so as to accommodate negative translations—which we find to be critically important in our experiments.

The connection between early work on boosting (Kearns & Valiant, 1989; Schapire, 1990; Freund, 1995; Freund & Schapire, 1997) and functional gradient descent was formalized by Friedman (1999) and Mason et al. (1999). This spawned a number of methods known collectively as *gradient boosting*. One notable descendent of this line of work is XGBoost (Chen & Guestrin, 2016), which has become a popular choice for practitioners due to its speed and efficacy with minimal parameter tuning. XGBoost derives from a second-order Taylor approximation of an arbitrary loss function, and is optimized for regression tree base learners. While our boosting objective could in theory be approximated by XGBoost, our smoothness-based derivation gives us a closed-form expression for the ensemble weight of any base learner (not just regression trees) and an empirical risk bound, thus motivating a custom algorithm.

Learning from logged bandit feedback can be cast as a form of cost-sensitive classification, wherein labels have different costs depending on the context. Boosting algorithms for cost-sensitive classification have been proposed (Fan et al., 1999; Abe et al., 2004; Appel et al., 2016). However, the primary difference between cost-sensitive learning and learning from bandit feedback is that, in the former, it is assumed that all costs are known *a priori*, whereas in the latter, only the cost of the selected action is observed.

The prior works that are most related to our own involve boosting with bandit feedback. Boosting has been applied to online multiclass classification with bandit feedback (Chen et al., 2014; Zhang et al., 2019), online convex optimization with bandit feedback (Brukhim & Hazan, 2021) and online reinforcement learning (Abel et al., 2016; Brukhim et al., 2021). The primary difference between these works and our own is that they consider an online setting in which the learning algorithm can interact directly with the environment and observe the corresponding outcome. In contrast, we consider an offline setting in which the learner cannot interact with the environment, and must therefore rely on logged interactions to estimate how the learned policy will perform online. The latter learning problem is inherently counterfactual, which is why off-policy corrections (e.g., importance weighting) are needed.

## 3 PRELIMINARIES

Let $\mathcal{X}$ denote a set of *contexts*, and let $\mathcal{A}$ denote a set of *actions* (sometimes referred to as *arms*).[1] We are interested in learning a *policy*, $\pi$, which defines a (stochastic) mapping from contexts to actions. Given a context, $x \in \mathcal{X}$, we use $\pi(x)$ to denote the policy's conditional probability distribution on $\mathcal{A}$, and $\pi(a \mid x)$ to denote the conditional probability of a given $a \in \mathcal{A}$. If $\pi$ is deterministic, its distribution is a delta function.

A policy interacts with the environment in the following process. At each interaction, the environment generates

[1] Though it is often the case that $\mathcal{A}$ depends on the context, to simplify notation, we assume that $\mathcal{A}$ is static.

a context, $x \in \mathcal{X}$, according to a stationary distribution, $\mathbb{D}_x$. The policy responds by selecting (sampling) an action, $a \sim \pi(x)$, and consequently receives a stochastic *reward*, $r(x, a) \in \mathbb{R}$, which measures how good the selected action is for the given context. Importantly, we only observe reward for actions the policy selects. This partial supervision is referred to as *bandit feedback*. We view the reward as a random function, $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, drawn from a stationary distribution, $\mathbb{D}_r$. Note that we do not assume that the rewards are nonnegative, to allow for the possibility that they could be *baselined* (i.e., translated). This has been shown to be a crucial tool in preventing the so-called *propensity overfitting* problem with complex model classes (Joachims et al., 2018).

We want to find a policy that maximizes expected reward. In the sequel, it will be more convenient to think in terms of minimization, so we instead seek a policy with minimum expected *negative* reward, which we call *risk*:

$$L(\pi) \triangleq \mathop{\mathbb{E}}_{x\sim\mathbb{D}_x} \mathop{\mathbb{E}}_{a\sim\pi(x)} \mathop{\mathbb{E}}_{r\sim\mathbb{D}_r} [-r(x, a)]. \tag{1}$$

Note that an optimal policy is one that always selects an action with maximum mean reward, $\arg\max_{a\in\mathcal{A}} \mathbb{E}_{r\sim\mathbb{D}_r}[r(x, a)]$. However, such a policy may not be in the class of policies under consideration; and even if one is, it may be impossible to find given finite training data. We discuss this further in Section 3.2, which motivates our off-policy learning objective.

### 3.1 Softmax Ensemble Policies

Let $\mathcal{F} \subseteq \{\mathcal{X} \times \mathcal{A} \to \mathbb{R}\}$ denote a class of *predictors*, each of which maps context-action pairs to real-valued scores. For instance, $\mathcal{F}$ could be a class of decision trees. It will at times be more convenient to think of a predictor, $f \in \mathcal{F}$, as a vector-valued function that, given $x$, outputs the scores for all $a \in \mathcal{A}$ simultaneously, denoted by $f(x) \in \mathbb{R}^{|\mathcal{A}|}$.

For a collection of $T$ predictors, $f_1, \dots, f_T \in \mathcal{F}$, with associated weights, $\alpha_1, \dots, \alpha_T \in \mathbb{R}$, let

$$F_T(x, a) \triangleq \sum_{t=1}^{T} \alpha_t f_t(x, a)$$

denote the *ensemble prediction*. We use $\mathcal{F}_T \triangleq \big\{(x, a) \mapsto F_T(x, a) : \forall t,\, f_t \in \mathcal{F},\, \alpha_t \in \mathbb{R}\big\}$ to denote the class of size-$T$ ensembles, and the shorthand $F_T \in \mathcal{F}_T$ to denote a member of this class.

Since we are not interested in predictions, but rather in a policy that selects actions, we use the following *softmax* transformation. It can be applied to individual predictors or ensemble predictors. In particular, for a given ensemble, $F_T \in \mathcal{F}_T$, and $\beta \geq 0$, the corresponding *softmax ensemble policy* is

$$\pi_T(a \,|\, x) \triangleq \pi(a \,|\, x; F_T, \beta) \triangleq \frac{\exp\left(\beta F_T(x, a)\right)}{\sum_{a'\in\mathcal{A}} \exp\left(\beta F_T(x, a')\right)}.$$

We use $\Pi_T \triangleq \big\{(x, a) \mapsto \pi(a \,|\, x; F_T, \beta) : F_T \in \mathcal{F}_T\big\}$ to denote the class of softmax ensemble policies (for a given $\beta \geq 0$), and the shorthand $\pi_T \in \Pi_T$ to denote a member of this class.

The hyperparameter $\beta$ (known as the *inverse temperature*) is only used for notational convenience, as $\beta \to \infty$ transforms the softmax into an *argmax*. Importantly, for any $\beta < \infty$, the softmax is differentiable, Lipschitz and (as we will later show) smooth, which are useful properties for optimization. For any softmax ensemble with finite $\beta \neq 1$, an equivalent policy with $\beta = 1$ can be obtained by rescaling the ensemble weights. Thus, during learning, we set $\beta = 1$.

### 3.2 Off-Policy Learning

Assume that we have collected data using an existing policy (not necessarily a softmax), $\pi_{\mathrm{L}}$, which we call the *logging policy*. Our only requirement for $\pi_{\mathrm{L}}$ is that it has *full support*; meaning, $\pi_{\mathrm{L}}(a \,|\, x) > 0$ for all $x \in \mathcal{X}$ and $a \in \mathcal{A}$. For $i = 1, \dots, n$ interactions, we log: the context, $x_i \sim \mathbb{D}_x$; the selected action, $a_i \sim \pi_{\mathrm{L}}(x_i)$; the *propensity* of the selected action, $p_i \triangleq \pi_{\mathrm{L}}(a_i \,|\, x_i)$; and the observed reward, $r_i \triangleq r(x_i, a_i)$, where $r \sim \mathbb{D}_r$. The resulting dataset, $S \triangleq (x_i, a_i, p_i, r_i)_{i=1}^n$, can be used to train and evaluate new policies.

Recall that our goal is to minimize the risk (Equation 1), and that the policy that achieves this is easy to derive if we have a perfect model of the mean rewards. In particular, if our hypothesis space is the class of softmax ensemble policies, $\Pi_T$, then an optimal policy is given by $F_T(x, a) = \mathbb{E}_{r\sim\mathbb{D}_r}[r(x, a)]$ and $\beta \to \infty$. This motivates a strategy wherein we train an ensemble to predict the mean reward—an approach sometimes called *reward regression*, or *Q-learning* in the reinforcement learning literature. If we assume that the observed rewards are Gaussian perturbations of the mean reward, then a (nonlinear) least-squares regression makes sense:

$$\min_{F_T\in\mathcal{F}_T} \frac{1}{n} \sum_{i=1}^{n} (F_T(x_i, a_i) - r_i)^2. \tag{2}$$

If $\mathcal{F}_T$ contains the mean reward function, then this optimization will find an optimal ensemble (hence, optimal policy) in the limit of $n \to \infty$.

Unfortunately, in realistic scenarios, neither of these assumptions hold, and the relationship between regression error and expected reward can quickly become vacuous (Beygelzimer & Langford, 2009). Though $\mathcal{F}_T$ may be a very expressive function class, it is still unlikely that it contains the mean reward function. Further, while logged data may be abundant in industrial settings (such as search or recommendation engines), it may never be large enough; or it may be impractical to train on an extremely large dataset.

The key drawback of reward regression is that it does not directly optimize the quantity we actually care about: the risk (or, expected reward) of the learned policy when deployed. To see why, note that a reward predictor can improve its fit of the logged data and yet still not produce a policy with lower risk (i.e., higher expected reward). This can happen if the fit improves on actions that the new policy is unlikely to select. More perverse is the case where a trade-off is made that improves the fit on actions the new policy is unlikely to select while deteriorating the fit on actions *it is* likely to select. In this case, reducing the squared error on the logged data can result in a worse policy.

We therefore consider a different approach based on *policy optimization*. Unlike reward regression, policy optimization fits a policy to directly minimize risk, $\min_{\pi \in \Pi} L(\pi)$. The true risk is unobservable, but given data, $S$, we can optimize an empirical estimate, $\hat{L}(\pi, S)$. This yields an *empirical risk minimization* (ERM) strategy, $\min_{\pi \in \Pi} \hat{L}(\pi, S)$.

The choice of risk estimator is crucial; if the estimator is biased, we may end up minimizing the wrong objective. Fortunately, we can obtain an unbiased estimate using the *inverse propensity scoring* (IPS) estimator,

$$\hat{L}(\pi, S) \triangleq \frac{1}{n} \sum_{i=1}^{n} -r_i \, \frac{\pi(a_i \,|\, x_i)}{p_i}. \tag{3}$$

When $\pi_{\mathrm{L}}$ has full support, it is straightforward to verify that $\mathbb{E}_S[\hat{L}(\pi, S)] = L(\pi)$. However, if the propensities can be very small, then the IPS estimator has large variance. For this reason, it is common practice to either design a logging policy whose propensities are lower-bounded, or to truncate the *importance weights*, $\pi(a_i \,|\, x_i)/p_i$ (Ionides, 2008). Alternatively, we could use an estimator that better balances the bias-variance trade-off, such as *self-normalizing IPS* (Swaminathan & Joachims, 2015b) or *doubly robust* (Dudik et al., 2011). For simplicity, we will proceed with the regular IPS estimator, noting that more sophisticated estimators are complementary to our proposed approach.

In the following section, it will be convenient to view the empirical risk as the average *loss* of an ensemble. For $i = 1, \ldots, n$, let $\ell_i(F_T) \triangleq -\frac{r_i}{p_i} \pi(a_i \,|\, x_i; F_T, \beta)$ denote the loss of $F_T$ with respect to the $i^{\text{th}}$ example. Using this notation, our learning objective is

$$\min_{\pi_T \in \Pi_T} \hat{L}(\pi_T, S) = \min_{F_T \in \mathcal{F}_T} \frac{1}{n} \sum_{i=1}^{n} \ell_i(F_T). \tag{4}$$

## 4 BOOSTED OFF-POLICY LEARNING

We will approach the optimization problem in Equation 4 using a greedy strategy known as *boosting*, which can be viewed as coordinate descent or functional gradient descent (Schapire & Freund, 2012). Boosting proceeds in *rounds*, wherein at each round, $t = 1, \ldots, T$, the goal is to select a new predictor, $f_t \in \mathcal{F}$, and corresponding weight, $\alpha_t \in \mathbb{R}$, to add to our current ensemble, $F_{t-1}$, such that the empirical risk is minimized. Formally, the optimization problem at round $t$ is

$$\min_{\substack{f_t \in \mathcal{F}, \\ \alpha_t \in \mathbb{R}}} \hat{L}(\pi_t, S) = \min_{\substack{f_t \in \mathcal{F}, \\ \alpha_t \in \mathbb{R}}} \frac{1}{n} \sum_{i=1}^{n} \ell_i(F_{t-1} + \alpha_t f_t).$$

In the following, we will ignore the $\beta$ hyperparameter of the softmax, essentially assuming $\beta = 1$.

### 4.1 Deriving the BOPL Algorithm

Due to space constraints, we provide only a sketch of the derivation here, deferring the full derivation to Appendix B.1. The crux of the derivation (and subsequent analysis) is the *smoothness* of the loss function, $\ell_i$. Informally, a differentiable function is $\sigma$-smooth if its gradient is Lipschitz. In Appendix A, we prove a new upper bound on the smoothness coefficient of the softmax function—which is, to our knowledge, the tightest such bound, and may be of independent interest. Using this result, we prove that $\ell_i$ is $\frac{|r_i|}{2p_i}$-smooth.

Having established smoothness, we construct a recursive upper bound for each $\ell_i(F_t)$ that isolates the influence of $\alpha_t$ and $f_t$:

$$\begin{aligned} \ell_i(F_t) \;\le\; & \ell_i(F_{t-1}) \\ & - \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top (\alpha_t f_t(x_i)) \\ & + \frac{|r_i|}{4p_i} \left\| \alpha_t f_t(x_i) \right\|^2, \end{aligned}$$

where $\mathbf{a}_i$ denotes a *one-hot* encoding of the logged action, $a_i$, as a vector. Averaging over $i = 1, \ldots, n$, we obtain a recursive upper bound on the empirical risk, $\hat{L}(\pi_t, S)$.

From there, we obtain a closed-form expression for the ensemble weight that minimizes the upper bound, for any given predictor:

$$\alpha_t^\star \triangleq \frac{\frac{2}{n} \sum_{i=1}^{n} \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f_t(x_i)}{\frac{1}{n} \sum_{i=1}^{n} \frac{|r_i|}{p_i} \left\| f_t(x_i) \right\|^2}.$$

Observe that the numerator is proportional to a weighted average,

$$\sum_{i=1}^{n} w_i \operatorname{sgn}(r_i) \Big( f_t(x_i, a_i) - \mathop{\mathbb{E}}_{a \sim \pi_{t-1}(x_i)} [f_t(x_i, a)] \Big), \tag{5}$$

where $w_i \triangleq \frac{2|r_i|}{np_i} \pi_{t-1}(a_i \,|\, x_i) \ge 0$. If the weighted-average difference between $f_t(x_i, a_i)$ and the mean of $f_t(x_i, a)$ over $a \sim \pi_{t-1}(x_i)$ is nonzero, then $\alpha_t^\star \neq 0$ and $f_t$ will contribute to the ensemble. This is analogous to AdaBoost's *weak learning* condition: boosting can proceed as long as the base learner performs better than random

**Algorithm 1** Boosted Off-Policy Learning (BOPL)

```
Input: predictor class, $\mathcal{F}$; base learner; rounds, $T \geq 1$; scale, $Z > 0$
1: $F_0 \leftarrow 0$
2: for $t = 1, \ldots, T$ do
3:     $f_t \leftarrow \arg\max_{f\in\mathcal{F}} \left| \frac{1}{n}\sum_{i=1}^n \frac{r_i}{p_i}\pi_{t-1}(a_i\,|\,x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f(x_i) \right|$    ▷ base learner
           s.t. $\frac{1}{n}\sum_{i=1}^n \frac{|r_i|}{p_i}\left\|f(x_i)\right\|^2 = Z$
4:     $\alpha_t \leftarrow \frac{2}{nZ}\sum_{i=1}^n \frac{r_i}{p_i}\pi_{t-1}(a_i\,|\,x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f_t(x_i)$
5:     $F_t \leftarrow F_{t-1} + \alpha_t f_t$
```

guessing (i.e., accuracy greater than $1/2$) under a weighted empirical distribution. In fact, when $f_t$ is a $\{\pm 1\}$-valued classifier, our weak learning condition coincides with AdaBoost's (see Appendix D.2). If the weak learning condition is not satisfied, then the new predictor adds no value to the ensemble, and boosting should terminate.

Plugging $\alpha_t^\star$ into the recursive upper bound, we find that the predictor that minimizes the upper bound is

$$f_t^\star \in \operatorname*{arg\,max}_{f\in\mathcal{F}} \frac{\left(\frac{1}{n}\sum_{i=1}^n \frac{r_i}{p_i}\pi_{t-1}(a_i\,|\,x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f(x_i)\right)^2}{\frac{1}{n}\sum_{i=1}^n \frac{|r_i|}{p_i}\left\|f(x_i)\right\|^2}.$$

Critically, $f_t^\star$ is *invariant to scaling*; meaning, for any $c > 0$, $cf_t^\star$ is still optimal. Thus, we can fix its scale by constraining $\frac{1}{n}\sum_{i=1}^n \frac{|r_i|}{p_i}\left\|f(x_i)\right\|^2 = Z$, for any $Z > 0$, and simply maximize the numerator. This results in a constrained optimization problem that we refer to as the *base learning objective*, which is solved by a *base learner* for the given class of predictors, $\mathcal{F}$. We give two examples of base learners in Section 4.4.

The resulting boosting algorithm, dubbed *BOPL*, is given in Algorithm 1. Line 3 is the base learning objective. In practice, an early stopping condition can be inserted at the end of each iteration. If the gradient or $\alpha_t$ is zero, then no further progress can be made. Alternatively, one could stop when a validation metric indicates overfitting. One could also apply regularization to the base learner or the ensemble weights; we discuss this in Appendix E.

### 4.2 Analysis of BOPL

Using the analysis from our smoothness-based derivation (Appendix B.1), we can upper-bound BOPL's excess empirical risk; that is, the learned policy's suboptimality relative to an empirically optimal policy. The proof is deferred to Appendix C.1.

**Theorem 1.** *Given a dataset, $S$, let $\hat{L}^\star \triangleq \inf_\pi \hat{L}(\pi, S)$ denote the minimum empirical risk, and $\Delta_0 \triangleq \hat{L}(\pi_0, S) - \hat{L}^\star$ the excess empirical risk of the initial (uniformly random) ensemble policy, $\pi_0$. If Algorithm 1 is run for $T > 0$ rounds with $Z > 0$, producing policy $\pi_T$, then*

$$\hat{L}(\pi_T, S) - \hat{L}^\star \leq \Delta_0 \exp\left(-\frac{Z}{4\Delta_0}\sum_{t=1}^T \alpha_t^2\right). \qquad (6)$$

The bound decreases as long as each $\alpha_t \neq 0$, which happens when Equation 5 is nonzero—our analog of the weak learning condition. If this condition is met at every round, then the excess empirical risk eventually converges to a stationary point. Further, if $|\alpha_t| \geq \gamma > 0$ for all $t = 1, \ldots, T$, then the bound is exponentially decreasing in $T$.[2]

Though $Z$ appears to be a free parameter, recall that $Z$ implicitly defines the scale of the base learner, and that any tuning of $Z$ will be automatically compensated for in the ensemble weights. Thus, one should instead think of $Z$ as a property of the base learner.

Note that the infimum, $\inf_\pi$, used to define $\hat{L}^\star$ is not constrained to any particular class of policies. Indeed, the empirically optimal policy may not be a member of $\Pi_T$, the class of ensemble policies. Thus, there may implicitly be some approximation gap, $\inf_{\pi_T\in\Pi_T} \hat{L}(\pi_T, S) - \hat{L}^\star$, governed by the expressive power of $\Pi_T$ and the hardness of the dataset, which is difficult to quantify. That being said, note that $\hat{L}^\star$ and $\Delta_0$ are straightforward to compute for a given dataset.

In Appendix C.2, we show how Theorem 1 can be used to upper-bound BOPL's excess risk with respect to an optimal policy (i.e., risk minimizer). By decomposing the excess risk into several error terms—estimation error, generalization error and excess empirical risk—we can leverage existing tools for concentration and uniform convergence to upper-bound the first two terms. We provide an example bound that is monotonically decreasing in $n$ and $T$ when the Rademacher complexity of $\mathcal{F}$ is $\mathrm{o}(1)$ and the weak learning condition holds.

### 4.3 Surrogate Objective and BOPL-S

From the viewpoint of functional gradient descent (Friedman, 1999; Mason et al., 1999), the objective function in Equation 4 may be difficult to optimize for two reasons. First, the softmax function is not convex in $F_T$, and there are exponentially many local optima (Chen et al., 2019), so convergence to a global optimum is not guaranteed. Second, the gradient of the softmax with respect to $F_T$ van-

[2] As shown in Appendix D.2, when $f_t$ is $\{\pm 1\}$-valued, this condition is equivalent to always having error rate less than some $\gamma < 1/2$ under a weighted empirical distribution.

**Algorithm 2** Boosted Off-Policy Learning with Surrogate Objective (BOPL-S)

```
Input: predictor class, $\mathcal{F}$; base learner; rounds, $T \geq 1$; scale, $Z > 0$
1: $F_0 \leftarrow 0$
2: for $t = 1, \ldots, T$ do
3:     $f_t \leftarrow \arg\max_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^{n} \frac{r_i \xi_{i,t}}{p_i} (\mathbf{a}_i - \pi_{t-1}(x_i))^\top f(x_i) \right|$        ▷ base learner
           s.t. $\frac{1}{n} \sum_{i=1}^{n} \frac{|r_i| \sigma_i}{p_i} \left\| f(x_i) \right\|^2 = Z$
           with $\xi_{i,t} \leftarrow \begin{cases} \pi_{t-1}(a_i \,|\, x_i) & \text{if } r_i < 0 \\ 1 & \text{if } r_i \geq 0 \end{cases}$   and   $\sigma_i \leftarrow \begin{cases} \frac{1}{2} & \text{if } r_i < 0 \\ 1 & \text{if } r_i \geq 0 \end{cases}$
4:     $\alpha_t \leftarrow \frac{1}{nZ} \sum_{i=1}^{n} \frac{r_i \xi_{i,t}}{p_i} (\mathbf{a}_i - \pi_{t-1}(x_i))^\top f_t(x_i)$
5:     $F_t \leftarrow F_{t-1} + \alpha_t f_t$
```

ishes quickly, meaning the optimization can get “stuck” early on.

To circumvent these issues (in certain cases), we propose to boost a surrogate objective. For the time being, we assume that the $i^{\text{th}}$ reward is nonnegative, $r_i \geq 0$. Then, using the identity $-c(\ln z + 1) \geq -cz$, for all $c \in \mathbb{R}_+$ and $z \in \mathbb{R}$, we define a *surrogate loss function*,

$$\tilde{\ell}_i(F_t) \triangleq -\frac{r_i}{p_i}(\ln \pi(a_i \,|\, x_i; F_t) + 1) \geq \ell_i(F_t),$$

variants of which have been used in the literature on policy optimization (Le Roux, 2016; Ma et al., 2019; London & Sandler, 2019; Jeunen et al., 2020). When $r_i \geq 0$, we have that $\tilde{\ell}_i(F_t)$ is convex in $F_t$ and upper-bounds $\ell_i(F_t)$; moreover, $\tilde{\ell}_i(F_t)$ has the same minimum as $\ell_i(F_t)$, and its gradient is zero only at the minimum. Like the original loss function, the surrogate loss function is smooth—albeit with a different coefficient, which we bound in Appendix A. We can therefore establish a recursive upper bound on the surrogate loss; and hence, on the empirical risk.

However, we started out by assuming that the reward is nonnegative, and this is not always true—especially if we wish to support reward translation. When the reward is negative, $\tilde{\ell}_i$ is no longer an upper bound for $\ell_i$, nor is it convex. To handle this case, we can resort to the original loss function. For the $i^{\text{th}}$ training example, we use $\ell_i$ when the reward is negative, and $\tilde{\ell}_i$ when the reward is nonnegative. The resulting surrogate objective is

$$\tilde{L}(\pi_t, S) \triangleq \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{r_i < 0\} \ell_i(F_t) + \mathbb{1}\{r_i \geq 0\} \tilde{\ell}_i(F_t), \quad (7)$$

which is an upper bound for $\hat{L}(\pi_t, S)$ because $\ell_i(F_t) \leq \tilde{\ell}_i(F_t)$ when $r_i \geq 0$. In the presence of negative rewards, this function is still non-convex, and still suffers from vanishing gradients. However, our hope is that the examples with nonnegative rewards contribute enough gradient to keep the optimization moving in the right direction.

Applying our smoothness analysis to the surrogate objective, we derive Algorithm 2, which we call *BOPL-S* (for *surrogate*). See Appendix B.2 for the full derivation.

### 4.4 Base Learners

Algorithms 1 and 2 depend on a base learner to solve the optimization in line 3. This algorithm will depend, to some extent, on the class of predictors, $\mathcal{F}$. We now sketch base learning reductions (full details given in Appendix D) for two classes, real-valued functions (e.g., regression trees) and binary classifiers (e.g., decision stumps), both of which can be implemented by a variety of readily available, off-the-shelf tools. In both cases, we assume that $\mathcal{F}$ is *symmetric*; meaning, for every $f \in \mathcal{F}$, we also have $-f \in \mathcal{F}$.

**Regression.** Assume that $\mathcal{F}$ is a symmetric class of real-valued functions, $\mathcal{F} \subseteq \{\mathcal{X} \times \mathcal{A} \to \mathbb{R}\}$. Since we assume that $\mathcal{F}$ is symmetric, we can omit the absolute value from the base learning objective. We then convert the constrained optimization problem to the following unconstrained one via Lagrangian relaxation:

$$\begin{aligned} \arg\max_{f \in \mathcal{F}} \min_{\lambda \in \mathbb{R}} \; & \frac{1}{n} \sum_{i=1}^{n} \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i) (\mathbf{a}_i - \pi_{t-1}(x_i))^\top f(x_i) \\ & - \lambda \left( \frac{1}{n} \sum_{i=1}^{n} \frac{|r_i|}{p_i} \left\| f(x_i) \right\|^2 - Z \right). \end{aligned}$$

For every $Z$, there exists a $\lambda$ that is optimal, and vice versa. Since $Z$ is arbitrary, we can choose any $\lambda$ for the optimization. We therefore take $\lambda = 1/2$ and, after dropping terms that are irrelevant to the optimization and rearranging the expression, we obtain a weighted least-squares regression,

$$\arg\min_{f \in \mathcal{F}} \; \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} w_i \left( y_{i,a} - f(x_i, a) \right)^2,$$

with nonnegative weights, $w_i \triangleq \frac{|r_i|}{p_i}$, and pseudo-labels,

$$y_{i,a} \triangleq \operatorname{sgn}(r_i) \pi_{t-1}(a_i \,|\, x_i) (\mathbb{1}\{a = a_i\} - \pi_{t-1}(a \,|\, x_i)).$$

Note that this optimization does not explicitly constrain the scale of the predictor to a given $Z$. Thus, for Theorem 1 to hold, the predictor can optionally be rescaled. The full details of the reduction and the resulting algorithm are given in Appendix D.1.

**Binary Classification.** Assume that $\mathcal{F}$ a symmetric class of binary classifiers, $\mathcal{F} \subseteq \{\mathcal{X} \times \mathcal{A} \to \{\pm 1\}\}$. For every example, $i \in \{1, \ldots, n\}$, and action, $a \in \mathcal{A}$, we define a nonnegative weight,

$$w_{i,a} \triangleq \left| \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbb{1}\{a = a_i\} - \pi_{t-1}(a \,|\, x_i)) \right|,$$

and a $\{\pm 1\}$-valued pseudo-label,

$$y_{i,a} \triangleq \operatorname{sgn}(r_i)(2\,\mathbb{1}\{a = a_i\} - 1).$$

Then, using the identities

$$w_{i,a} y_{i,a} = \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbb{1}\{a = a_i\} - \pi_{t-1}(a \,|\, x_i))$$

and

$$\mathbb{1}\{y_{i,a} \neq f(x_i, a)\} = \frac{1}{2}(1 - y_{i,a} f(x_i, a)),$$

we derive an equivalence between the base learning objective and minimizing the weighted classification error:

$$\arg\max_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^{n} \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f(x_i) \right|$$
$$= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} w_{i,a} \mathbb{1}\{y_{i,a} \neq f(x_i, a)\}.$$

This optimization problem can be (approximately) solved by any learning algorithm for weighted binary classification. Since the predictions are $\{\pm 1\}$-valued, every $f \in \mathcal{F}$ satisfies $\|f(x)\|^2 = |\mathcal{A}|$. Thus, the base learning scale constraint is automatically satisfied by $Z = \frac{|\mathcal{A}|}{n} \sum_{i=1}^{n} \frac{|r_i|}{p_i}$, which is nonnegative as long as there is at least one nonzero reward in the dataset. The full details of the reduction and the resulting algorithm are given in Appendix D.2.

# 5 EXPERIMENTS

The following experiments are designed to answer several questions: first, whether boosted ensemble policies are competitive with other complex policy classes, such as deep neural networks; second, whether BOPL's off-policy learning objective (Equation 4) is more effective at training ensemble policies than the reward regression objective (Equation 2); and finally, whether BOPL-S's surrogate learning objective (Equation 7) is easier to optimize.

## 5.1 Data and Methodology

We use the standard supervised-to-bandit conversion (Beygelzimer & Langford, 2009) to simulate bandit feedback, since it allows us to compute accurate evaluation metrics from the ground-truth data. We report results on four public datasets, which have been used in related work (Swaminathan & Joachims, 2015a; London & Sandler, 2019): Covertype (Blackard & Dean, 1999), Fashion-MNIST (Xiao et al., 2017), Scene (Boutell et al., 2004) and TMC2007-500 (Srivastava & Zane-Ulman, 2005). Details about the datasets, partitioning, preprocessing and task reward structures are given in Appendix F.

For each method, we tune its associated hyperparameters via random search over a grid, whose limits were determined using the validation reward. Each hyperparameter configuration is used to train a policy on 10 simulated bandit datasets; then, each policy is evaluated on the validation data. The configuration with the highest average validation reward is selected, and its associated policies are evaluated on the testing data. Final rewards are averaged over the 10 trials (i.e., 10 trained policies).

With the exception of the logging policy, we evaluate all policies as argmax policies (i.e., softmax with $\beta \to \infty$), which deterministically selects the highest scoring (or, most likely) action. This eliminates the effect of exploration, and puts stochastic and deterministic policies on the same footing.

## 5.2 Algorithms Tested

Both Algorithm 1 (BOPL) and Algorithm 2 (BOPL-S) can be instantiated with two base learners, regression (suffix *-regr*) or binary classification (*-class*), for a total of four combinations. We use decision tree base learners throughout. Since boosted ensembles are complex model classes, we combat propensity overfitting using reward translation (Joachims et al., 2018).

To validate BOPL's learning objective, which directly optimizes an estimate of the ensemble policy's expected reward—rather than reward prediction error—we compare BOPL to several baselines based on reward regression (Equation 2). The first two baselines, prefixed *BRR* (boosted reward regression), use boosted regression trees, so as to control for the model class and isolate the effect of the learning objective. *BRR-gb* is our own implementation of a "vanilla" gradient boosting algorithm (Friedman, 1999) for least-squares regression. *BRR-xgb* is the highly optimized and state-of-the-art *XGBoost* (Chen & Guestrin, 2016) with the squared error loss. Both algorithms have a tunable learning rate (a.k.a. *shrinkage* parameter), which effectively scales the ensemble weights[3]. The third baseline, *DRR* (deep reward regression), trains a neural network regressor. Thus, DRR differs from BOPL in both the model class and learning objective. Note that none of the reward regression baselines require reward translation.

Finally, to focus on the question of boosting versus deep learning for off-policy learning, we compare to *Bandit-*

[3] We tried a learning rate with BOPL and BOPL-S as well, but found that it was not helpful.

Table 1: Average reward on the test data, averaged over 10 trials, with $95\%$ confidence intervals. Best scores for each dataset are bolded. Multiple bold cells occur when confidence intervals overlap with those of the best score.

| | **Covertype** | **Fashion-MNIST** | **Scene** | **TMC2007-500** |
|---|---|---|---|---|
| **Logging** | $0.4907 \pm 0.0024$ | $0.4708 \pm 0.0009$ | $0.4317 \pm 0.0042$ | $0.3187 \pm 0.0029$ |
| **BRR-gb** | $0.9033 \pm 0.0015$ | $0.8652 \pm 0.0012$ | $0.6328 \pm 0.0319$ | $0.7288 \pm 0.0046$ |
| **BRR-xgb** | $0.9465 \pm 0.0004$ | $0.8739 \pm 0.0011$ | $0.6854 \pm 0.0189$ | $\mathbf{0.7742 \pm 0.0034}$ |
| **DRR** | $0.8878 \pm 0.0011$ | $\mathbf{0.8947 \pm 0.0019}$ | $0.7636 \pm 0.0098$ | $\mathbf{0.7787 \pm 0.0038}$ |
| **BanditNet** | $0.8565 \pm 0.0019$ | $\mathbf{0.8921 \pm 0.0025}$ | $\mathbf{0.7794 \pm 0.0124}$ | $0.7603 \pm 0.0076$ |
| **BOPL-regr** | $0.9504 \pm 0.0005$ | $0.8893 \pm 0.0016$ | $\mathbf{0.7778 \pm 0.0105}$ | $0.7361 \pm 0.0047$ |
| **BOPL-class** | $0.9262 \pm 0.0010$ | $0.8775 \pm 0.0012$ | $0.7651 \pm 0.0112$ | $0.7131 \pm 0.0039$ |
| **BOPL-S-regr** | $\mathbf{0.9531 \pm 0.0007}$ | $0.8876 \pm 0.0022$ | $\mathbf{0.7703 \pm 0.0132}$ | $0.7399 \pm 0.0030$ |
| **BOPL-S-class** | $0.9191 \pm 0.0010$ | $0.8759 \pm 0.0014$ | $\mathbf{0.7877 \pm 0.0091}$ | $0.7339 \pm 0.0051$ |

Table 2: Direct method (DM) reward estimation versus actual test reward.

| | **Scene** | | **TMC2007-500** | |
|---|---|---|---|---|
| | **DM Reward** | **True Reward** | **DM Reward** | **True Reward** |
| **BRR-xgb** | $\mathbf{0.5126 \pm 0.0103}$ | $0.6854 \pm 0.0189$ | $\mathbf{0.7609 \pm 0.0027}$ | $\mathbf{0.7742 \pm 0.0034}$ |
| **BOPL-regr** | $0.4972 \pm 0.0099$ | $\mathbf{0.7778 \pm 0.0105}$ | $0.7023 \pm 0.0034$ | $0.7361 \pm 0.0047$ |

*Net* (Joachims et al., 2018). Like BOPL, BanditNet trains a softmax policy by optimizing the IPS estimator (Equation 3), with reward translation to combat propensity overfitting. However, BanditNet's underlying model (i.e., action scoring function) is a neural network. Thus, this comparison controls for the learning objective and isolates the effect of the model class.

Further details about the algorithm implementations and hyperparameters can be found in Appendix G.

### 5.3 Results

Table 1 summarizes the results of our experiments. Overall, on two of the four datasets, at least one of the proposed BOPL variants performs best, and the other BOPL variants are typically competitive as well. In particular, BOPL-S-regr is always either the best or statistically tied with the best variant. The following investigates our stated research questions in greater detail.

**Is boosting competitive with deep learning?** While BOPL significantly outperforms all baselines on Covertype, the deep learning methods, DRR and BanditNet, outperform other methods on Fashion-MNIST. These results are not that surprising, as tree-based models are known to perform well on tabular data (Grinsztajn et al., 2022) and deep learning is known to excel at computer vision problems. We also find that BanditNet is statistically tied with BOPL on Scene, and BRR-xgb is statistically tied with DRR on TMC2007-500. It is worth noting, however, that the deep learning methods have more hyperparameters to tune, take much longer to train, and consume more resources—requiring GPUs to run in a reasonable amount of time, while the boosting methods only use CPUs. Indeed, boosted tree ensembles live up to to their reputation of performing robustly without complex tuning or massive computing resources. From this perspective, boosting not only performs competitively (if not better), it is often a more *practical* choice.

**How does BOPL compare to reward regression?** Overall, DRR is the strongest reward regression baseline, and it outperforms BOPL on two datasets. However, BOPL outperforms BRR (which controls for the model class) on three out of four datasets. To understand why, consider the fact that BOPL directly maximizes the IPS estimate of the policy's expected reward, while BRR can be viewed as *indirectly* optimizing a *different* reward estimator: the so-called *direct method* (DM). Instead of importance weighting, DM uses a reward regressor to predict the reward for each action selected by the target policy. Clearly, the policy that maximizes the DM estimate is one that always picks the best action according to the reward regressor. Therefore, whether BOPL or BRR performs better can now be understood in terms of the bias-variance trade-offs of the IPS and DM estimators.

To illustrate this point, Table 2 analyzes the bias of the DM estimator (using BRR-xgb) on the two datasets where BOPL and BRR perform best and worst, respectively. On Scene, where BRR performs poorly, the DM estimate of the test reward is highly biased; it predicts $0.5126$ for the BRR-xgb policy when the true reward is $0.6854$, and it is even less accurate when estimating the reward of the BOPL-regr policy. This means that picking a policy based on the DM estimator is bound to be unreliable, which explains BRR's poor performance on Scene. On the other hand, the DM estimator is substantially less biased on TMC2007-500, which explains BRR's relatively good performance on that dataset. Unfortunately, it is difficult to determine when, and to what degree, the DM estimator will be biased, so it is difficult to say ahead of time whether reward regression will be successful. In contrast, the IPS estimator (used in BOPL) has reliable tools to control the bias-variance trade-off (Dudik et al., 2011; Swaminathan & Joachims, 2015b).

**Is BOPL-S's surrogate objective easier to optimize?** Recall that BOPL's learning objective is non-convex, and its gradients tend to vanish quickly. This motivated BOPL-S, which boosts a surrogate objective that partially addresses BOPL's issues. For any example with nonnegative rewards, BOPL-S's objective is convex and has a non-vanishing gradient. Unfortunately, examples with negative rewards are subject to the same problems as BOPL, but our hope is that correcting the nonnegative-reward examples will help. To validate this claim, we compare BOPL and BOPL-S on the Covertype data, since this is the dataset where BOPL-S has a statistically significant advantage. Since the rewards for this dataset are nonnegative, negative reward can only come from reward translation. We therefore evaluate two variants of BOPL-S: one with reward translation, and one without called *BOPL-CS* since its surrogate objective is convex. Figure 1 plots the norm of the gradient, as well as the average reward on the training and testing data, as they evolve over a single run of each algorithm.[4] We find that BOPL's gradient is almost monotonically decreasing in magnitude, meaning the optimization is largely decelerating, and its rewards have a bumpy upward trajectory compared to the surrogate objectives. In contrast, BOPL-CS's gradient shoots up in early rounds, meaning the optimization is making fast progress. Indeed, we see BOPL-CS's rewards rise faster than BOPL's. BOPL-S's gradient looks like a somewhat "softened" version of BOPL-CS's. This is likely because, with reward translation, some examples still use BOPL's objective. That being said, BOPL-S ultimately achieves the highest reward. Therefore, while convexity is helpful, reward translation may be more so.

# 6 CONCLUSIONS AND FUTURE WORK

We proposed the first boosting algorithm for off-policy learning from logged bandit feedback. Both our theory and our empirical results highlight the importance of boosting the right off-policy objective, and suggest that our boosting algorithm is a robust alternative to deep off-policy learning. This novel ability to directly optimize a policy's expected reward via boosting opens a new space of research questions and further improvements. In particular, one could continue to refine the algorithm with more sophisticated surrogate objectives (e.g., (Le Roux, 2016; Chen et al., 2019)), optimize estimators beyond vanilla IPS (e.g., (Dudik et al., 2011; Swaminathan & Joachims, 2015b)) so as to reduce variance, or examine off-policy regularizers that do not easily decompose over the ensemble (e.g., (Swaminathan & Joachims, 2015a)).

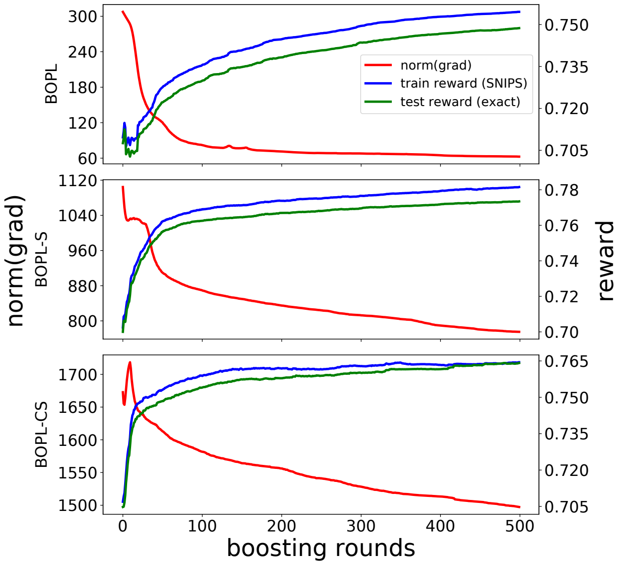

Figure 1: Plots of the gradient norm and train/test reward over a single run of BOPL, BOPL-S and BOPL-CS on the Covertype data. We use the self-normalized IPS estimator (Swaminathan & Joachims, 2015b) to estimate reward on the training data. Note that the hyperparameters used in these plots are not the optimized ones used in Table 1.

**Acknowledgements**

We thank Emily Butler, Bob Kaucic, Gert Lanckriet and Mani Sethuraman from Amazon Music, as well as Leo Dirac, Avi Geiger and Paulina Varshavskaya from Groundlight.ai, for supporting this work.

## References

N. Abe, B. Zadrozny, and J. Langford. An iterative method for multi-class cost-sensitive learning. In *Knowledge Discovery and Data Mining*, 2004.

D. Abel, A. Agarwal, F. Diaz, A. Krishnamurthy, and R. Schapire. Exploratory gradient boosting for reinforcement learning in complex domains. *CoRR*, abs/1603.04119, 2016.

[4] We use a single run to illustrate a typical optimization trajectory in which BOPL-S outperforms BOPL. Given the inter-run variability of the optimization, averaging over multiple runs might wash out any non-monotonicity in the trajectories of individual runs, which would hide the phenomena we are trying to present.

A. Agarwal, S. Kakade, J. Lee, and G. Mahajan. Optimality and approximation with policy gradient methods in Markov decision processes. In *Conference on Learning Theory*, 2020.

R. Appel, X. Burgos-Artizzu, and P. Perona. Improved multi-class cost-sensitive boosting via estimation of the minimum-risk class. *CoRR*, abs/1607.03547, 2016.

P. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3, 2003.

S. Ben-David, N. Eiron, and P. Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3), 2003.

A. Beygelzimer and J. Langford. The offset tree for learning with partial labels. In *Knowledge Discovery and Data Mining*, 2009.

J. Blackard and D. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3), 1999.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Neural Information Processing Systems*, 2007.

L. Bottou, J. Peters, J. Quiñonero-Candela, D. Charles, D. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14, 2013.

M. Boutell, J. Luo, X. Shen, and C. Brown. Learning multi-label scene classification. *Pattern recognition*, 37 (9), 2004.

N. Brukhim and E. Hazan. Online boosting with bandit feedback. In *Conference on Algorithmic Learning Theory*, 2021.

N. Brukhim, E. Hazan, and K. Singh. A boosting approach to reinforcement learning. *CoRR*, abs/2108.09767, 2021.

M. Chen, R. Gummadi, C. Harris, and D. Schuurmans. Surrogate objectives for batch policy optimization in one-step decision making. In *Neural Information Processing Systems*, 2019.

S.-T. Chen, H.-T. Lin, and C.-J. Lu. Boosting with online binary learners for the multiclass bandit problem. In *International Conference on Machine Learning*, 2014.

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Knowledge Discovery and Data Mining*, 2016.

T. Chen, M., Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274, 2015.

J. Duchi and Y. Singer. Boosting with structural sparsity. In *International Conference on Machine Learning*, 2009.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

M. Dudik, J. Langford, and L. Lihong. Doubly robust policy evaluation and learning. In *International Conference on Machine Learning*, 2011.

W. Fan, S. Stolfo, J. Zhang, and P. Chan. AdaCost: Misclassification cost-sensitive boosting. In *International Conference on Machine Learning*, 1999.

L. Faury, U. Tanielian, F. Vasile, E. Smirnova, and E. Dohmatob. Distributionally robust counterfactual risk minimization. In *AAAI*, 2020.

M. Feurer, J. van Rijn, A. Kadra, P. Gijsbers, N. Mallik, S. Ravi, A. Müller, J. Vanschoren, and F. Hutter. OpenML-Python: an extensible Python API for OpenML. *CoRR*, abs/1911.02490, 2019.

Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121, 1995.

Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 1997.

J. Friedman. Greedy function approximation: a gradient boosting machine. Technical report, Stanford University, 1999.

B. Gao and L. Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *ArXiv*, abs/1704.00805, 2017.

L. Grinsztajn, E. Oyallon, and G. Varoquaux. Why do tree-based models still outperform deep learning on tabular data? *CoRR*, abs/2207.08815, 2022.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301), 1963.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

E. Ionides. Truncated importance sampling. *Journal of Computational and Graphical Statistics*, 17(2):295–311, 2008.

O. Jeunen, D. Rohde, F. Vasile, and M. Bompaire. Joint policy-value learning for recommendation. In *Knowledge Discovery and Data Mining*, 2020.

T. Joachims, A. Swaminathan, and M. de Rijke. Deep learning with logged bandit feedback. In *International Conference on Learning Representations*, 2018.

N. Kallus. More efficient policy learning via optimal retargeting. *Journal of the American Statistical Association*, 116, 2019.

M. Kearns and L. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. In *Symposium on Theory of Computing*, 1989.

N. Le Roux. Efficient iterative policy optimization. *CoRR*, abs/1612.08967, 2016.

B. London and T. Sandler. Bayesian counterfactual risk minimization. In *International Conference on Machine Learning*, 2019.

Y. Ma, Y.-X. Wang, and B. Narayanaswamy. Imitation-regularized offline learning. In *Artificial Intelligence and Statistics*, 2019.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *Neural Information Processing Systems*, 1999.

A. Maurer. A vector-contraction inequality for Rademacher complexities. In *Conference on Algorithmic Learning Theory*, 2016.

J. Mei, C. Xiao, C. Szepesvari, and D. Schuurmans. On the global convergence rates of softmax policy gradient methods. In *International Conference on Machine Learning*, 2020.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. Adaptive computation and machine learning. MIT Press, 2012. ISBN 978-0-262-01825-8.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2011.

R. Schapire. The strength of weak learnability. *Machine Learning*, 5, 1990.

R. Schapire and Y. Freund. *Boosting: Foundations and Algorithms*. Adaptive computation and machine learning. MIT Press, 2012. ISBN 9780262017183.

A. Srivastava and B. Zane-Ulman. Discovering recurring anomalies in text reports regarding complex space systems. In *Aerospace Conference*, 2005.

A. Strehl, J. Langford, L. Li, and S. Kakade. Learning from logged implicit exploration data. In *Neural Information Processing Systems*, 2010.

A. Swaminathan and T. Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research*, 2015a.

A. Swaminathan and T. Joachims. The self-normalized estimator for counterfactual learning. In *Neural Information Processing Systems*, 2015b.

J. Vanschoren, J. van Rijn, B. Bischl, and L. Torgo. OpenML: networked science in machine learning. *SIGKDD Explorations*, 15(2), 2013.

H. Wu and M. Wang. Variance regularized counterfactual risk minimization via variational divergence minimization. In *International Conference on Machine Learning*, 2018.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

C. Yang, J. Fan, Z. Wu, and M. Udell. AutoML pipeline selection: Efficiently navigating the combinatorial space. In *Knowledge Discovery and Data Mining*, 2020.

D. Zhang, Y. Jung, and A. Tewari. Online multiclass boosting with bandit feedback. In *Artificial Intelligence and Statistics*, 2019.

# A SMOOTHNESS

Our algorithm derivations rely on a property of the loss function known as *smoothness*.

**Definition 1.** A differentiable function, $\phi : \Omega \to \mathbb{R}$, is *$\sigma$-smooth* if, for all $\omega, \omega' \in \Omega$,

$$\left\|\nabla\phi(\omega) - \nabla\phi(\omega')\right\|^2 \leq \frac{\sigma}{2}\left\|\omega - \omega'\right\|_2^2 .$$

In this appendix, we prove that two loss functions—one used in Algorithm 1; the other used in Algorithm 2—are smooth. We begin with several technical lemmas, from which the smoothness proofs directly follow.

**Lemma 1.** *For a symmetric, rank-1 matrix, $\mathbf{A} = \mathbf{x}\mathbf{x}^\top$, with $\mathbf{x} \in \mathbb{R}^{n\times n}$, its spectral norm (i.e., largest eigenvalue) is $\|\mathbf{A}\|_2 = \|\mathbf{x}\|_2^2$.*

*Proof.* By definition, the spectral norm of $\mathbf{A}$ is

$$\begin{aligned}
\|\mathbf{A}\|_2 &= \sup_{\mathbf{u}:\|\mathbf{u}\|_2=1} \|\mathbf{A}\mathbf{u}\|_2 \\
&= \sup_{\mathbf{u}:\|\mathbf{u}\|_2=1} \left\|\mathbf{x}\mathbf{x}^\top\mathbf{u}\right\|_2 \\
&= \|\mathbf{x}\|_2 \sup_{\mathbf{u}:\|\mathbf{u}\|_2=1} (\mathbf{x}^\top\mathbf{u}) \\
&= \|\mathbf{x}\|_2 \|\mathbf{x}\|_2 .
\end{aligned}$$

The last equality follows from an alternate definition of the Euclidean norm. □

**Lemma 2.** *The Hessian of the softmax function, $\pi(a \mid x; f, \beta) = \frac{\exp(\beta f(x,a))}{\sum_{a'\in\mathcal{A}} \exp(\beta f(x,a'))}$, has bounded spectral norm:*

$$\left\|\nabla^2\pi(a \mid x; f, \beta)\right\|_2 \leq 2\beta^2\pi(a \mid x; f, \beta)\big(1 - \pi(a \mid x; f, \beta)\big). \tag{8}$$

*Therefore, $\pi(a \mid x; f, \beta)$ is $\frac{\beta^2}{2}$-smooth.*

*Proof.* For a twice-differentiable function (like the softmax), smoothness is equivalent to having a uniformly upper-bounded second derivative. Consequently, our first step will be to derive a formula for the Hessian, $\nabla^2\pi(a \mid x; f, \beta)$, with respect to $f(x)$ (which, for boosting, is an ensemble prediction, $F_t(x)$). We will show that the Hessian decomposes into the sum of two symmetric, rank-1 matrices. We then upper-bound the norm of each matrix using Lemma 1 and some additional reasoning.

To simplify notation, we will omit $x$, $f$ and $\beta$, and simply write $\pi \triangleq \pi(x; f, \beta) \in \mathbb{R}^{|\mathcal{A}|}$ to denote the vector of softmax probabilities, and $\pi(a) \triangleq \pi(a \mid x; f, \beta)$ to denote the conditional probability of $a$.

Using the log derivative trick and the product rule, we have that

$$\begin{aligned}
\nabla^2\pi(a) &= \nabla\big(\nabla\pi(a)\big) \\
&= \nabla\big(\pi(a)\nabla\ln\pi(a)\big) \\
&= (\nabla\pi(a))(\nabla\ln\pi(a))^\top + \pi(a)\nabla^2\ln\pi(a) \\
&= \pi(a)\Big(\underbrace{(\nabla\ln\pi(a))(\nabla\ln\pi(a))^\top}_{\mathbf{H}_1} + \underbrace{\nabla^2\ln\pi(a)}_{\mathbf{H}_2}\Big).
\end{aligned}$$

By the sub-additivity of the norm,

$$\left\|\nabla^2\pi(a)\right\|_2 = \|\pi(a)(\mathbf{H}_1 + \mathbf{H}_2)\|_2 \leq \pi(a)\big(\|\mathbf{H}_1\|_2 + \|\mathbf{H}_2\|_2\big),$$

so we can upper-bound the norms of $\mathbf{H}_1$ and $\mathbf{H}_2$ separately.

Via Lemma 1, we have that

$$\|\mathbf{H}_1\|_2 = \left\|(\nabla\ln\pi(a))(\nabla\ln\pi(a))^\top\right\|_2 = \|\nabla\ln\pi(a)\|_2^2 .$$

The gradient of $\ln \pi(a)$ is $\beta(\mathbf{a} - \pi)$, where $\mathbf{a}$ (without subscript) denotes the one-hot encoding of $a$. Therefore,

$$\|\nabla \ln \pi(a)\|_2^2 = \beta^2 \|\mathbf{a} - \pi\|_2^2 = \beta^2\big(\mathbf{a}^\top \mathbf{a} - 2\mathbf{a}^\top \pi + \pi^\top \pi\big) = \beta^2\big(1 - 2\pi(a) + \pi^\top \pi\big).$$

Turning now to $\mathbf{H}_2$, we note that $\nabla^2 \ln \pi(a_i)$ is the covariance of the distribution $\pi$, scaled by $-\beta^2$:

$$\nabla^2 \ln \pi(a) = -\beta^2 \mathop{\mathbb{E}}_{a' \sim \pi} \left[(\mathbf{a}' - \pi)(\mathbf{a}' - \pi)^\top\right]. \tag{9}$$

Thus,

$$\begin{aligned} \|\mathbf{H}_2\|_2 &= \left\|\nabla^2 \ln \pi(a)\right\|_2 \\ &= \beta^2 \left\| \mathop{\mathbb{E}}_{a' \sim \pi} \left[(\mathbf{a}' - \pi)(\mathbf{a}' - \pi)^\top\right]\right\|_2 \\ &\leq \beta^2 \mathop{\mathbb{E}}_{a' \sim \pi} \left[\left\|(\mathbf{a}' - \pi)(\mathbf{a}' - \pi)^\top\right\|_2\right] \\ &= \beta^2 \mathop{\mathbb{E}}_{a' \sim \pi} \left[\|\mathbf{a}' - \pi\|_2^2\right]. \end{aligned} \tag{10}$$

The inequality is from Jensen's inequality; the final equality is from Lemma 1. Continuing,

$$\mathop{\mathbb{E}}_{a' \sim \pi} \left[\|\mathbf{a}' - \pi\|_2^2\right] = \mathop{\mathbb{E}}_{a' \sim \pi} \left[\mathbf{a}'^\top \mathbf{a}' - 2\mathbf{a}'^\top \pi + \pi^\top \pi\right] = 1 - 2 \mathop{\mathbb{E}}_{a' \sim \pi}[\mathbf{a}']^\top \pi + \pi^\top \pi = 1 - \pi^\top \pi. \tag{11}$$

Finally, combining the inequalities, we have

$$\begin{aligned} \left\|\nabla^2 \pi(a)\right\|_2 &\leq \pi(a)\big(\, \|\mathbf{H}_1\|_2 + \|\mathbf{H}_2\|_2 \,\big) \\ &\leq \beta^2 \pi(a)\big((1 - 2\pi(a) + \pi^\top \pi) + (1 - \pi^\top \pi)\big) \\ &= 2\beta^2 \pi(a)(1 - \pi(a)), \end{aligned}$$

which proves Equation 8. To finish the proof, we note that

$$2\beta^2 \pi(a)(1 - \pi(a)) \leq 2\beta^2 \times \frac{1}{4} = \frac{\beta^2}{2},$$

which follows from the fact that $\pi(a)(1 - \pi(a)) \leq \frac{1}{4}$ for $\pi(a) \in [0, 1]$. □

*Remark* 1. To the best of our knowledge, Lemma 2 is the best (and possibly first documented) upper bound on the smoothness coefficient of the softmax. It complements work by Gao & Pavel (2017), who showed that the softmax is $\beta$-Lipschitz. We are aware of only two other related smoothness results: Agarwal et al. (2020) showed that the function $\pi^\top \mathbf{r}$, for $\mathbf{r} \in \mathbb{R}^{|\mathcal{A}|}$, is $5\,\|\mathbf{r}\|_\infty$-smooth; similarly, Mei et al. (2020) showed that $\pi^\top \mathbf{r}$, for $\mathbf{r} \in [0, 1]^{|\mathcal{A}|}$, is $\frac{5}{2}$-smooth. Equation 8 can be used to show that $\pi^\top \mathbf{r}$, for $\mathbf{r} \in \mathbb{R}^{|\mathcal{A}|}$, is actually $2\,\|\mathbf{r}\|_\infty$-smooth, thereby improving upon both prior results. △

**Lemma 3.** *The log-softmax function,* $\ln \pi(a \,|\, x; f, \beta)$*, is* $\beta^2(1 - |\mathcal{A}|^{-1})$*-smooth.*

*Proof.* The proof builds on the proof of Lemma 2. We will reuse our previous notation, where $\pi(a) \triangleq \pi(a \,|\, x; f, \beta)$ and $\pi \triangleq \pi(x; f, \beta)$. Recall (from Equation 9) that the Hessian of the log-softmax is $\nabla^2 \ln \pi(a) = -\beta^2\, \mathbb{E}_{a' \sim \pi}\left[(\mathbf{a}' - \pi)(\mathbf{a}' - \pi)^\top\right]$, and (via Equations 10 and 11) its norm has upper bound

$$\left\|\nabla^2 \ln \pi(a)\right\|_2 \leq \beta^2(1 - \pi^\top \pi).$$

Since $\pi$ is constrained to the simplex ($\|\pi\|_1 = 1$), it is straightforward to show that

$$\inf_{\pi : \|\pi\|_1 = 1} \|\pi\|_2^2 \geq |\mathcal{A}|^{-1}.$$

Thus,

$$\beta^2(1 - \pi^\top \pi) \leq \beta^2(1 - |\mathcal{A}|^{-1}),$$

which completes the proof. □

Having established that the softmax and log-softmax are smooth, we are now ready to prove our main smoothness results.

**Proposition 1.** *The loss function, $\ell_i(F_t) = -\frac{r_i}{p_i}\pi(a_i \,|\, x_i; F_t)$, is $\frac{|r_i|}{2p_i}$-smooth.*

*Proof.* Since the softmax is $\frac{1}{2}$-smooth (for $\beta = 1$), we have that

$$\left\|\nabla^2 \ell_i(F_t)\right\|_2 = \frac{|r_i|}{p_i}\left\|\nabla^2 \pi(a_i \,|\, x_i; F_t)\right\|_2 \leq \frac{|r_i|}{2p_i}.$$

Thus completes the proof. □

**Proposition 2.** *The surrogate loss function, $\tilde{\ell}_i(F_t) = -\frac{r_i}{p_i}(\ln \pi(a_i \,|\, x_i; F_t) + 1)$, is $\frac{|r_i|}{p_i}$-smooth.*

*Proof.* To prove Proposition 2, we will simplify Lemma 3 by noting that $\beta^2(1 - |\mathcal{A}|^{-1}) \leq \beta^2$. Then, for $\beta = 1$, the log-softmax is 1-smooth. Therefore,

$$\left\|\nabla^2 \tilde{\ell}_i(F_t)\right\|_2 = \frac{|r_i|}{p_i}\left\|\nabla^2 \ln \pi(a_i \,|\, x_i; F_t)\right\|_2 \leq \frac{|r_i|}{p_i},$$

which completes the proof. □

# B DERIVATIONS OF BOOSTING ALGORITHMS

This appendix contains the full derivations of our boosting algorithms, which were deferred from the main paper.

## B.1 Derivation of Algorithm 1 (BOPL)

In this section, we give an unabridged derivation of Algorithm 1. It all starts by applying our smoothness result (Proposition 1) to construct a recursive upper bound on $\ell_i$ that isolates the influence of $\alpha_t$ and $f_t$. To do so, we use the following technical lemma.

**Lemma 4.** *If $\phi : \Omega \to \mathbb{R}$ is $\sigma$-smooth, then for all $\omega, \omega' \in \Omega$,*

$$\phi(\omega) \leq \phi(\omega') + \nabla\phi(\omega')^\top(\omega - \omega') + \frac{\sigma}{2}\left\|\omega - \omega'\right\|_2^2.$$

To apply Lemma 4, first note that the gradient of $\ell_i$ with respect to $F_t(x_i)$ is

$$\nabla \ell_i(F_t) = -\frac{r_i}{p_i}\pi(a_i \,|\, x_i; F_t)(\mathbf{a}_i - \pi(x_i; F_t)) = -\frac{r_i}{p_i}\pi_t(a_i \,|\, x_i)(\mathbf{a}_i - \pi_t(x_i)),$$

where $\mathbf{a}_i$ denotes the one-hot encoding of $a_i$. Combining this with Proposition 1 and Lemma 4, we have that

$$\ell_i(F_t) \leq \ell_i(F_{t-1}) - \frac{r_i}{p_i}\pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top(\alpha_t f_t(x_i)) + \frac{|r_i|}{4p_i}\left\|\alpha_t f_t(x_i)\right\|^2. \tag{12}$$

Averaging Equation 12 over $i = 1, \ldots, n$, we obtain a recursive upper bound:

$$\hat{L}(\pi_t, S) \leq \hat{L}(\pi_{t-1}, S) - \frac{1}{n}\sum_{i=1}^{n}\left[\frac{r_i}{p_i}\pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top(\alpha_t f_t(x_i)) - \frac{|r_i|}{4p_i}\left\|\alpha_t f_t(x_i)\right\|^2\right], \tag{13}$$

Observe that the bound is quadratic in the ensemble weight, $\alpha_t$. Thus, we can obtain a closed-form expression for the ensemble weight that minimizes the upper bound, for any given predictor:

$$\alpha_t^\star \triangleq \frac{\frac{2}{n}\sum_{i=1}^{n}\frac{r_i}{p_i}\pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f_t(x_i)}{\frac{1}{n}\sum_{i=1}^{n}\frac{|r_i|}{p_i}\left\|f_t(x_i)\right\|^2}. \tag{14}$$

Plugging $\alpha_t^\star$ into Equation 13 yields another recursive upper bound:

$$\hat{L}(\pi_t, S) \leq \hat{L}(\pi_{t-1}, S) - \frac{\left(\frac{1}{n}\sum_{i=1}^n \frac{r_i}{p_i}\pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f_t(x_i)\right)^2}{\frac{1}{n}\sum_{i=1}^n \frac{|r_i|}{p_i}\left\|f_t(x_i)\right\|^2}. \tag{15}$$

From this, it is clear that an optimal predictor at round $t$ is

$$f_t^\star \in \operatorname*{arg\,max}_{f\in\mathcal{F}} \frac{\left(\frac{1}{n}\sum_{i=1}^n \frac{r_i}{p_i}\pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f(x_i)\right)^2}{\frac{1}{n}\sum_{i=1}^n \frac{|r_i|}{p_i}\left\|f(x_i)\right\|^2}. \tag{16}$$

Since $f_t^\star$ is invariant to scaling, we can fix its scale by constraining $\frac{1}{n}\sum_{i=1}^n \frac{|r_i|}{p_i}\left\|f(x_i)\right\|^2 = Z$, for $Z > 0$. We can then simply maximize the magnitude of the numerator in Equation 16, subject to this constraint. This results in the base learning objective:

$$\max_{f\in\mathcal{F}} \left|\frac{1}{n}\sum_{i=1}^n \frac{r_i}{p_i}\pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f(x_i)\right| \quad \text{s.t.} \quad \frac{1}{n}\sum_{i=1}^n \frac{|r_i|}{p_i}\left\|f(x_i)\right\|^2 = Z. \tag{17}$$

This optimization becomes line 3 of Algorithm 1. Equation 14 becomes line 4.

### B.2 Derivation of Algorithm 2 (BOPL-S)

Recall from Section 4.3 that we use a composite of the original loss function, $\ell_i$, and the surrogate loss function, $\tilde{\ell}_i$, to upper bound the empirical risk:

$$\hat{L}(\pi_t, S) \leq \tilde{L}(\pi_t, S) = \frac{1}{n}\sum_{i=1}^n \mathbb{1}\{r_i < 0\}\ell_i(F_t) + \mathbb{1}\{r_i \geq 0\}\tilde{\ell}_i(F_t).$$

Since both $\ell_i$ and $\tilde{\ell}_i$ are smooth (see Appendix A), we can construct a recursive upper bound on the righthand side that isolates $\alpha_t$ and $f_t$. Recall the upper bound for $\ell_i$ given in Equation 12. Further, using Proposition 2 and Lemma 4, and noting the gradient,

$$\nabla\tilde{\ell}_i(F_t) = -\frac{r_i}{p_i}(\mathbf{a}_i - \pi_t(x_i)),$$

we have that

$$\tilde{\ell}_i(F_t) \leq \tilde{\ell}_i(F_{t-1}) - \frac{r_i}{p_i}(\mathbf{a}_i - \pi_{t-1}(x_i))^\top(\alpha_t f_t(x_i)) + \frac{|r_i|}{2p_i}\left\|\alpha_t f_t(x_i)\right\|^2. \tag{18}$$

Thus, combining Equations 12 and 18, we obtain a recursive upper bound,

$$\tilde{L}(\pi_t, S) \leq \tilde{L}(\pi_{t-1}, S) - \frac{1}{n}\sum_{i=1}^n \left[\frac{r_i\xi_{i,t}}{p_i}(\mathbf{a}_i - \pi_{t-1}(x_i))^\top(\alpha_t f_t(x_i)) - \frac{|r_i|\sigma_i}{2p_i}\left\|\alpha_t f_t(x_i)\right\|^2\right],$$

$$\text{where} \quad \xi_{i,t} \triangleq \begin{cases} \pi_{t-1}(a_i \,|\, x_i) & \text{if } r_i < 0, \\ 1 & \text{if } r_i \geq 0; \end{cases} \quad \text{and} \quad \sigma_i \triangleq \begin{cases} \frac{1}{2} & \text{if } r_i < 0, \\ 1 & \text{if } r_i \geq 0. \end{cases}$$

The rest of the derivation proceeds similarly to Appendix B.1. Solving for the optimal ensemble weight, we get

$$\alpha_t^\star \triangleq \frac{\frac{1}{n}\sum_{i=1}^n \frac{r_i\xi_{i,t}}{p_i}(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f_t(x_i)}{\frac{1}{n}\sum_{i=1}^n \frac{|r_i|\sigma_i}{p_i}\left\|f_t(x_i)\right\|^2}. \tag{19}$$

Then, using this value in the upper bound, we get

$$\tilde{L}(\pi_t, S) \leq \tilde{L}(\pi_{t-1}, S) - \frac{\left(\frac{1}{n}\sum_{i=1}^n \frac{r_i\xi_{i,t}}{p_i}(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f_t(x_i)\right)^2}{\frac{2}{n}\sum_{i=1}^n \frac{|r_i|\sigma_i}{p_i}\left\|f_t(x_i)\right\|^2}.$$

Therefore, an optimal predictor at round $t$ is given by

$$f_t^\star \in \arg\max_{f\in\mathcal{F}} \frac{\left(\frac{1}{n}\sum_{i=1}^n \frac{r_i\xi_{i,t}}{p_i}(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f(x_i)\right)^2}{\frac{2}{n}\sum_{i=1}^n \frac{|r_i|\sigma_i}{p_i}\left\|f(x_i)\right\|^2}.$$

Ignoring the $1/2$ scaling (which does not affect the argmax), and recognizing that $f_t^\star$ is scale-invariant, we obtain the following constrained optimization problem for the base learner:

$$\max_{f\in\mathcal{F}} \left|\frac{1}{n}\sum_{i=1}^n \frac{r_i\xi_{i,t}}{p_i}(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f(x_i)\right| \quad \text{s.t.} \quad \frac{1}{n}\sum_{i=1}^n \frac{|r_i|\sigma_i}{p_i}\left\|f(x_i)\right\|^2 = Z.$$

This becomes line 3 of Algorithm 2, and Equation 19 becomes line 4.

# C EXCESS RISK ANALYSIS

This appendix provides the proof of our excess empirical risk bound (Theorem 1), and then shows how it can be combined with concentration and uniform convergence to bound the excess population risk.

## C.1 Proof of Theorem 1

When we substitute the algorithm's constraint that $\frac{1}{n}\sum_{i=1}^n \frac{|r_i|}{p_i}\left\|f(x_i)\right\|^2 = Z$ into Equation 15, we obtain

$$\begin{aligned}\hat{L}(\pi_t, S) &\leq \hat{L}(\pi_{t-1}, S) - \frac{1}{Z}\left(\frac{1}{n}\sum_{i=1}^n \frac{r_i}{p_i}\pi_{t-1}(a_i \mid x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f_t(x_i)\right)^2 \\ &= \hat{L}(\pi_{t-1}, S) - \frac{Z}{4}\alpha_t^2,\end{aligned}$$

in which we reduce the righthand expression using Equation 14. This bound is recursive, depending on the empirical risk of the previous ensemble policy, $\pi_{t-1}$. The base case is $\hat{L}(\pi_0, S)$. Unraveling the recursion from round $T$, we get

$$\hat{L}(\pi_T, S) \leq \hat{L}(\pi_{T-1}, S) - \frac{Z}{4}\alpha_T^2 \leq \hat{L}(\pi_0, S) - \frac{Z}{4}\sum_{t=1}^T \alpha_t^2.$$

Subtracting $\hat{L}^\star$ from both sides of the inequality, we get

$$\hat{L}(\pi_T, S) - \hat{L}^\star \leq \hat{L}(\pi_0, S) - \hat{L}^\star - \frac{Z}{4}\sum_{t=1}^T \alpha_t^2 = \Delta_0 - \frac{Z}{4}\sum_{t=1}^T \alpha_t^2. \tag{20}$$

Note that $\Delta_0$ is nonnegative, by definition of $\hat{L}^\star$. Therefore, using the identity $c(1-z) \leq ce^{-z}$, for all $c \in \mathbb{R}_+$ and $z \in \mathbb{R}$, we have that

$$(20) = \Delta_0\left(1 - \frac{Z}{4\Delta_0}\sum_{t=1}^T \alpha_t^2\right) \leq \Delta_0 \exp\left(-\frac{Z}{4\Delta_0}\sum_{t=1}^T \alpha_t^2\right),$$

which completes the proof.

## C.2 Excess Population Risk Bound

We now explain how to relate Theorem 1 to an upper bound on BOPL's excess risk relative to an optimal policy. Let $\pi^\star \in \arg\min_\pi L(\pi)$ denote an optimal policy (i.e., risk minimizer), and let $L^\star \triangleq L(\pi^\star)$ denote its corresponding risk. Recall that $\hat{\pi}^\star \in \arg\min_\pi \hat{L}(\pi, S)$ is an *empirically* optimal policy (i.e., empirical risk minimizer) for a given dataset, $S$, and $\hat{L}^\star \triangleq \hat{L}(\hat{\pi}^\star)$ is its corresponding empirical risk. Using these definitions, the excess risk can be expressed as $L(\pi_T) - L^\star$, and the excess empirical risk is $\hat{L}(\pi_T, S) - \hat{L}^\star$.

Via simple arithmetic, we can expand the excess risk into several terms:

$$\begin{aligned}\underbrace{L(\pi_T) - L^\star}_{\text{excess risk}} &= L(\pi_T) - \hat{L}(\pi_T, S) + \hat{L}(\pi_T, S) - \hat{L}^\star + \hat{L}^\star - \hat{L}(\pi^\star, S) + \hat{L}(\pi^\star, S) - L^\star \\ &\leq \underbrace{L(\pi_T) - \hat{L}(\pi_T, S)}_{\text{generalization error}} + \underbrace{\hat{L}(\pi_T, S) - \hat{L}^\star}_{\text{excess empirical risk}} + \underbrace{\hat{L}(\pi^\star, S) - L^\star}_{\text{estimation error}}.\end{aligned}$$

Starting on the right, the difference $\hat{L}(\pi^\star, S) - L^\star$ captures our ability to *estimate* the risk of a policy (in this case, the optimal policy) using a finite sample of data. Similarly, on the left, $L(\pi_T) - \hat{L}(\pi_T, S)$ captures the learning algorithm's ability to *generalize* from finite data, as quantified by the difference of the risk and empirical risk—the latter of which is being optimized. Finally, the middle difference is the excess empirical risk—which is what we upper-bound in Theorem 1.

Thus, to upper-bound the excess risk, we must bound the estimation and generalization errors. Estimation and generalization have been studied extensively in statistical learning theory, so we have many tools at our disposal to upper-bound those error terms. In the following, we provide an example bound—which is by no means optimal, but is merely meant to illustrate how to apply existing theory to complete the picture.

For simplicity, we will assume that rewards are bounded in $[-1, 1]$, and that the logged propensities are uniformly lower-bounded by some positive constant, $\pi_{\mathrm{L}}( \cdot \mid \cdot ) \geq \tau > 0$. With these assumptions, we have that the random variable $\varphi(\pi, x, a, p, r) \triangleq -\frac{r}{p}\pi(a \,|\, x)$ is almost-surely bounded in $[-\tau^{-1}, \tau^{-1}]$. Note that $L(\pi) = \mathbb{E}[\varphi(\pi, x, a, p, r)]$ and $\hat{L}(\pi, S) \triangleq \frac{1}{n}\sum_{i=1}^{n} \varphi(\pi_i, x_i, a_i, p_i, r_i)$.

Accordingly, since $\hat{L}(\pi, S)$ is just an average of bounded, i.i.d. random variables, we can use any applicable concentration inequality to upper-bound the estimation error. For example, Hoeffding's inequality (Hoeffding, 1963) yields

$$\Pr_{S}\left\{\hat{L}(\pi^\star, S) - L(\pi^\star) \geq \epsilon\right\} \leq \exp\left(-\frac{n\tau^2\epsilon^2}{2}\right);$$

so, with probability at least $1 - \delta/2$ over draws of $S$,

$$\hat{L}(\pi^\star, S) - L(\pi^\star) \leq \frac{1}{\tau}\sqrt{\frac{2}{n}\ln\frac{2}{\delta}}. \tag{21}$$

There are many tools available to bound the generalization error of a learning algorithm. We will approach it from the perspective of *uniform convergence*—that is, we will show that, with high probability, the generalization error of any $\pi_T \in \Pi_T$ is vanishing in $n$. To do so, we will leverage a standard Rademacher complexity-based generalization bound. For a generic real-valued function class, $\mathcal{G}$, its *Rademacher complexity* is

$$\mathfrak{R}_n(\mathcal{G}) \triangleq \mathbb{E}\left[\sup_{g \in \mathcal{G}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i g(z_i)\right],$$

where $z_1, \ldots, z_n$ are i.i.d. draws from an arbitrary distribution, and $\sigma_1, \ldots, \sigma_n$ are independent *Rademacher variables*, which are uniformly distributed over $\{\pm 1\}$. Let

$$\Phi \triangleq \varphi \circ \Pi_T = \{(x, a, p, r) \mapsto \varphi(\pi_T, x, a, p, r) : \pi_T \in \Pi_T\}$$

denote the composition of $\varphi$ and $\Pi_T$, where $\phi \in \Phi$ is a member of the class. Leveraging (Mohri et al., 2012, Theorem 3.3), we have that, for any $\delta' \in (0, 1)$ (to be defined later), with probability at least $1 - \delta'$,

$$\begin{aligned} L(\pi_T) - \hat{L}(\pi_T, S) &\leq \sup_{\pi_T \in \Pi_T} L(\pi_T) - \hat{L}(\pi_T, S) \\ &= \sup_{\phi \in \Phi} \mathbb{E}[\phi(x, a, p, r)] - \frac{1}{n}\sum_{i=1}^{n} \phi(x_i, a_i, p_i, r_i) \\ &\leq 2\,\mathfrak{R}_n(\Phi) + \frac{1}{\tau}\sqrt{\frac{2}{n}\ln\frac{1}{\delta'}}. \end{aligned} \tag{22}$$

Note that we modified the bound to account for the range of $\varphi$. Taking $\delta' = \delta/2$, we have that Equations 21 and 22 hold with probability at least $1 - \delta$; and when combined with Equation 6, we upper-bound the excess risk with high probability.

However, we can make the dependence on $\mathcal{F}$ more explicit by focusing on the Rademacher term. First, we note that $\varphi$ is $\tau^{-1}$-Lipschitz with respect to the 2-norm of the ensemble predictor output—that is, for any $F_T, F'_T \in \mathcal{F}_T$ (with associated policies, $\pi_T, \pi'_T$), $x \in \mathcal{X}$, $a \in \mathcal{A}$, $p \in [\tau, 1]$ and $r \in [-1, 1]$,

$$|\varphi(\pi, x, a, p, r) - \varphi(\pi', x, a, p, r)| \leq \frac{1}{\tau} \left\| F_T(x) - F'_T(x) \right\|_2 .$$

This is readily verified by noting that $|r/p| \leq \tau^{-1}$, and using the fact that the softmax function (for $\beta = 1$) is 1-Lipschitz (Gao & Pavel, 2017, Proposition 4). Therefore, using a vector-valued extension of Talagrand's contraction lemma (Maurer, 2016, Corollary 1), we have that

$$\begin{aligned}
\mathfrak{R}_n(\Phi) &= \mathbb{E}\left[\sup_{\phi \in \Phi} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \phi(x_i, a_i, p_i, r_i)\right] \\
&= \mathbb{E}\left[\sup_{\pi_T \in \Pi_T} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \varphi(\pi_T, x_i, a_i, p_i, r_i)\right] \\
&\leq \frac{\sqrt{2}}{\tau} \underbrace{\mathbb{E}\left[\sup_{F_T \in \mathcal{F}_T} \frac{1}{n} \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} \sigma_{i,a} F_T(x_i, a)\right]}_{\triangleq \mathfrak{R}_n(\mathcal{F}_T)} .
\end{aligned}$$

We are left with the (vector-valued) Rademacher complexity of the class of ensemble predictors, $\mathcal{F}_T$. To get to the complexity of $\mathcal{F}$, we will appeal to well known results for convex combinations of hypotheses.

Unfortunately, softmax ensemble policies are not convex combinations of predictors. Nonetheless, we can transform $\Pi_T$ into a class of convex ensembles, so that we can leverage existing Rademacher bounds. First, we will assume that $\mathcal{F}$ is symmetric. By implication, we can assume that every ensemble weight, $\alpha_t$, is nonnegative—since any pair, $(f, \alpha) : f \in \mathcal{F},\ \alpha < 0$, has a corresponding $f' \in \mathcal{F}$ such that $\alpha f = (-\alpha) f'$. Then, we will temporarily assume that the ensemble weights have 1-norm bounded by some constant, $B > 0$, which will allow us to normalize the weights and thereby obtain the Rademacher complexity of convex ensembles, scaled by $B$. Finally, we construct a covering of all $B$, which allows us to obtain high-probability bounds that hold for all $B$ simultaneously.

Let

$$\mathcal{F}_T^B \triangleq \left\{ (x, a) \mapsto F_T(x, a) : \forall t,\ f_t \in \mathcal{F},\ \alpha_t \in \mathbb{R}_+;\ \sum_{t=1}^{T} \alpha_t \leq B \right\}$$

denote the set of ensembles for $\mathcal{F}$ with nonnegative weights, whose sum is upper-bounded by $B$. Further, let

$$\tilde{\mathcal{F}}_T \triangleq \left\{ (x, a) \mapsto F_T(x, a) : \forall t,\ f_t \in \mathcal{F},\ \alpha_t \in [0, 1];\ \sum_{t=1}^{T} \alpha_t = 1 \right\}$$

denote the set of convex ensembles for $\mathcal{F}$. For simplicity, we will write $\left\| \boldsymbol{\alpha} \right\|_1 \triangleq \sum_{t=1}^{T} |\alpha_t|$, where $\boldsymbol{\alpha} \triangleq (\alpha_1, \ldots, \alpha_T)$ denotes a vector of ensemble weights, whose length, $T$, should be clear from context. (The absolute value is unnecessary when all weights are nonnegative, but we include it for correctness.) Observe that, for any $F_T \in \mathcal{F}_T$, with weights $\boldsymbol{\alpha}$, there exists a $\tilde{F}_T \in \tilde{\mathcal{F}}_T$ such that $\tilde{F}_T(\,\cdot\,,\,\cdot\,) = \frac{F_T(\,\cdot\,,\,\cdot\,)}{\left\| \boldsymbol{\alpha} \right\|_1}$ (when $\mathcal{F}$ is assumed to be symmetric). Thus,

$$\begin{aligned}
\mathfrak{R}_n(\mathcal{F}_T^B) &= \mathbb{E}\left[\sup_{F_T \in \mathcal{F}_T} \frac{\left\| \boldsymbol{\alpha} \right\|_1}{n} \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} \sigma_{i,a} \frac{F_T(x_i, a)}{\left\| \boldsymbol{\alpha} \right\|_1}\right] \\
&\leq \mathbb{E}\left[\sup_{F_T \in \mathcal{F}_T} \frac{B}{n} \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} \sigma_{i,a} \frac{F_T(x_i, a)}{\left\| \boldsymbol{\alpha} \right\|_1}\right] \\
&= B\,\mathbb{E}\left[\sup_{\tilde{F}_T \in \tilde{\mathcal{F}}_T} \frac{1}{n} \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} \sigma_{i,a} \tilde{F}_T(x_i, a)\right] \\
&= B\,\mathfrak{R}_n(\tilde{\mathcal{F}}_T).
\end{aligned}$$

Having reduced $\mathfrak{R}_n(\mathcal{F}_T^B)$ to a function of $\mathfrak{R}_n(\tilde{\mathcal{F}}_T)$, we can leverage a classical result (Bartlett & Mendelson, 2003, Theorem 12), which states (among other things): (1) for classes $\mathcal{G}$ and $\mathcal{H}$, if $\mathcal{G} \subseteq \mathcal{H}$, then $\mathfrak{R}_n(\mathcal{G}) \leq \mathfrak{R}_n(\mathcal{H})$; (2) for the *convex hull* of $\mathcal{H}$, denoted $\operatorname{conv}(\mathcal{H})$, we have $\mathfrak{R}_n(\operatorname{conv}(\mathcal{H})) = \mathfrak{R}_n(\mathcal{H})$. Therefore, since $\tilde{\mathcal{F}}_T \subseteq \operatorname{conv}(\mathcal{F})$, we have that $\mathfrak{R}_n(\tilde{\mathcal{F}}_T) \leq \mathfrak{R}_n(\operatorname{conv}(\mathcal{F})) = \mathfrak{R}_n(\mathcal{F})$; and thus, $\mathfrak{R}_n(\mathcal{F}_T^B) \leq B\,\mathfrak{R}_n(\mathcal{F})$.

Finally, having related $\mathfrak{R}_n(\Phi)$ to $\mathfrak{R}_n(\mathcal{F})$ when the ensemble weights are constrained to a specific sum, $B$, we want Equation 22 to hold, with probability at least $1 - \delta/2$, for *all* $B$ simultaneously. For $j = 0, 1, 2, \ldots$, let $B_j \triangleq 2^j$ and $\delta_j \triangleq \frac{\delta}{4} B_j^{-1}$. Observe that $\delta_j$ forms a geometric series, and $\sum_{j=0}^{\infty} \delta_j = \frac{\delta}{4} \sum_{j=0}^{\infty} 2^{-j} = \delta/2$. Thus, assigning probability $\delta_j$ to each $B_j$, we have that, with probability at least $1 - \delta/2$, for all $j$ simultaneously,

$$\begin{aligned}\sup_{\pi_T \in \Pi_T^{B_j}} L(\pi_T) - \hat{L}(\pi_T, S) &\leq 2\,\mathfrak{R}_n(\Phi^{B_j}) + \frac{1}{\tau}\sqrt{\frac{2}{n}\ln\frac{1}{\delta_j}} \\ &\leq \frac{\sqrt{8}}{\tau}\,\mathfrak{R}_n(\mathcal{F}_T^{B_j}) + \frac{1}{\tau}\sqrt{\frac{2}{n}\ln\frac{1}{\delta_j}} \\ &\leq \frac{\sqrt{8}}{\tau} B_j \mathfrak{R}_n(\mathcal{F}) + \frac{1}{\tau}\sqrt{\frac{2}{n}\ln\frac{1}{\delta_j}},\end{aligned}$$

where we use superscript $B_j$ in $\Pi_T^{B_j}$ and $\Phi^{B_j}$ to indicate that the underlying class of ensembles has weights bounded accordingly. What remains is to pick a value of $j$ for the learned ensemble policy, $\pi_T$. Taking

$$j^\star \triangleq \left\lceil (\ln 2)^{-1} \ln \max\{\|\boldsymbol{\alpha}\|_1, 1\} \right\rceil,$$

we have that

$$B_{j^\star} = 2^{\left\lceil (\ln 2)^{-1} \ln \max\{\|\boldsymbol{\alpha}\|_1, 1\} \right\rceil} \geq 2^{(\ln 2)^{-1} \ln \max\{\|\boldsymbol{\alpha}\|_1, 1\}} = \max\{\|\boldsymbol{\alpha}\|_1, 1\} \geq \|\boldsymbol{\alpha}\|_1 ;$$

meaning, the learned ensemble, $F_T$, is contained in the class $\mathcal{F}_T^{B_{j^\star}}$, so the bound for $j^\star$ is valid for $\pi_T$. Further,

$$B_{j^\star} \leq 2^{(\ln 2)^{-1} \ln \max\{\|\boldsymbol{\alpha}\|_1, 1\} + 1} = 2 \max\{\|\boldsymbol{\alpha}\|_1, 1\},$$

and

$$\delta_{j^\star}^{-1} = \frac{4}{\delta} B_{j^\star} \leq \frac{8}{\delta} \max\{\|\boldsymbol{\alpha}\|_1, 1\}.$$

Putting it all together, we have that, with probability at least $1 - \delta/2$,

$$L(\pi_T) - \hat{L}(\pi_T, S) \leq \frac{\sqrt{32}}{\tau} \max\{\|\boldsymbol{\alpha}\|_1, 1\}\,\mathfrak{R}_n(\mathcal{F}) + \frac{1}{\tau}\sqrt{\frac{2}{n}\ln\left(\frac{8}{\delta}\max\{\|\boldsymbol{\alpha}\|_1, 1\}\right)}. \tag{23}$$

Combining Equations 6, 21 and 23, we obtain a full characterization of the excess risk that holds with probability at least $1 - \delta$:

$$\begin{aligned}\underbrace{L(\pi_T) - L^\star}_{\text{excess risk}} \quad &\leq \quad \underbrace{\Delta_0 \exp\left(-\frac{Z}{4\Delta_0}\sum_{t=1}^{T}\alpha_t^2\right)}_{\text{excess empirical risk}} \\ &\quad + \underbrace{\frac{\sqrt{32}}{\tau}\max\{\|\boldsymbol{\alpha}\|_1, 1\}\,\mathfrak{R}_n(\mathcal{F}) + \frac{1}{\tau}\sqrt{\frac{2}{n}\ln\left(\frac{8}{\delta}\max\{\|\boldsymbol{\alpha}\|_1, 1\}\right)}}_{\text{generalization error}} \\ &\quad + \underbrace{\frac{1}{\tau}\sqrt{\frac{2}{n}\ln\frac{2}{\delta}}}_{\text{est. error}}.\end{aligned}$$

Note that the bound is stated in terms of the Rademacher complexity of the predictor class, $\mathcal{F}$. For many useful hypothesis classes (such as decision stumps and trees), the Rademacher complexity vanishes at rate $\mathrm{O}(n^{-1/2})$. In such cases, if the excess empirical risk is small, then the excess risk also vanishes at rate $\mathrm{O}(n^{-1/2})$.

*Remark* 2. Since $\pi^\star$ and $\hat{\pi}^\star$ are defined via *unconstrained* minimizations over all valid policies (not just those contained in $\Pi_T$), there could be an implicit gap between the minimum attainable risk and the risk of the best ensemble policy, $\pi_T^\star \in \arg\min_{\pi_T \in \Pi_T} L(\pi_T)$. We can make this gap explicit by modifying our excess risk decomposition:

$$\underbrace{L(\pi_T) - L^\star}_{\text{excess risk}} \le \underbrace{L(\pi_T) - \hat{L}(\pi_T, S)}_{\text{generalization error}} + \underbrace{\hat{L}(\pi_T, S) - \hat{L}^\star}_{\text{excess empirical risk}} + \underbrace{\hat{L}(\pi_T^\star, S) - L(\pi_T^\star)}_{\text{estimation error}} + \underbrace{L(\pi_T^\star) - L^\star}_{\text{approximation error}} .$$

The new term—the so-called *approximation error* (Bottou & Bousquet, 2007)—measures how well $\Pi_T$ fits the distribution. Unfortunately, the approximation error is unknowable in all but trivial cases, so it is typically assumed to be some small constant. The rest of the bound, when instantiated with the above analysis, would work out the same. Thus, in this case, there does not appear to be any advantage to making the approximation error explicit. △

# D BASE LEARNING REDUCTIONS

In this appendix, we derive base learners for two classes of predictors: (nonlinear) real-valued functions and binary classifiers. In both cases, we assume that the class, $\mathcal{F}$, is *symmetric*; meaning, for every $f \in \mathcal{F}$, its negation, $-f$, is also in $\mathcal{F}$.

For brevity, we focus on Algorithm 1, though it is straightforward to adapt the base learners for Algorithm 2.

## D.1 Boosting via Regression

Assume that $\mathcal{F}$ is a symmetric class of real-valued functions, $\mathcal{F} \subseteq \{\mathcal{X} \times \mathcal{A} \to \mathbb{R}\}$. Since an ensemble, $F_T$, is a linear combination of predictors, it is important that $\mathcal{F}$ is a nonlinear function class (such as regression trees), so that $F_T$ can have greater expressive power than its constituents. (If the predictors were linear functions, then $F_T$ would still be a linear function of its input.)

Recall the base learning objective in Equation 17. Since we assume that $\mathcal{F}$ is symmetric, we can omit the absolute value; if some $f \in \mathcal{F}$ is a minimizer of the expression inside the absolute value, then its negation, $-f$, is also in $\mathcal{F}$. We then convert the constrained optimization problem to the following unconstrained one via Lagrangian relaxation:

$$(17) \;=\; \max_{f \in \mathcal{F}} \min_{\lambda \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f(x_i) - \lambda \left( \frac{1}{n} \sum_{i=1}^{n} \frac{|r_i|}{p_i} \left\| f(x_i) \right\|^2 - Z \right).$$

For every $Z$, there exists a $\lambda$ that is optimal. The converse of this statement is that, for every $\lambda$, one can construct a $Z$ for which $\lambda$ is optimal. Since $Z$ is arbitrary, we can choose any $\lambda$ for the optimization. Without loss of generality, we take $\lambda = 1/2$, which results in

$$\begin{aligned}
&\arg\max_{f \in \mathcal{F}} \sum_{i=1}^{n} \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f(x_i) - \frac{|r_i|}{2p_i} \left\| f(x_i) \right\|^2 \\
&= \arg\max_{f \in \mathcal{F}} \sum_{i=1}^{n} \frac{|r_i|}{p_i} \Big( \operatorname{sgn}(r_i) \pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f(x_i) - \frac{1}{2} \left\| f(x_i) \right\|^2 \Big) \\
&= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \frac{|r_i|}{p_i} \left\| \operatorname{sgn}(r_i) \pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i)) - f(x_i) \right\|^2 .
\end{aligned}$$

This is a weighted least-squares regression, which can be solved by a variety of off-the-shelf tools.

The resulting boosting algorithm is given in Algorithm 3. Line 3 uses a given subroutine for solving weighted least-squares regression problems with the class $\mathcal{F}$.

Since the regression base learner does not explicitly enforce the constraint in Equation 17, that $\frac{1}{n} \sum_{i=1}^{n} \frac{|r_i|}{p_i} \left\| f_t(x_i) \right\|^2 = Z$ for a given $Z > 0$, one cannot immediately apply Theorem 1 to Algorithm 3. However, recall that $\lambda$ implicitly ensures that $\frac{1}{n} \sum_{i=1}^{n} \frac{|r_i|}{p_i} \left\| f_t(x_i) \right\|^2 = Z'$ for some $Z' > 0$; and if the base learning problem can be solved for $Z'$, then it can be solved for any $Z$ by simply rescaling the predictor, $f_t \leftarrow f_t \sqrt{Z/Z'}$, which does not affect the predictor's optimality with respect to Equation 16 if it is rescaled prior to computing $\alpha_t$. Thus, given the output of the base learner (line 3), we compute $Z'$, then perform the rescaling prior to line 4, thereby ensuring that Theorem 1 holds for $Z$. All that being said, it is important to remember that this modification is not strictly necessary for the algorithm to work; only for Theorem 1 to hold.

**Algorithm 3** Boosted Off-Policy Learning via Regression

```
Input: symmetric, real-valued class, $\mathcal{F}$; solver for weighted least-squares; rounds, $T \geq 1$
1: $F_0 \leftarrow 0$
2: for $t = 1, \ldots, T$ do
3:     $f_t \leftarrow \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} w_i (y_{i,a} - f(x_i, a))^2$        ▷ weighted least-squares
           with  $w_i \leftarrow \frac{|r_i|}{p_i}$
           and  $y_{i,a} \leftarrow \operatorname{sgn}(r_i) \pi_{t-1}(a_i \,|\, x_i)(\mathbb{1}\{a = a_i\} - \pi_{t-1}(a \,|\, x_i))$
4:     $\alpha_t \leftarrow \frac{2 \sum_{i=1}^{n} \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f_t(x_i)}{\sum_{i=1}^{n} \frac{|r_i|}{p_i} \|f_t(x_i)\|^2}$
5:     $F_t \leftarrow F_{t-1} + \alpha_t f_t$
```

### D.2 Boosting via Binary Classification

The following reduction to binary classification is inspired by (Schapire & Freund, 2012, Section 7.4.3). Assume that $\mathcal{F}$ a symmetric class of binary classifiers, $\mathcal{F} \subseteq \{\mathcal{X} \times \mathcal{A} \to \{\pm 1\}\}$, such as decision stumps. For every example, $i \in \{1, \ldots, n\}$, and action, $a \in \mathcal{A}$, we define a nonnegative weight,

$$w_{i,a} \triangleq \left| \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbb{1}\{a = a_i\} - \pi_{t-1}(a \,|\, x_i)) \right|,$$

and a $\{\pm 1\}$-valued pseudo-label,

$$y_{i,a} \triangleq \operatorname{sgn}(r_i)(2\,\mathbb{1}\{a = a_i\} - 1) = \operatorname{sgn}\left( \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbb{1}\{a = a_i\} - \pi_{t-1}(a \,|\, x_i)) \right).$$

(These variables are local to the current round, $t$, but we ignore this to simplify notation.) Using the righthand equivalence, we have that

$$w_{i,a} y_{i,a} = \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbb{1}\{a = a_i\} - \pi_{t-1}(a \,|\, x_i)).$$

We also have that

$$\mathbb{1}\{y_{i,a} \neq f(x_i, a)\} = \frac{1}{2}(1 - y_{i,a} f(x_i, a)).$$

Therefore, if we minimize the weighted classification error, we end up with the following equivalence:

$$\begin{aligned}
& \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} w_{i,a} \mathbb{1}\{y_{i,a} \neq f(x_i, a)\} \\
= \; & \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} \frac{w_{i,a}}{2} (1 - y_{i,a} f(x_i, a)) \\
= \; & \arg\max_{f \in \mathcal{F}} \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} w_{i,a} y_{i,a} f(x_i, a) \\
= \; & \arg\max_{f \in \mathcal{F}} \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbb{1}\{a = a_i\} - \pi_{t-1}(a \,|\, x_i)) f(x_i, a) \\
= \; & \arg\max_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^{n} \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f(x_i) \right|.
\end{aligned}$$

The last equality uses the symmetry of $\mathcal{F}$ to introduce the absolute value. The base learning scale constraint, $\frac{1}{n} \sum_{i=1}^{n} \frac{|r_i|}{p_i} \|f(x_i)\|^2 = Z$, is automatically satisfied for $Z = \frac{|\mathcal{A}|}{n} \sum_{i=1}^{n} \frac{|r_i|}{p_i}$ by the fact that $\|f(x_i)\|^2 = |\mathcal{A}|$. If there is at least one nonzero reward in the dataset, then $Z > 0$. Thus, minimizing the weighted classification error is equivalent to solving the base learning objective.

The resulting algorithm is given in Algorithm 4. The optimization problem in line 3—minimizing the weighted classification error—is generally NP-hard (Ben-David et al., 2003). However, it is important to remember that the base learner need not fully optimize its learning objective for boosting to be successful; indeed, any predictor for which $\alpha_t \neq 0$ reduces the empirical risk upper bound. In fact, for binary classification, we can relate this condition to a weak learning property.

Let

$$\tilde{w}_{i,a} \triangleq \frac{w_{i,a}}{\sum_{i=1}^{n} \sum_{a \in \mathcal{A}} w_{i,a}},$$

denote a *normalized* weight, which defines an empirical distribution over the data passed to the base learner. Let

$$\epsilon_t(f) \triangleq \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} \tilde{w}_{i,a} \mathbb{1}\{y_{i,a} \neq f(x_i, a)\}$$

denote the *error rate* under this distribution, and note that it is proportional to the weighted classification error. Recalling the definition of $\alpha_t$ from Equation 14, note that the numerator determines the sign of the expression, and

$$\begin{aligned}
\frac{2}{n} \sum_{i=1}^{n} \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f_t(x_i) &= \frac{2}{n} \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} w_{i,a} y_{i,a} f_t(x_i, a) \\
&\propto \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} \tilde{w}_{i,a} y_{i,a} f_t(x_i, a) \\
&= \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} \tilde{w}_{i,a} (1 - 2\,\mathbb{1}\{y_{i,a} \neq f_t(x_i, a)\}) \\
&= 1 - 2\epsilon_t(f_t).
\end{aligned}$$

Thus, $\alpha_t$ is positive when $\epsilon_t(f_t) < 1/2$, negative when $\epsilon_t(f_t) > 1/2$, and zero when $\epsilon_t(f_t) = 1/2$. Recall that boosting can proceed as long as $\alpha_t \neq 0$; meaning, as long as the base learner can produce a classifier that performs better than random guessing under the weighted distribution. This is the weak learning condition for binary classification base learners. If the base learner *always* satisfies this condition, and there exists a constant, $\gamma \in [0, 1/2)$, such that $\epsilon_t(f_t) < \gamma$ at every round, then (appealing to Theorem 1) the excess empirical risk decays exponentially fast.

**Algorithm 4** Boosted Off-Policy Learning via Binary Classification

```
Input: symmetric, $\{\pm 1\}$-valued class, $\mathcal{F}$; learning algorithm for weighted binary classification; rounds, $T \geq 1$
1: $F_0 \leftarrow 0$
2: for $t = 1, \ldots, T$ do
3:     $f_t \leftarrow \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} w_{i,a} \mathbb{1}\{y_{i,a} \neq f(x_i, a)\}$        ▷ weighted binary classification
           with $w_{i,a} \leftarrow \left| \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbb{1}\{a = a_i\} - \pi_{t-1}(a \,|\, x_i)) \right|$
           and $y_{i,a} \leftarrow \operatorname{sgn}(r_i)(2\,\mathbb{1}\{a = a_i\} - 1)$
4:     $\alpha_t \leftarrow \frac{2 \sum_{i=1}^{n} \frac{r_i}{p_i} \pi_{t-1}(a_i \,|\, x_i)(\mathbf{a}_i - \pi_{t-1}(x_i))^\top f_t(x_i)}{\sum_{i=1}^{n} \frac{|r_i|}{p_i} \|f_t(x_i)\|^2}$
5:     $F_t \leftarrow F_{t-1} + \alpha_t f_t$
```

# E REGULARIZATION

Recall that our original goal is to find a policy with low *expected* risk. Though the IPS estimator is unbiased, optimizing the empirical risk can sometimes lead to overfitting—e.g., if the class of policies is very rich. Accordingly, it is common to add some form of regularization to the learning objective, to penalize model complexity. In our boosting framework, we can assume an ensemble regularizer that decomposes as a sum of base regularizers, $R(F_T) = \sum_{t=1}^{T} R_t(f_t)$. Then, regularization can be applied at the base learner by adding $R_t(f)$ to line 3 of Algorithms 1 and 2.

Recognizing that the ensemble's complexity is partially determined by the magnitudes of the ensemble weights, we could also choose to regularize the ensemble weights (Duchi & Singer, 2009). Note that if we penalize the squared norm of the ensemble weights, adding $\lambda \sum_{t=1}^{T} \alpha_t^2$ to the learning objective (Equation 4), then the closed-form expression for the ensemble weights (using our smoothness-based derivation) would be inversely proportional to the regularization parameter, $\lambda$. From the perspective of functional gradient descent, $\lambda > 0$ has the effect of *shrinking* the learning rate, with larger values leading to smaller step sizes.

Table 3: Details of the datasets used in our experiments. *Task-types* "MC" and "ML" denote multiclass and multilabel, respectively. *Logging Training* refers to the number of examples used to train the logging policy, and *Target Training* is the number of simulated bandit feedback examples used to train the target policy.

| **Dataset** | **Task-type** | **Features** | **Classes** | **Logging Training** | **Target Training** | **Validation** | **Testing** |
|---|---|---|---|---|---|---|---|
| **Covertype** | MC | 54 | 7 | 46,481 | 418,329 | 58,101 | 58,101 |
| **Fashion-MNIST** | MC | 784 | 10 | 5,400 | 48,600 | 6,000 | 10,000 |
| **Scene** | ML | 294 | 6 | 154 | 1,387 | 385 | 481 |
| **TMC2007-500** | ML | 500 | 22 | 2,316 | 20,846 | 2,574 | 2,860 |

# F DATASET DETAILS

Covertype is a multiclass classification dataset, consisting of 581,012 records with 54 binary, ordinal and real-valued features. The task is to predict one of 7 classes of ground cover for each record. Fashion-MNIST is a multiclass image classification dataset, consisting of 70,000 grayscale images from 10 categories of apparel and accessories. We extract features from each image by flattening the $(28 \times 28)$-pixel grid to a 784-dimensional vector. The Scene dataset is a multilabel image classification dataset, consisting of 2,407 records with 294 numeric features, derived from the images' spatial color moments in LUV space. The task is to determine which of 6 settings are depicted in the image. Lastly, TMC2007-500 is a multilabel document classification dataset, consisting of 28,596 airplane failure reports. Each record consists of 500 binary features, indicating the presence or absence of the 500 most frequent words in the corpus. The task is to determine which of 22 types of problems are described in the document.

## F.1 Feature Scaling

As feature ranges are important to deep learning (but not boosted tree ensembles), for baselines DRR and BanditNet, we performed some feature scaling. In Covertype, we standardized the ordinal and real-valued features; and in Fashion-MNIST, we normalized the pixel intensities to $[0, 1]$. The Scene and TMC2007-500 datasets' features are already constrained to $[0, 1]$, so no additional feature scaling is necessary.

## F.2 Data Splits

Of the four datasets, only Fashion-MNIST has a standard training/testing split (60,000 training; 10,000 testing), which we preserve. We withold a random 10% of the training data for validation. For Covertype, we use a random 80%/10%/10% training/validation/testing split. For Scene, we first perform a random 80%/20% split of data into training and testing partitions, then another 80%/20% split of the training data into training and validation partitions, ultimately resulting in a 64%/16%/20% training/validation/testing split. We perform the same procedure for TMC2007-500 using 90%/10% splits, resulting in a 81%/9%/10% training/validation/testing split. The final dataset sizes are summarized in Table 3.

Note that all of these splits are prior to the supervised-to-bandit conversion (described below), so we have full information for evaluating performance metrics.

## F.3 Supervised-to-Bandit Conversion

To simulate logged bandit feedback from supervised datasets, we use the procedure proposed by Beygelzimer & Langford (2009), which has become standard. We start by randomly sampling 10% of the training examples (without replacement) to train a softmax logging policy using supervised learning—in this case, multinomial logistic regression. To avoid overfitting such a small dataset—which could cause the logging policy to become overly confident in certain contexts—we turn up the regularization and employ early stopping. We then use the logging policy to sample a label (i.e., action) for each remaining training example. In some cases, we enforce a minimum probability for each action by mixing the softmax probabilities with $\epsilon$-greedy exploration. For each sampled label, we record its propensity (under the logging policy) and corresponding reward.

We repeat this procedure 10 times, using 10 random splits of the training data, to generate 10 datasets of logged contexts, actions, propensities and rewards.

### F.4 Task Reward Structure

For the multilabel datasets, Scene and TMC2007, we assign a full reward of one if the selected action matches one of the true labels, and zero otherwise. For the multiclass datasets, Fashion-MNIST and Covertype, we assign full reward if the true label was selected, but give partial credit if the selected action belongs to a "near miss" category. For instance, in Fashion-MNIST, selecting *T-shirt* instead of *shirt* yields a reward of $1/4$. And in Covertype, partial credit is given when the selected action predicts a class of vegetation from the same genus as the actual type; e.g., predicting *aspen* instead of *cottonwood* tree yields a reward of $1/4$.

We define the partial credit class-groupings for Fashion-MNIST as follows:

- $\textit{Outerwear} = \{\textit{coat}, \textit{pullover}\}$
- $\textit{Shirts} = \{\textit{T-shirt/top}, \textit{shirt}\}$
- $\textit{Footwear} = \{\textit{sandal}, \textit{sneaker}, \textit{ankle-boot}\}$

The rest of the classes ('*trouser*', '*dress*', and '*bag*'), are left as singleton groups with no possibility for partial credit.

Our partial credit groupings for Covertype are:

- $\textit{Firs} = \{\textit{Spruce/Fir}, \textit{Douglas-fir}\}$
- $\textit{Pines} = \{\textit{Lodgepole}, \textit{Ponderosa}\}$
- $\textit{Populus} = \{\textit{Cottonwood/Willow}, \textit{Aspen}\}$

The remaining class, '*Krummholz*', is placed in a singleton group.

### F.5 Representation of Contextualized Actions

For all datasets and methods (except for the logging policy), we construct feature vectors for actions by augmenting the original features in the dataset with an encoding of each action. We accomplish this by concatenating the original features, $\mathbf{x}$ with a one-hot vector, $\mathbf{a}$, identifying each action, $a$; that is $(\mathbf{x}, a) \mapsto [\mathbf{x}; \mathbf{a}]$, for all $a \in \mathcal{A}$. This results in $|\mathcal{A}|$ feature vectors for each example in the dataset. A policy therefore scores all actions (using its predictor or ensemble) and then uses the scores to select an action (either by softmax sampling or argmax).

## G ALGORITHM IMPLEMENTATION DETAILS

Following are some relevant implementation details of the algorithms compared in Section 5.

- **BRR-gb** uses our own implementation of "vanilla" gradient boosting, with XGBoost as the base learner, configured to fit a single regression tree using the "exact" splitting algorithm.
- **BRR-xgb** uses XGBoost's implementation of gradient boosted regression trees, with the "exact" splitting algorithm.
- **DRR** and **BanditNet** are implemented in MXNet (Chen et al., 2015). Each neural network starts with a hidden layer of a given width. For each successive hidden layer (up to the given depth), the width is halved, unless a minimum width of 32 is reached. Batch normalization (Ioffe & Szegedy, 2015) and ReLU activations are used for all hidden layers. For training, we use AdaGrad (Duchi et al., 2011) with early stopping on the training reward (estimated via self-normalized IPS (Swaminathan & Joachims, 2015b)) and a "patience" of 10 epochs.
- **BOPL** and **BOPL-S** use our own implementations. The **-regr** variants use XGBoost as the base learner, configured to fit a single regression tree using the "exact" splitting algorithm. The **-class** variants use Scikit-learn's (Pedregosa et al., 2011) decision tree classifier as the base learner. We use early stopping whenever the gradient, base predictions or ensemble weight are less than $10^{-10}$ in magnitude.

It is important to note that fitting a single regression tree in XGBoost is equivalent (modulo optimizations) to fitting one in a dedicated tree learner (such as Scikit-learn's). This equivalence is unique to the squared error loss, since XGBoost's second-order Taylor approximation is, in this case, exact.

All hyperparameters are tuned via random search and evaluated against held-out validation data for each dataset. Table 4 catalogs the hyperparameters and their associated ranges; Table 5 catalogs the values that were selected for each dataset.

Table 4: Hyperparameter ranges for each method and dataset.

| **Method** | **Parameter** | **Covertype** | **Fashion** | **Scene** | **TMC2007-500** |
|---|---|---|---|---|---|
| **BRR-gb** | rounds | $[200, 600]$ | $[100, 500]$ | $[100, 1000]$, | $[100, 300]$ |
| | max_depth | $[6, 25]$ | $[4, 20]$ | $[5, 14]$ | $[6, 20]$ |
| | min_child_weight | $[2, 400]$ | $[2, 640]$ | $[5, 120]$ | $[2, 200]$ |
| | reg_lambda | $[0]$ | $[0]$ | $[0, .05]$ | $[0]$ |
| | learning_rate | $[0.01, 0.1]$ | $[0.1, 2]$ | $[0.0001, 0.3]$ | $[0.01, 0.1]$ |
| **BRR-xgb** | rounds (n_estimators) | $[400, 600]$ | $[100, 500]$ | $[100, 2000]$ | $[100, 300]$ |
| | max_depth | $[6, 25]$ | $[10, 20]$ | $[3, 10]$ | $[6, 20]$ |
| | min_child_weight | $[2, 200]$ | $[40, 320]$ | $[5, 120]$ | $[2, 200]$ |
| | reg_lambda | $[0, 1.0]$ | $[0, 0.1]$ | $[0, 0.05]$ | $[0, 1.0]$ |
| | learning_rate | $[0.01, 0.1]$ | $[0.05, 0.1]$ | $[0.001, 0.1]$ | $[0.01, 0.1]$ |
| **DRR** | num_hidden_layers | $[1, 8]$ | $[1, 8]$ | $[1, 8]$ | $[1, 8]$ |
| | first_layer_width | $[32, 256]$ | $[32, 512]$ | $[32, 512]$ | $[32, 512]$ |
| | dropout | $[0.1, 0.2]$ | $[0.1, 0.2]$ | $[0.1, 0.2]$ | $[0.1, 0.2]$ |
| | weight_decay | $[0, 0.001]$ | $[0, 0.001]$ | $[0, 0.001]$ | $[0, 0.001]$ |
| | learning_rate | $[0.01, 0.1]$ | $[0.01, 0.1]$ | $[0.01, 0.1]$ | $[0.01, 0.1]$ |
| | epochs | $[116]$ | $[1000]$ | $[1000]$ | $[1000]$ |
| | batch_size | $[1000]$ | $[100]$ | $[100]$ | $[100]$ |
| **BanditNet** | num_hidden_layers | $[1, 8]$ | $[1, 8]$ | $[1, 8]$ | $[1, 8]$ |
| | first_layer_width | $[32, 256]$ | $[32, 512]$ | $[32, 512]$ | $[32, 512]$ |
| | dropout | $[0.1, 0.2]$ | $[0.1, 0.2]$ | $[0.1, 0.2]$ | $[0.1, 0.2]$ |
| | weight_decay | $[0, 0.001]$ | $[0, 0.001]$ | $[0, 0.001]$ | $[0, 0.001]$ |
| | learning_rate | $[0.01, 0.1]$ | $[0.01, 0.1]$ | $[0.01, 0.1]$ | $[0.01, 0.1]$ |
| | epochs | $[116]$ | $[1000]$ | $[1000]$ | $[1000]$ |
| | batch_size | $[1000]$ | $[100]$ | $[100]$ | $[100]$ |
| | reward_translation | $[-0.50]$ | $[-0.50]$ | $[-0.60]$ | $[-0.50]$ |
| **BOPL-regr** | rounds | $[200, 600]$ | $[150, 250]$ | $[100, 1000]$ | $[100, 300]$ |
| | max_depth | $[10, 25]$ | $[10, 20]$ | $[5, 14]$ | $[6, 20]$ |
| | min_child_weight | $[2, 400]$ | $[100, 640]$ | $[5, 120]$ | $[2, 200]$ |
| | reg_lambda | $[0]$ | $[0]$ | $[0, 0.05]$ | $[0]$ |
| | reward_translation | $[-0.4, -0.3]$ | $[-0.5, -0.3]$ | $[-0.60]$ | $[-0.5, -0.3]$ |
| **BOPL-class** | rounds | $[300]$ | $[150, 250]$ | $[500, 1000]$ | $[200]$ |
| | max_depth | $[15, 25]$ | $[15, 25]$ | $[6, 14]$ | $[15, 20]$ |
| | min_child_weight | $[50, 100]$ | $[10, 50]$ | $[60, 240]$ | $[10, 50]$ |
| | reward_translation | $[-0.2]$ | $[-0.2]$ | $[-0.4, -0.2]$ | $[-0.15, -0.1]$ |
| **BOPL-S-regr** | rounds | $[400, 600]$ | $[100, 250]$ | $[100, 1000]$ | $[100, 300]$ |
| | max_depth | $[10, 25]$ | $[10, 20]$ | $[5, 14]$ | $[6, 20]$ |
| | min_child_weight | $[2, 400]$ | $[100, 640]$ | $[5, 120]$ | $[2, 200]$ |
| | reg_lambda | $[0]$ | $[0]$ | $[0, 0.05]$ | $[0]$ |
| | reward_translation | $[-0.4, -0.3]$ | $[-0.5, -0.3]$ | $[-0.60]$ | $[-0.5, -0.3]$ |
| **BOPL-S-class** | rounds | $[300]$ | $[150, 250]$ | $[500, 1000]$ | $[200]$ |
| | max_depth | $[15, 25]$ | $[15, 25]$ | $[6, 14]$ | $[15, 20]$ |
| | min_child_weight | $[50, 100]$ | $[10, 50]$ | $[60, 240]$ | $[10, 50]$ |
| | reward_translation | $[-0.2]$ | $[-0.2]$ | $[-0.4, -0.2]$ | $[-0.15, -0.1]$ |

Table 5: Selected hyperparameters for each method and dataset.

| Method | Parameter | Covertype | Fashion | Scene | TMC2007-500 |
|---|---|---|---|---|---|
| **BRR-gb** | rounds | 600 | 250 | 1000 | 100 |
| | max_depth | 20 | 10 | 12 | 10 |
| | min_child_weight | 2 | 100 | 20 | 2 |
| | reg_lambda | 0 | 0 | 0.01 | 0 |
| | learning_rate | 0.1 | 0.1 | 0.01 | 0.1 |
| **BRR-xgb** | rounds (n_estimators) | 400 | 500 | 2000 | 200 |
| | max_depth | 25 | 15 | 8 | 10 |
| | min_child_weight | 10 | 100 | 20 | 2 |
| | reg_lambda | 1 | 0 | 0.05 | 1 |
| | learning_rate | 0.1 | 0.1 | 0.005 | 0.1 |
| **DRR** | num_hidden_layers | 4 | 4 | 2 | 1 |
| | first_layer_width | 256 | 512 | 512 | 256 |
| | dropout | 0.1 | 0.1 | 0.2 | 0.1 |
| | weight_decay | 0 | $10^{-6}$ | $10^{-5}$ | $10^{-6}$ |
| | learning_rate | 0.01 | 0.01 | 0.01 | 0.01 |
| | epochs | 116 | 1000 | 1000 | 1000 |
| | early_stopping_patience | 10 | 10 | 10 | 10 |
| | batch_size | 1000 | 100 | 100 | 100 |
| **BanditNet** | num_hidden_layers | 4 | 4 | 4 | 2 |
| | first_layer_width | 256 | 512 | 512 | 512 |
| | dropout | 0.1 | 0.1 | 0.1 | 0.1 |
| | weight_decay | $10^{-4}$ | $10^{-5}$ | $10^{-4}$ | $10^{-3}$ |
| | learning_rate | 0.01 | 0.01 | 0.01 | 0.01 |
| | epochs | 116 | 1000 | 1000 | 1000 |
| | early_stopping_patience | 10 | 10 | 10 | 10 |
| | batch_size | 1000 | 100 | 100 | 100 |
| | reward_translation | $-0.4$ | $-0.42$ | $-0.2$ | $-0.2$ |
| **BOPL-regr** | rounds | 600 | 250 | 500 | 300 |
| | max_depth | 25 | 20 | 12 | 20 |
| | min_child_weight | 200 | 200 | 60 | 200 |
| | reg_lambda | 0 | 0 | 0.01 | 0 |
| | reward_translation | $-0.4$ | $-0.41$ | $-0.2$ | $-0.3$ |
| **BOPL-class** | rounds | 300 | 150 | 1000 | 200 |
| | max_depth | 25 | 25 | 14 | 20 |
| | min_child_weight | 50 | 50 | 60 | 50 |
| | reward_translation | $-0.2$ | $-0.2$ | $-0.35$ | $-0.1$ |
| **BOPL-S-regr** | rounds | 600 | 250 | 100 | 300 |
| | max_depth | 25 | 20 | 10 | 20 |
| | min_child_weight | 200 | 200 | 60 | 200 |
| | reg_lambda | 0 | 0 | 0 | 0 |
| | reward_translation | $-0.4$ | $-0.4$ | $-0.2$ | $-0.3$ |
| **BOPL-S-class** | rounds | 300 | 150 | 1000 | 200 |
| | max_depth | 25 | 25 | 14 | 20 |
| | min_child_weight | 50 | 50 | 120 | 50 |
| | reward_translation | $-0.2$ | $-0.2$ | $-0.2$ | $-0.1$ |