A Contextual Bandit Bake-off

Alberto Bietti, Alekh Agarwal, John Langford

A Contextual Bandit Bake-off

Alberto Bietti
Inria,* MSR-Inria Joint Center
alberto.bietti@inria.fr

Alekh Agarwal
Microsoft Research
alekha@microsoft.com

John Langford
Microsoft Research
jcl@microsoft.com

Abstract

Contextual bandit algorithms are essential for solving many real-world interactive machine learning problems. Despite multiple recent successes on statistically and computationally efficient methods, the practical behavior of these algorithms is still poorly understood. We leverage the availability of large numbers of supervised learning datasets to compare and empirically optimize contextual bandit algorithms, focusing on practical methods that learn by relying on optimization oracles from supervised learning. We find that a recent method [14] using optimism under uncertainty works the best overall. A surprisingly close second is a simple greedy baseline that only explores implicitly through the diversity of contexts, followed by a variant of Online Cover [4] which tends to be more conservative but robust to problem specification by design. Along the way, we also evaluate and improve several internal components of contextual bandit algorithm design. Overall, this is a thorough study and review of contextual bandit methodology.

1 Introduction

At a practical level, how should contextual bandit learning and exploration be done?

In the contextual bandit problem, a learner repeatedly observes a context, chooses an action, and observes a loss for the chosen action only. Many real-world interactive machine learning tasks are well-suited to this setting: a movie recommendation system selects a movie for a given user and receives feedback (click or no click) only for that movie; a choice of medical treatment may be prescribed to a patient with an outcome observed for (only) the chosen treatment. The limited feedback (known as *bandit* feedback) received by the learner highlights the importance of *exploration*, which needs to be addressed by contextual bandit algorithms.

The focal point of contextual bandit (henceforth CB) learning research is efficient exploration algorithms [1, 3, 4, 6, 11, 24, 32]. However, many of these algorithms remain far from practical, and even when considering more practical variants, their empirical behavior is poorly understood, typically with limited evaluation on just a handful of scenarios. In particular, strategies based on upper confidence bounds [1, 25] or Thompson sampling [6, 32] are often intractable for sparse, high-dimensional datasets, and make strong assumptions on the model representation. The statistically optimal method of [4] alleviates some of these difficulties, but the analyzed version is still far from practical, and the worst-case guarantees may lead to overly conservative exploration that can be inefficient in practice.

Our work considers practical methods that are relevant to practitioners. We focus on algorithms that rely on *optimization oracles* from supervised learning such as cost-sensitive classification or regression oracles, which provides computational efficiency and support for generic representations. We further rely on online learning implementations of the oracles, which are desirable in practice due to the sequential nature of contextual bandits. While confidence-based strategies and Thompson

*Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France. Part of this work was done while AB was visiting Microsoft Research NY.

Preprint. Work in progress.

Figure 1: A greedy approach can outperform an optimized exploration method in many cases. Red points indicate datasets with a *statistically significant* difference in loss between two methods.

sampling are not directly adapted to this setting, we achieve it with online Bootstrap approximations for Thompson sampling [4, 13, 28], and with the confidence-based method of [14] based on regression oracles, which contains LinUCB as a special case. Additionally, we consider practical design choices such as loss encodings (*e.g.*, if values have a range of 1, should we encode the costs of best and worst outcomes as 0/1 or -1/0?), and for methods that learn by reduction to off-policy learning, we study different reduction techniques beyond the simple inverse propensity scoring approach. All of our experiments are based on the online learning system Vowpal Wabbit[2] which has already been successfully used in production systems [2].

The interactive aspect of CB problems makes them notoriously difficult to evaluate in real-world settings beyond a handful of tasks. Instead, we leverage the wide availability of supervised learning datasets with different cost structures on their predictions, and obtain contextual bandit instances by simulating bandit feedback, treating labels as actions and hiding the loss of all actions but the chosen one. This setup captures the generality of the i.i.d. contextual bandit setting, while avoiding some difficult aspects of real-world settings that are not supported by most existing algorithms, such as non-stationarity. We consider a large collection of over 500 datasets with varying levels of difficulty and various cost structures, including multiclass, multilabel and more general cost-sensitive datasets with real-valued costs. To our knowledge, this is the first evaluation of contextual bandit algorithms on such a large and diverse corpus of datasets.

Our evaluation considers online implementations of Bootstrap Thompson sampling [4, 13, 28], the Cover approach of [4], ϵ-greedy [24], RegCB [14] (which includes LinUCB as a special case), and a basic greedy method similar to the one studied in [7, 21]. We propose simple modifications of each method that improve practical performance, as well as a novel off-policy learning technique for exploration algorithms which use this as a building block (e.g. ϵ-greedy). As the first conclusion of our study, we find that the recent RegCB method [14] performs the best overall across a number of experimental conditions. Remarkably, we discover that a close second in our set of methods is the simple greedy baseline, often outperforming most exploration algorithms. Both these methods have drawbacks in theory; greedy can fail arbitrarily poorly in problems where intentional exploration matters, while UCB methods make stronger modeling assumptions and can have an uncontrolled regret when the assumptions fail. The logs collected by deploying these methods in practice are also unfit for later off-policy experiments, an important practical consideration. Our third conclusion is that several methods which are more robust in that they make only a relatively milder i.i.d. assumption on the problem tend to be overly conservative and often pay a steep price on easier datasets. Nevertheless, we find an adaptation of Online Cover [4] is quite competitive on a large fraction of our datasets. We show pairwise comparisons between the top 3 methods in our evaluation in Figure 1 for datasets with 5 or more actions. For future theoretical research, our results motivate an emphasis on understanding greedy strategies, building on recent progress [7, 21], as well as effectively leveraging easier datasets in existing exploration problems [5].

[2] `http://hunch.net/~vw/`

2 Contextual Bandit Setup

In this section, we present the learning setup considered in this paper, recalling the stochastic contextual bandit setting, the notion of optimization oracles, various techniques used by contextual bandit algorithms for leveraging these oracles, and finally our experimental setup.

2.1 Learning Setting

The stochastic (i.i.d.) contextual bandit learning problem can be described as follows. At each time step t, the environment produces a pair $(x_t, \ell_t) \sim D$, where $x_t \in \mathcal{X}$ is a context vector and $\ell_t = (\ell_t(1), \ldots, \ell_t(K)) \in \mathbb{R}^K$ is a loss vector, with K the number of possible actions. After observing the context x_t, the learner chooses an action a_t, and only observes the loss $\ell_t(a_t)$ corresponding to the chosen action. The goal of the learner is to trade-off exploration and exploitation in order to incur a small cumulative regret

$$R_T := \sum_{t=1}^{T} \ell_t(a_t) - \min_{\pi \in \Pi} \sum_{t=1}^{T} \ell_t(\pi(x_t)),$$

where Π is a (large, possibly infinite) set of policies $\pi : \mathcal{X} \to \{1, \ldots, K\}$. It is often important for the learner to use randomized strategies, hence we let $p_t(a) \in [0, 1]$ denote the probability that the agent chooses action $a \in \{1, \ldots, K\}$, so that $a_t \sim p_t$.

2.2 Optimization Oracle

In this paper, we focus on CB algorithms which rely on access to an *optimization oracle* for solving optimization problems similar to those that arise in supervised learning, leading to methods that are suitable for general policy classes Π. The main example is the **cost-sensitive classification (CSC) oracle** [4, 11, 24], which given a collection $(x_1, c_1), \ldots, (x_T, c_T) \in \mathcal{X} \times \mathbb{R}^K$ computes

$$\arg\min_{\pi \in \Pi} \sum_{t=1}^{T} c_t(\pi(x_t)). \tag{1}$$

The cost vectors $c_t = (c_t(1), \ldots, c_t(K)) \in \mathbb{R}^K$ are often constructed using counterfactual estimates of the true (unobserved) losses, as we describe in the next section.

Another approach is to use **regression oracles**, which find a regressor $f : \mathcal{X} \times \{1, \ldots, K\} \to \mathbb{R}$ from a class of functions $\mathcal{F}$ to predict a cost y_t, given a context x_t and action a_t (see, *e.g.*, [3, 14]). In this paper, we consider the following regression oracle with importance weights $\omega_t > 0$:

$$\arg\min_{f \in \mathcal{F}} \sum_{t=1}^{T} \omega_t (f(x_t, a_t) - y_t)^2. \tag{2}$$

While the theory typically requires exact solutions to (1) or (2), this is often impractical due to the difficulty of the underlying optimization problem (especially for CSC), and more importantly because the size of the problems to be solved keeps increasing after each iteration. In this work, we consider instead the use of *online* optimization for solving problems (1) or (2), by incrementally updating a given policy or regression function after each new observation, using for instance an online gradient method. This is natural in an online learning context such as the CB setting.

2.3 Loss Estimates and Reductions

A common approach to solving problems with bandit (partial) feedback is to compute an estimate of the full feedback using the observed loss and then apply methods for the full-information setting to these estimated values. In the case of CBs, this allows an algorithm to find a good policy based on *off-policy* exploration data collected by the algorithm. These loss estimates are commonly used to create CSC instances to be solved by the optimization oracle introduced above [4, 11, 24], a process sometimes referred as *reduction* to cost-sensitive classification. We now describe the three different estimation methods considered in this paper, and how each is typically used for reduction to

policy learning with a CSC or regression oracle. In what follows, we consider observed interaction records $(x_t, a_t, \ell_t(a_t), p_t(a_t))$.

Perhaps the simplest approach is the **inverse propensity-scoring** (IPS) estimator:

$$\hat{\ell}_t(a) := \frac{\ell_t(a_t)}{p_t(a_t)} \mathbb{1}\{a = a_t\}. \tag{3}$$

For any action a with $p_t(a) > 0$, this estimator is unbiased, *i.e.* $\mathbb{E}_{a_t \sim p_t}[\hat{\ell}_t(a)] = \ell_t(a)$, but can have high variance when $p_t(a_t)$ is small. The estimator leads to a straightforward CSC example $(x_t, \hat{\ell}_t)$. Using such examples in (1) provides a way to perform off-policy (or counterfactual) evaluation and optimization, which in turn allows a CB algorithm to identify good policies for exploration. In order to obtain good unbiased estimates, one needs to control the variance of the estimates, *e.g.*, by enforcing a minimum exploration probability $p_t(a) \geq \epsilon > 0$ on all actions.

In order to reduce the variance of IPS, the **doubly robust** (DR) estimator [12] uses a separate, possibly biased, estimator of the loss $\hat{\ell}(x, a)$:

$$\hat{\ell}_t(a) := \frac{\ell_t(a_t) - \hat{\ell}(x_t, a_t)}{p_t(a_t)} \mathbb{1}\{a = a_t\} + \hat{\ell}(x_t, a). \tag{4}$$

When $\hat{\ell}(x_t, a_t)$ is a good estimate of $\ell_t(a_t)$, the small numerator in the first term helps reduce the variance induced by a small denominator, while the second term ensures that the estimator is unbiased. Typically, $\hat{\ell}(x, a)$ is learned by regression on all past observed losses. The reduction to cost-sensitive classification is similar to IPS.

We introduce a third method that directly reduces to the importance-weighted regression oracle (2), which we call IWR (for **importance-weighted regression**), and is suitable for algorithms which rely on off-policy learning[3]. This approach finds a regression function

$$\hat{f} := \arg\min_{f \in \mathcal{F}} \sum_{t=1}^{T} \frac{1}{p_t(a_t)} (f(x_t, a_t) - \ell_t(a_t))^2, \tag{5}$$

and considers the policy $\hat{\pi}(x) = \arg\min_a \hat{f}(x, a)$. Note that if p_t has full support, then the objective is an unbiased estimate of the full regression objective on all actions, $\sum_{t=1}^{T} \sum_{a=1}^{K} (f(x_t, a) - \ell_t(a))^2$. In contrast, if the learner only explores a single action (so that $p_t(a_t) = 1$ for all t), and if we consider a linear class of regressors of the form $f(x, a) = \theta_a^\top x$ with $x \in \mathbb{R}^d$, then the IWR reduction computes least-squares estimates $\hat{\theta}_a$ from the data observed when action a was chosen. When actions are selected according to the greedy policy $a_t = \arg\min_a \hat{\theta}_a^\top x_t$, this setup corresponds to the greedy algorithm considered, *e.g.*, in [7].

Note that while CSC is typically intractable and requires approximations in order to work in practice, importance-weighted regression does not suffer from these issues. In addition, while the computational cost for an approximate CSC online update scales with the number of actions K, IWR only requires an update for a single action, making the approach more attractive computationally. Another benefit of IWR in an online setting is that it can leverage importance weight aware online updates [22], which makes it easier to handle large inverse propensity scores.

2.4 Experimental Setup

Our experiments are conducted by simulating the contextual bandit setting using multiclass classification datasets, and use the online learning system Vowpal Wabbit (VW).

Simulated contextual bandit setting. The experiments in this paper are based on leveraging supervised cost-sensitive classification datasets for simulating CB learning. In particular, we treat a CSC example $(x_t, c_t) \in \mathcal{X} \times \mathbb{R}^K$ as a CB example, with x_t given as the context to a CB algorithm, and we only reveal the loss of the action a_t. For a multiclass example with label $y_t \in \{1, \ldots, K\}$, we set $c_t(a) := \mathbb{1}\{a \neq y_t\}$; for multilabel examples with label set $Y_t \subseteq \{1, \ldots, K\}$, we set

[3] Note that IWR is not directly applicable to methods that explicitly reduce to CSC oracles, such as [4, 11].

$c_t(a) := \mathbb{1}\{a \notin Y_t\}$; the cost-sensitive datasets we consider have $c_t \in [0,1]^K$. We consider more general *loss encodings* defined by:

$$\ell_t^c(a) = c + c_t(a), \tag{6}$$

for some $c \in \mathbb{R}$. Although some techniques [12] attempt to remove a dependence on encoding, in practice, and particularly in an online scenario, this is imperfect. The behavior observed for different choices of c allows us to get a sense of the robustness of the algorithms to the scale of observed losses, which might be unknown. Separately, different values of c can lead to low estimation variance in different scenarios: $c = 0$ might be preferred if $c_t(a)$ is often 0, while $c = -1$ is preferred when $c_t(a)$ is often 1. In order to have a meaningful comparison between different algorithms, loss encodings, as well as supervised multiclass classification, we evaluate using c_t with progressive validation [8].

Online learning in VW. Online learning is an important tool for having machine learning systems that quickly and efficiently adapt to observed data [2, 16, 27]. We run our CB algorithms in an online fashion using Vowpal Wabbit: instead of exact solutions of the optimization oracles from Section 2.2, we consider online CSC or regression oracles. Online CSC itself reduces to multiple online regression problems in VW, solved with adaptive [10], normalized [31] and importance-weight-aware [22] gradient updates, with a single tunable step-size parameter.

Parameterization and *baseline*. We consider linearly parameterized policies of the form $\pi(x) = \arg\min_a \theta_a^\top x$, or in the case of the IWR reduction, regressors $f(x,a) = \theta_a^\top x$. For the DR loss estimator, we use a similar linear parameterization $\hat{\ell}(x,a) = \phi_a^\top x$. We note that the algorithms we consider do not rely on this specific form, and easily extend to more complex, problem-dependent representations, such as action-dependent features. We also consider the use of an *action-independent additive baseline* term in our loss estimators, which can help learn better estimates with fewer samples. In this case the regressors take the form $f(x,a) = \theta_0 + \theta_a^\top x$ (IWR) or $\hat{\ell}(x,a) = \phi_0 + \phi_a^\top x$ (DR). In order to learn the baseline term more quickly, we propose to use a separate online update for the parameters θ_0 or ϕ_0 to regress on observed losses, followed by an online update on the residual for the action-dependent part. We scale the step-size of these baseline updates by the largest observed magnitude of the loss, in order to adapt to the observed loss range for normalized updates [31].

3 Algorithms

In this section, we present the main algorithms we study in this paper, along with simple modifications that achieve improved exploration efficiency. More details as well as pseudo-code for each method are given in Appendix A, due to space limitations.

ϵ-greedy. We consider an importance-weighted variant of the epoch-greedy approach [24]. The method acts greedily with probability $1-\epsilon$, and otherwise explores uniformly on all actions.[4] Learning is achieved by reduction to off-policy optimization, through any of the three reductions presented in Section 2.3. We also consider an "active" variant that restricts exploration to only the actions which plausibly could be taken by the best policy. This is achieved using techniques from active learning [15, 17], and yields improved regret guarantees in favorable settings (see Appendix D).

Greedy. When taking $\epsilon = 0$ in the ϵ-greedy approach, with the IWR reduction, we are left with a fully greedy approach that always selects the action given by the current policy. This gives us an online variant of the greedy algorithm in [7], which regresses on observed losses and acts by selecting the action with minimum predicted loss. Somewhat surprisingly, our experiments show that Greedy can perform very well in practice, despite doing no explicit exploration (see Section 4).

Bagging. This approach maintains a collection of N policies meant to approximate a posterior distribution over policies (or, in Bayesian terminology, the parameter that generated the data) via the online Bootstrap [4, 13, 28, 29, 30], and explores in a Thompson sampling fashion [6, 9, 32, 33]. In contrast to [13, 28], which simply play the arm given by one of the N policies chosen at random, we compute the full action distribution resulting from such a sampling, and leverage this for loss

[4] Ties to pick the best action are broken randomly in ϵ-greedy, Greedy, Bagging and RegCB approaches as indicated in Appendix A.

estimation, allowing learning by reduction to off-policy optimization. Again all three reductions are admissible. We also consider a simple optimization that we call *greedy bagging*, for which the first policy is trained on the true data sample (like Greedy) instead of a bootstrap sample. We found this approach to often improve on bagging, particularly for small N.

Cover. This method is based on Online Cover, an online approximation of the ILOVETOCONBANDITS algorithm of Agarwal et al. [4]. The approach maintains a collection of N policies meant to approximate a covering distribution over policies that are good for both exploration and exploitation. The first policy is trained by reduction to off-policy optimization as in previous algorithms, while subsequent policies are trained using cost-sensitive examples which encourage diversity in the predicted actions compared to the previous policies. Our implementation introduces a parameter ψ which controls the level of diversity, as detailed in Appendix A.3. While Cover requires some uniform exploration across all actions, our experiments suggest that this can make exploration highly inefficient, thus we introduce a variant, *Cover-NU*, with *n*o *u*niform exploration outside the set of actions selected by covering policies.

RegCB. We consider online approximations of the two algorithms introduced in [14] based on regression oracles. Both algorithms estimate confidence intervals of the loss for each action given the current context, by considering regressors that are good for loss estimation. The *optimistic* variant then selects the action with smallest lower bound estimate, similar to LinUCB, while the *elimination* variant explores uniformly on actions that may plausibly be the best. Our online implementation, detailed in Section A.4, computes these upper and lower bounds on the loss of each action using a sensitivity analysis of the current regressor based on importance weighting, similar to [23] in the context of active learning. We note that this requires knowledge of the loss range, in contrast to other methods. The width of confidence bounds is controlled by a parameter C_0.

4 Evaluation

In this section, we present our evaluation of the contextual bandit algorithms described in Section 3. All code will be made public.

Evaluation setup. Our evaluation consists in simulating a CB setting from cost-sensitive classification datasets, as described in Section 2.4. We consider a collection of 524 multiclass classification datasets from the `openml.org` platform, including among others, medical, gene expression, text, sensory or synthetic data, as well as 5 multilabel datasets[5] and a cost-sensitive version of the RCV1 multilabel dataset used in [23], where the cost of a news topic is equal to the tree distance to a correct topic. More details on these datasets are given in Appendix B. Because of the online setup, we consider a fixed, shuffled ordering of each dataset. The datasets widely vary in noise levels, and number of actions, features, examples etc., allowing us to model varying difficulties in CB problems.

We evaluate the algorithms described in Section 3. We ran each method on every dataset with different choices of algorithm-specific hyperparameters, learning rates, reductions, and loss encodings with or without *baseline*. Details are given in Appendix C.1. Unless otherwise specified, we consider *fixed choices* which are chosen to optimize performance on a subset of multiclass datasets with a voting mechanism and are highlighted in Table 7 of Appendix C, except for the learning rate, which is always optimized.

The performance of method $\mathcal{A}$ on a dataset of size n is measured by the **progressive validation loss** [8]:

$$PV_{\mathcal{A}} = \frac{1}{n}\sum_{t=1}^{n} c_t(a_t),$$

where a_t is the action chosen by the algorithm on the t-th example, and c_t the true cost vector. This metric allows us to capture the explore-exploit trade-off, while providing a measure of generalization that is independent of the choice of loss encodings, and comparable with online supervised learning. We also consider a *normalized loss* variant given by $\frac{PV_{\mathcal{A}} - PV_{\text{OAA}}}{PV_{\text{OAA}}}$, where OAA denotes an online (supervised) cost-sensitive one against all classifier. This helps highlight the difficulty of exploration for some datasets in our plots.

[5] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html`

Table 1: Entry (row, column) shows the *statistically significant* win-loss difference of row against column. Encoding fixed to -1/0. (left) held-out datasets only, fixed hyperparameters, only the learning rate is optimized; (right) choice of hyperparameters, reduction, baseline are optimized on each dataset (different methods have different hyperparameter settings from 2 to 36, see Table 7 in Appendix C).

$\downarrow$ vs $\rightarrow$	G	RO	C-nu	B-g	ϵG
G	-	-7	9	50	53
RO	**7**	-	**25**	**48**	**67**
C-nu	-9	-25	-	22	56
B-g	-50	-48	-22	-	16
ϵG	-53	-67	-56	-16	-

$\downarrow$ vs $\rightarrow$	G	RO	C-nu	B-g	ϵG
G	-	-23	-50	16	91
RO	23	-	-4	41	111
C-nu	**50**	**4**	-	**64**	**141**
B-g	-16	-41	-64	-	89
ϵG	-91	-111	-141	-89	-

Table 2: Progressive validation loss for RCV1 with real-valued costs in $[0, 1]$ based on tree distance. The learning rate is optimized on the original dataset, and we show mean and standard error based on 10 different random reshufflings.

G	RO	C-nu	B-g	ϵG
0.215 $\pm$ 0.010	0.225 $\pm$ 0.008	**0.215** $\pm$ 0.006	0.251 $\pm$ 0.005	0.230 $\pm$ 0.009

In order to compare two methods on a given dataset with binary costs (multiclass or multilabel), we consider a notion of **statistically significant win or loss**. We use the following definition of significance based on a Z-test: if p_a and p_b denote the PV loss of a and b on a given dataset of size n, then a wins over b if

$$1 - \Phi\left(\frac{p_a - p_b}{\sqrt{\frac{p_a(1-p_a)}{n} + \frac{p_b(1-p_b)}{n}}}\right) < 0.05,$$

where Φ is the Gauss error function. We also define the *significant win-loss difference* of one algorithm against another to be the difference between the number of significant wins and significant losses. We have found these metrics to provide more insight into the behavior of different methods, compared to strategies based on aggregation of loss measures across all datasets. Indeed, we often found the relative performance of two methods to vary significantly across datasets, making aggregate metrics less appealing.

Results in this section focus on Greedy (G), RegCB-optimistic (RO), Cover-NU (C-nu), Bag-greedy (B-g) and ϵ-greedy (ϵG), deferring other variants to Appendix C, as their performance is typically comparable to or dominated by these methods. We combine results on multiclass and multilabel datasets, but show them separately in Appendix C.

Efficient exploration methods. Our experiments suggest that the best performing method is the RegCB approach [14], as shown in Table 1 (left), where the significant wins of RO against all other methods exceed significant losses for the -1/0 encoding, which yields the best performance overall. This is even more prominent with 0/1 encodings (see Appendix C.2). The simple greedy approach comes a close second, outperforming other methods on a large number of datasets, despite the lack of an explicit exploration mechanism. A possible reason for this success is the diversity that is inherently present in the distribution of contexts across actions, which has been shown to yield no-regret guarantees under various assumptions [7, 21]. The noise induced by the dynamics of online learning and random tie-breaking may also be a source of more exploration. Nevertheless, both Greedy and RegCB have known failure modes which the Cover approach is robust to by design While the basic approach with uniform exploration is too conservative, we found our Cover-NU variant to be quite competitive overall. The randomization in its exploration logs that can be additionally used for offline evaluation. A more granular comparison of these methods is given in Figure 1 and Figure 3 in Appendix C, which highlight the failure of Greedy and RegCB against Cover-NU on some of the more difficult datasets. Table 1 (right) optimizes over hyperparameters for each dataset, which captures the best potential of each method. Cover-NU does the best here, but also has the most hyperparameters, indicating that a more adaptive variant could be desirable. RegCB stays competitive, while Greedy pales possibly due to fewer hyperparameters.

Table 3: Impact of reductions for Bag (left) and ϵ-greedy (right), with hyperparameters optimized and encoding fixed to -1/0, without *baseline*. Each (row, column) entry shows the *statistically significant* win-loss difference of row against column. IWR outperforms the other reductions for both methods.

$\downarrow$ vs $\rightarrow$	ips	dr	iwr
ips	-	-42	-58
dr	42	-	-27
iwr	**58**	**27**	-

$\downarrow$ vs $\rightarrow$	ips	dr	iwr
ips	-	63	-132
dr	-63	-	-154
iwr	**132**	**154**	-

Table 4: Impact of encoding on different algorithms, with hyperparameters optimized and no *baseline*. Each entry indicates the number of *statistically significant* wins/losses of -1/0 against 0/1. -1/0 is the better overall choice of encoding, but 0/1 can be preferable on larger datasets (the bottom row considers the 64 datasets in our corpus with more than 10,000 examples).

datasets	G	RO	C-nu	B-g	ϵG
all	128 / 38	60 / 47	76 / 46	77 / 27	99 / 27
$\geq$ 10,000	16 / 12	10 / 18	14 / 20	15 / 11	14 / 5

IWR reduction. The IWR reduction introduced in Section 2.3 has desirable properties, such as a computational cost that is independent of the total number of actions, only requiring updates for the chosen action. In addition to Greedy, we found IWR to work very well for bagging and ϵ-greedy approaches, as shown in Table 3 (see also Table 7 in Appendix C, which shows that IWR is also preferred when considering fixed hyperparameters for these methods). This may be attributed to the difficulty of the CSC problem compared to regression, as well as importance weight aware online updates, which can be helpful for small ϵ. Together with its computational benefits, our results suggest that IWR is often a compelling alternative to CSC reductions based on IPS or DR. In particular, when the number of actions is prohibitively large for using Cover-NU or RegCB, Bag with IWR may be a good default choice of exploration algorithm. While Cover-NU does not directly support the IWR reduction, making them work together well would be a promising future direction.

Encodings. Table 4 indicates that the -1/0 encoding is preferred to 0/1 on many of the datasets, and for all methods. We now give one possible explanation. As discussed in Section 2.4, the -1/0 encoding yields low variance loss estimates when the cost is often close to 1. For datasets with binary costs, since the learner may often be wrong in early iterations, a cost of 1 is a good initial bias for learning. With enough data, however, the learner should reach better accuracies and observe losses closer to 0, in which case the 0/1 encoding should lead to lower variance estimates, yielding better performance as observed in Table 4. We tried shifting the loss range in the RCV1 dataset with real-valued costs from $[0, 1]$ to $[-1, 0]$, but saw no improvements compared to the results in Table 2. Indeed, a cost of 1 may not be a good initial guess in this case, in contrast to the binary cost setting.

Baseline. The additive baseline introduced in Section 2.4 can be helpful to quickly adapt to a constant loss estimate thanks to the separate online update. This appears particularly useful with the -1/0 encoding, for which the initialization at 0 may give pessimistic loss estimates which can be damaging in particular for the greedy method, that often gets some initial exploration from an optimistic cost encoding. This can be seen in Figure 2 (right) and Figure 4, Appendix C. In an online learning setting, baseline can also help to quickly reach an unknown target range of loss estimates. This is demonstrated in Figure 2, where the addition of baseline is shown to help various methods with 9/10 encodings on a large number of datasets. We do not evaluate RegCB as it needs a priori known upper and lower bounds on costs.

5 Discussion and Concluding Remarks

In this paper, we presented an evaluation of practical contextual bandit algorithms on a large collection of supervised learning datasets with simulated bandit feedback. We find that a worst-case theoretical robustness forces several common methods to often over-explore, damaging their empirical performance, and strategies that limit (RegCB and Cover-NU) or simply forgo (Greedy) explicit exploration dominate the field. For practitioners, our study also provides a reference for practical implementations, while stressing the importance of loss estimation and other design choices such

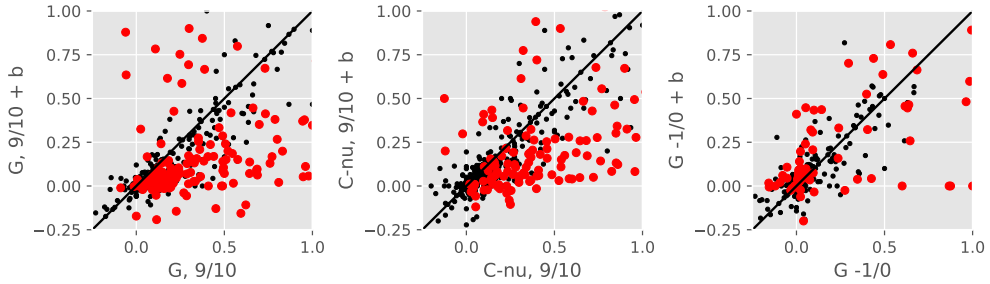


Figure 2: Baseline helps with difficult encodings and improves robustness to the range of losses.

as how to encode observed feedback. We acknowledge that these conclusions come with a grain of salt, as the violation of stationarity in real-world CB applications can change these considerations. The lack of public CB datasets as well as challenges in counterfactual evaluation of CB algorithms make a more realistic study challenging, but we hope that an emergence of platforms [2, 19] to easily deploy CB algorithms will enable studies with real CB datasets in the future.

References

[1] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[2] A. Agarwal, S. Bird, M. Cozowicz, L. Hoang, J. Langford, S. Lee, J. Li, D. Melamed, G. Oshri, O. Ribas, et al. A multiworld testing decision service. *arXiv preprint arXiv:1606.03966*, 2016.

[3] A. Agarwal, M. Dudík, S. Kale, J. Langford, and R. E. Schapire. Contextual bandit learning with predictable rewards. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

[4] A. Agarwal, D. Hsu, S. Kale, J. Langford, L. Li, and R. E. Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. *arXiv preprint arXiv:1402.0555*, 2014.

[5] A. Agarwal, A. Krishnamurthy, J. Langford, H. Luo, et al. Open problem: First-order regret bounds for contextual bandits. In *Conference on Learning Theory (COLT)*, 2017.

[6] S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013.

[7] H. Bastani, M. Bayati, and K. Khosravi. Exploiting the natural exploration in contextual bandits. *arXiv preprint arXiv:1704.09011*, 2017.

[8] A. Blum, A. Kalai, and J. Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Conference on Learning Theory (COLT)*, 1999.

[9] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

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

[11] M. Dudik, D. Hsu, S. Kale, N. Karampatziakis, J. Langford, L. Reyzin, and T. Zhang. Efficient optimal learning for contextual bandits. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2011.

[12] M. Dudik, J. Langford, and L. Li. Doubly robust policy evaluation and learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2011.

[13] D. Eckles and M. Kaptein. Thompson sampling with the online bootstrap. *arXiv preprint arXiv:1410.4009*, 2014.

[14] D. J. Foster, A. Agarwal, M. Dudík, H. Luo, and R. E. Schapire. Practical contextual bandits with regression oracles. In *ICML*, 2018.

[15] S. Hanneke. Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 7(2-3), 2014.

[16] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, 2014.

[17] D. J. Hsu. *Algorithms for active learning*. PhD thesis, UC San Diego, 2010.

[18] T.-K. Huang, A. Agarwal, D. J. Hsu, J. Langford, and R. E. Schapire. Efficient and parsimonious agnostic active learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[19] K. G. Jamieson, L. Jain, C. Fernandez, N. J. Glattard, and R. Nowak. Next: A system for real-world development, evaluation, and application of active learning. In *Advances in Neural Information Processing Systems*, pages 2656–2664, 2015.

[20] S. M. Kakade and A. Tewari. On the generalization ability of online strongly convex programming algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[21] S. Kannan, J. Morgenstern, A. Roth, B. Waggoner, and Z. S. Wu. A smoothed analysis of the greedy algorithm for the linear contextual bandit problem. *arXiv preprint arXiv:1801.03423*, 2018.

[22] N. Karampatziakis and J. Langford. Online importance weight aware updates. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2011.

[23] A. Krishnamurthy, A. Agarwal, T.-K. Huang, H. Daume III, and J. Langford. Active learning for cost-sensitive classification. *arXiv preprint arXiv:1703.01014*, 2017.

[24] J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[25] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, 2010.

[26] P. Massart, É. Nédélec, et al. Risk bounds for statistical learning. *The Annals of Statistics*, 34(5), 2006.

[27] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM international conference on Knowledge discovery and data mining (KDD)*, 2013.

[28] I. Osband and B. Van Roy. Bootstrapped thompson sampling and deep exploration. *arXiv preprint arXiv:1507.00300*, 2015.

[29] N. C. Oza and S. Russell. Online bagging and boosting. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2001.

[30] Z. Qin, V. Petricek, N. Karampatziakis, L. Li, and J. Langford. Efficient online bootstrapping for large scale learning. In *Workshop on Parallel and Large-scale Machine Learning (BigLearning@NIPS)*, 2013.

[31] S. Ross, P. Mineiro, and J. Langford. Normalized online learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2013.

[32] D. Russo, B. Van Roy, A. Kazerouni, and I. Osband. A tutorial on thompson sampling. *arXiv preprint arXiv:1707.02038*, 2017.

[33] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4), 1933.

Algorithm 1 Generic contextual bandit algorithm

for $t = 1, \ldots$ **do**
  Observe context x_t, compute $p_t = \texttt{explore}(x_t)$;
  Choose action $a_t \sim p_t$, observe loss $\ell_t(a_t)$;
  $\texttt{learn}(x_t, a_t, \ell_t(a_t), p_t)$;
end for

Algorithm 2 ϵ-greedy

π_1; $\epsilon > 0$ (or $\epsilon = 0$ for Greedy).
$\texttt{explore}(x_t)$:
  return $p_t(a) = \epsilon/K + (1-\epsilon)\,\mathbb{1}\{\pi_t(x_t) = a\}$;
$\texttt{learn}(x_t, a_t, \ell_t(a_t), p_t)$:
  $\pi_{t+1} = \texttt{oracle}(\pi_t, x_t, a_t, \ell_t(a_t), p_t(a_t))$;

A Algorithm Details

This section present the algorithms introduced in Section 3 in more detail. All methods are based on the generic scheme in Algorithm 1. `explore` computes the exploration distribution p_t over actions, and `learn` updates the algorithm's policies. For simiplicity, we consider a function `oracle` which performs an online update to a policy using IPS, DR or IWR reductions given an interaction record. In some cases, the CSC oracle is called explicitly with different cost vectors, and we denote such a call by `csc_oracle`, along with a loss estimator `estimator` which takes an interaction record and computes IPS or DR loss estimates. RegCB directly uses a regression oracle, which we denote `reg_oracle`.

A.1 ϵ-greedy and greedy

ϵ-greedy. We consider an importance-weighted variant of the epoch-greedy approach of Langford and Zhang [24], given in Algorithm 2. We also experimented with a variant we call active ϵ-greedy, that uses notions from disagreement-based active learning [15, 17] in order to reduce uniform exploration to only actions that could plausibly be taken by the optimal policy. This approach is detailed in Appendix D, along with a theoretical analysis.

Greedy. When taking $\epsilon = 0$, only the greedy action given by the current policy is explored. With the IWR reduction and the linear regressors described in Section 2.4, this corresponds to an online version of the greedy algorithm in [7].

If multiple actions get the same score according to the current regressor, we break ties randomly.

A.2 Bagging

Algorithm 3 Bag

$\pi_1^1, \ldots, \pi_1^N$.
$\texttt{explore}(x_t)$:
  return $p_t(a) \propto |\{i : \pi_t^i(x_t) = a\}|$;[6]
$\texttt{learn}(x_t, a_t, \ell_t(a_t), p_t)$:
  for $i = 1, \ldots, N$ **do**
    $\tau^i \sim Poisson(1)$; {with $\tau^1 = 1$ for bag-greedy}
    $\pi_{t+1}^i = \texttt{oracle}^{\tau^i}(\pi_t^i, x_t, a_t, \ell_t(a_t), p_t(a_t))$;
  end for

[6] When policies are parametrized using regressors as in our implementation, we let $\pi_t^i(x)$ be uniform over all actions tied for the lowest cost, and the final distribution is uniform across all actions tied for best according to one of the policies in the bag. The added randomization gives useful variance reduction in our experiments.

Algorithm 4 Cover

$\pi_1^1, \ldots, \pi_1^N$; $\epsilon_t = \min(1/K, 1/\sqrt{Kt})$; $\psi > 0$.
`explore`(x_t):
  $p_t(a) \propto |\{i : \pi_t^i(x_t) = a\}|$;
  return $\epsilon_t + (1 - \epsilon_t)p_t$; {for cover}
  return p_t; {for cover-nu}
`learn`$(x_t, a_t, \ell_t(a_t), p_t)$:
  $\pi_{t+1}^1 =$ `oracle`$(\pi_t^1, x_t, a_t, \ell_t(a_t), p_t(a_t))$;
  $\hat{\ell}_t =$ `estimator`$(x_t, a_t, \ell_t(a_t), p_t(a_t))$;
  for $i = 2, \ldots, N$ **do**
    $q_i(a) \propto |\{j \leq i - 1 : \pi_{t+1}^j(x_t) = a\}|$;
    $\hat{c}(a) = \hat{\ell}_t(a) - \frac{\psi\epsilon_t}{\epsilon_t + (1-\epsilon_t)q_i(a)}$;
    $\pi_{t+1}^i =$ `csc_oracle`$(\pi_t^i, x_t, \hat{c})$;
  end for

This approach, shown in Algorithm 3, maintains a collection of N policies $\pi_t^1, \ldots, \pi_t^N$ meant to approximate a posterior distribution over policies (or, in Bayesian terminology, the parameter that generated the data) via the Bootstrap. This approximate posterior is used to choose actions in a Thompson sampling fashion [6, 9, 32, 33]. Each policy is trained on a different online bootstrap sample of the observed data [30, 29]. The online bootstrap performs a random number τ of online updates to each policy instead of one (this is denoted by `oracle`$^\tau$). This is also known as online Bootstrap Thompson sampling [13, 28]. In contrast to these works, which simply play the arm given by one of the N policies chosen at random, we compute the full action distribution p_t resulting from such a sampling, and leverage this information for improved importance weights in loss estimation as in previous work [4].

Greedy bagging. With a single policy ($N = 1$), Algorithm 3 resembles the Greedy algorithm, up to the randomized number of online updates. We found this method to often be outperformed by Greedy, thus our experiments consider a simple optimization of bagging where the first policy is always updated once ($\tau^1 = 1$ in Algorithm 3), which typically performs better than bagging, though not significantly for larger values of N.

A.3 Cover

This method, given in Algorithm 4, is based on Online Cover, an online approximation of the ILOVETOCONBANDITS algorithm of [4]. The approach maintains a collection of N policies, $\pi_t^1, \ldots, \pi_t^N$, meant to approximate a covering distribution over policies that are good for both exploration and exploitation. The first policy π_t^1 is trained on observed data using the oracle as in previous algorithms, while subsequent policies are trained using cost-sensitive examples which encourage diversity in the predicted actions compared to the previous policies.

Our implementation differs from the Online Cover algorithm [4, Algorithm 5] in how the diversity term in the definition of $\hat{c}(a)$ is handled (the second term). When creating cost-sensitive examples for a given policy π^i, this term rewards an action a that is not well-covered by previous policies (*i.e.*, small $q_i(a)$), by subtracting a term that decreases with $q_i(a)$ from the loss. While Online Cover considers a fixed $\epsilon_t = \epsilon$, we let ϵ_t decay with t, and introduce a parameter ψ to control the overall reward term, which bears more similarity with the analyzed algorithm. In particular, the magnitude of the reward is ψ whenever action a is not covered by previous policies (*i.e.*, $q_i(a) = 0$), but decays with $\psi\epsilon_t$ whenever $q_i(a) > 0$, so that the level of induced diversity can decrease over time as we gain confidence that good policies are covered.

Cover-NU. In order to reduce the level of exploration of Cover and be more competitive with the Greedy method, we propose a variant of Cover with no exploration outside of the actions chosen by covering policies, denoted by Cover-NU (for No Uniform exploration).

A.4 RegCB

Algorithm 5 RegCB

f_1; $C_0 > 0$.
`explore`(x_t):
  $l_t(a) =$ `lcb`$(f_t, x_t, a, \Delta_{t,C_0})$;
  $u_t(a) =$ `ucb`$(f_t, x_t, a, \Delta_{t,C_0})$;
  $p_t(a) \propto \mathbb{1}\{a \in \arg\min_{a'} l_t(a')\}$; {RegCB-opt variant}
  $p_t(a) \propto \mathbb{1}\{l_t(a) \leq \min_{a'} u_t(a')\}$; {RegCB-elim variant}
  return p_t;
`learn`$(x_t, a_t, \ell_t(a_t), p_t)$:
  $f_{t+1} =$ `reg_oracle`$(f_t, x_t, a_t, \ell_t(a_t))$;

We consider the confidence-bound algorithm introduced by Foster et al. [14] based on regression oracles, together with an online approximation that follows the sensitivity analysis given by Krishnamurthy et al. [23] in the context of active learning. Our method is shown in Algorithm 5. The algorithm maintains a regressor $f_t : \mathcal{X} \times \{1, \ldots, K\}$ and, given a new context x_t, computes lower and upper confidence bounds $l_t(a) \leq f_t(x_t, a) \leq u_t(a)$. These are computed by adding "virtual" importance-weighted regression examples with low and high costs, and finding the largest importance weight leading to an excess squared loss smaller than Δ_{t,C_0}, where

$$\Delta_{t,C_0} = \frac{C_0 \log(Kt)}{t},$$

and C_0 controls the width of the confidence bounds. This importance weight can be found using regressor sensitivities and a binary search procedure as described in [23, Section 7.1]. Note that this requires costs to be in a known range $[c_{min}, c_{max}]$. In contrast to [23], we set the labels of the "virtual" examples to $c_{min} - 1$ for the lower bound and $c_{max} + 1$ for the upper bound, instead of c_{min} and c_{max}.

The *optimistic* variant then selects the action with smallest lower bound estimate, similar to LinUCB, while the *elimination* variant explores uniformly on actions that may plausibly be the best.

B Datasets

This section gives some details on the cost-sensitive classification datasets considered in our study.

Multiclass classification datasets. We consider 524 multiclass datasets from the `openml.org` platform, including among others, medical, gene expression, text, sensory or synthetic data. Table 5 provides some statistics about these datsets. The full list of datasets is given below.

Table 5: Statistics on number of multiclass datasets by number of examples, actions and unique features in our collection of 524 datasets.

actions	#
2	403
3-9	73
10+	48

examples	#
$\leq 10^2$	94
10^2-10^3	270
10^3-10^5	132
$> 10^5$	28

features	#
≤ 50	392
51-100	35
101-1000	16
1000+	81

Multilabel classification datasets. We consider 5 multilabel datasets from the LibSVM website[7], listed in Table 6.

Cost-sensitive classification datasets. For more general real-valued costs in $[0, 1]$, we use a modification of the multilabel RCV1 dataset introduced in [23]. Each example consists of a news article labeled with the topics it belongs to, in a collection of 103 topics. Instead of fixing the cost to 1 for incorrect topics, the cost is defined as the tree distance to the set of correct topics in a topic hierarchy.

[7] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html`

Table 6: List of multilabel datasets.

Dataset	# examples	# features	# actions
mediamill	30,993	120	101
rcv1	23,149	47,236	103
scene	1,211	294	6
tmc	21,519	30,438	22
yeast	1,500	103	14

List of multiclass datasets. The datasets we used can be accessed at `https://www.openml.org/d/<id>`, with `id` in the following list:

3, 6, 8, 10, 11, 12, 14, 16, 18, 20, 21, 22, 23, 26, 28, 30, 31, 32, 36, 37, 39, 40, 41, 43, 44, 46, 48, 50, 53, 54, 59, 60, 61, 62, 150, 151, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 180, 181, 182, 183, 184, 187, 189, 197, 209, 223, 227, 273, 275, 276, 277, 278, 279, 285, 287, 292, 293, 294, 298, 300, 307, 310, 312, 313, 329, 333, 334, 335, 336, 337, 338, 339, 343, 346, 351, 354, 357, 375, 377, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 444, 446, 448, 450, 457, 458, 459, 461, 462, 463, 464, 465, 467, 468, 469, 472, 475, 476, 477, 478, 479, 480, 554, 679, 682, 683, 685, 694, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 758, 759, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 782, 783, 784, 785, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 799, 800, 801, 803, 804, 805, 806, 807, 808, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 832, 833, 834, 835, 836, 837, 838, 841, 843, 845, 846, 847, 848, 849, 850, 851, 853, 855, 857, 859, 860, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 884, 885, 886, 888, 891, 892, 893, 894, 895, 896, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 931, 932, 933, 934, 935, 936, 937, 938, 941, 942, 943, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 958, 959, 962, 964, 965, 969, 970, 971, 973, 974, 976, 977, 978, 979, 980, 983, 987, 988, 991, 994, 995, 996, 997, 1004, 1005, 1006, 1009, 1011, 1012, 1013, 1014, 1015, 1016, 1019, 1020, 1021, 1022, 1025, 1026, 1036, 1038, 1040, 1041, 1043, 1044, 1045, 1046, 1048, 1049, 1050, 1054, 1055, 1056, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1071, 1073, 1075, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1100, 1104, 1106, 1107, 1110, 1113, 1115, 1116, 1117, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1169, 1216, 1217, 1218, 1233, 1235, 1236, 1237, 1238, 1241, 1242, 1412, 1413, 1441, 1442, 1443, 1444, 1449, 1451, 1453, 1454, 1455, 1457, 1459, 1460, 1464, 1467, 1470, 1471, 1472, 1473, 1475, 1481, 1482, 1483, 1486, 1487, 1488, 1489, 1496, 1498.

C Evaluation Details

C.1 Algorithms and Hyperparameters

We ran each method on every dataset with the following hyperparameters:

- algorithm-specific *hyperparameters*, shown in Table 7.
- 9 choices of *learning rates*, on a logarithmic grid from 0.001 to 10 (see Section 2.4).
- 3 choices of *reductions*: IPS, DR and IWR (see Section 2.3). Note that these mainly apply to methods that reduce to off-policy optimization (*i.e.*, (ϵ-)greedy and bagging), and to some extent, methods that reduce to cost-sensitive classification (*i.e.*, cover and active ϵ-greedy, though the IWR reduction is heuristic in this case). Both RegCB variants directly reduce to regression.
- 3 choices of loss *encodings*: 0/1, -1/0 and 9/10 (see Eq. (6)). 0/1 and -1/0 encodings are typically a design choice, while the experiments with 9/10 are aimed at assessing some robustness to loss range.
- with or without *baseline* (denoted 'b', see Section 2.4).

Table 7: Choices of hyperparameters, reduction, and *baseline* for each method. Fixed choices of hyperparameters for -1/0 encodings and no *baseline* are in **bold**. These were obtained for each method with an instant-runoff voting mechanism on 200 of the multiclass datasets with -1/0 encoding and no *baseline*, where each dataset ranks hyperparameter choices according to the difference between significant wins and losses against all other choices (the vote of each dataset is divided by the number of tied choices ranked first). Table 8 shows optimized choices of hyperparameters for different settings of encodings and *baseline* used in our study.

Name	Method	Hyperparameters	Reduction	Baseline
G	Greedy	-	**IWR**	Y/**N**
R/RO	RegCB-elim/RegCB-opt	$C_0 \in 10^{-\{1,2,\mathbf{3}\}}$	-	Y/**N**
C-nu	Cover-NU	$N \in \{\mathbf{4}, 8, 16\}$ $\psi \in \{0.01, \mathbf{0.1}, 1\}$	IPS/**DR**	Y/**N**
C-u	Cover	$N \in \{\mathbf{4}, 8, 16\}$ $\psi \in \{0.01, \mathbf{0.1}, 1\}$	**IPS**/DR	Y/**N**
B/B-g	Bag/Bag-greedy	$N \in \{\mathbf{4}, 8, 16\}$	IPS/DR/**IWR**	Y/**N**
ϵG	ϵ-greedy	$\epsilon \in \{\mathbf{0.02}, 0.05, 0.1\}$	IPS/DR/**IWR**	Y/**N**
A	active ϵ-greedy	$\epsilon \in \{\mathbf{0.02}, 1\}$ $C_0 \in 10^{-\{2,4,\mathbf{6}\}}$	IPS/DR/**IWR**	Y/**N**

Table 8: Optimized choices of hyperparameters for different settings of encodings and *baseline*, obtained using the voting mechanism described in Table 7: -1/0 and no *baseline* (same as bold choices in Table 7, used in Tables 1left, 9a 10a as well as in the figures); 0/1 and no *baseline* (used in Tables 2, 9b, 6b, 11); -1/0 with *baseline* optimized per dataset (used in Table 13).

Algorithm	-1/0, no *baseline*	0/1, no *baseline*	-1/0, optimized *baseline*
G	-	-	-
R/RO	$C_0 = 10^{-3}$	$C_0 = 10^{-3}$	$C_0 = 10^{-3}$
C-nu	$N = 4, \psi = 0.1$, DR	$N = 4, \psi = 0.01$, DR	$N = 16, \psi = 0.1$, DR
C-u	$N = 4, \psi = 0.1$, IPS	$N = 4, \psi = 0.1$, DR	$N = 4, \psi = 0.1$, IPS
B	$N = 4$, IWR	$N = 16$, IWR	$N = 4$, IWR
B-g	$N = 4$, IWR	$N = 8$, IWR	$N = 4$, IWR
ϵG	$\epsilon = 0.02$, IWR	$\epsilon = 0.02$, IWR	$\epsilon = 0.02$, IWR
A	$\epsilon = 0.02, C_0 = 10^{-6}$, IWR	$\epsilon = 0.02, C_0 = 10^{-6}$, IWR	$\epsilon = 0.02, C_0 = 10^{-6}$, IWR

C.2 Additional Evaluation Results

This sections provides additional experimental results, and more detailed win/loss statistics for tables in the main paper, showing both significant wins and significant losses, rather than just their difference.

Figure 3 is an extended version of Figure 1, showing pairwise comparisons among our main exploration methods. Note that bagging is outperformed by other methods in most cases, likely because of the additional variance induced by the bootstrap sampling, but nevertheless provides a good exploration approach with significant computational benefits when using the IWR reduction with very large numbers of actions.

Tables 9 and 10 are extended versions of Table 1, showing both significant wins and loss, both -1/0 and 0/1 encodings, more methods, and separate statistics for multiclass and multilabel datasets. In particular, we can see that both variants of RegCB become even more competitive against all other methods when using 0/1 encodings. Table 11 extends Table 2 with additional methods. Table 12 is a more detailed win/loss version of Table 3, and additionally shows statistics for 0/1 encodings.

Figure 4 shows that *baseline* can significantly help on some datasets with -1/0 encodings for Greedy and RegCB-opt, as a way to mitigate loss estimates that may initially be pessimistic, while it mostly degrades performance for Cover-NU. Table 13 shows that optimizing over the use of *baseline* on each dataset can improve the performance of Greedy and RegCB-opt when compared to other methods such as Cover-NU.

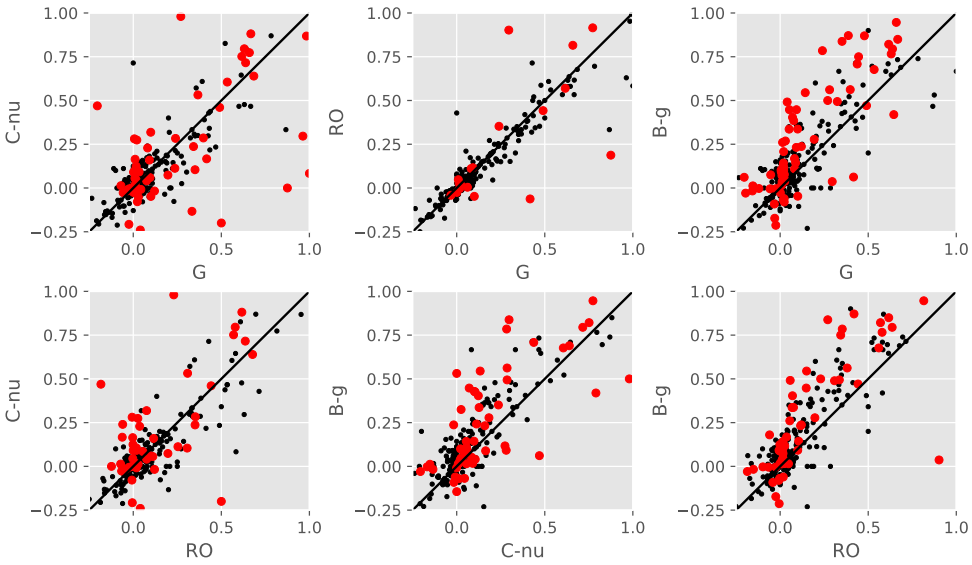


Figure 3: Pairwise comparisons among four successful methods: Greedy, Cover-nu, Bag-greedy, and RegCB-opt. Hyperparameters fixed as in Table 7. All held-out multiclass and multilabel datasets are shown, in contrast to Figure 1, which only shows held-out datasets with 5 or more actions. The plots consider normalized loss, with red points indicating significant wins.

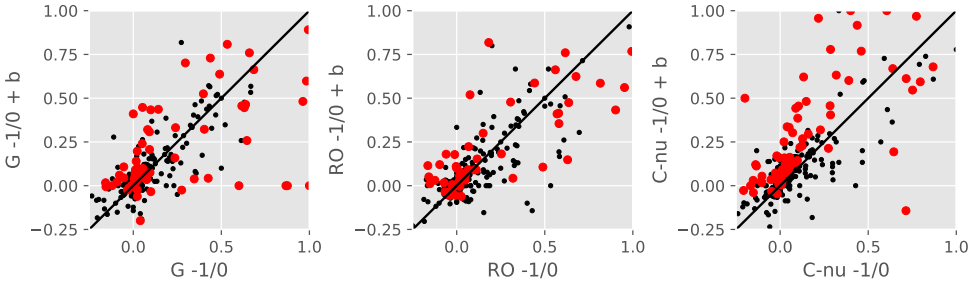


Figure 4: Impact of *baseline* on different algorithms with -1/0 encodings. For Greedy and RegCB-opt, it can significantly help against pessimistic initial costs in some datasets. Hyperparameters fixed as in Table 7. Encoding fixed to -1/0. The plots consider normalized loss on held-out datasets, with red points indicating significant wins.

Figure 5 shows a comparison between RegCB-opt and Greedy or Cover-NU on our corpus, for different values of C_0, which controls the level of exploration through the width of confidence bounds. Figure 6 shows the improvements that the active ϵ-greedy algorithm can achieve compared to ϵ-greedy, under different settings.

Table 9: *Statistically significant* wins / losses of all methods on the 324 held-out multiclass classification datasets. Hyperparameters are fixed as given in Table 8, with no *baseline*.

↓ vs →	G	R	RO	C-nu	B	B-g	ϵG	C-u	A
G	-	22 / 25	8 / 16	40 / 33	76 / 15	65 / 16	58 / 6	158 / 10	33 / 10
R	25 / 22	-	16 / 23	48 / 31	73 / 19	61 / 20	69 / 12	158 / 11	48 / 15
RO	16 / 8	23 / 16	-	46 / 21	75 / 13	59 / 13	70 / 4	165 / 6	46 / 7
C-nu	33 / 40	31 / 48	21 / 46	-	66 / 26	51 / 31	74 / 19	158 / 10	50 / 28
B	15 / 76	19 / 73	13 / 75	26 / 66	-	11 / 32	45 / 41	125 / 16	26 / 54
B-g	16 / 65	20 / 61	13 / 59	31 / 51	32 / 11	-	48 / 31	129 / 11	27 / 42
ϵG	6 / 58	12 / 69	4 / 70	19 / 74	41 / 45	31 / 48	-	124 / 14	2 / 36
C-u	10 / 158	11 / 158	6 / 165	10 / 158	16 / 125	11 / 129	14 / 124	-	9 / 151
A	10 / 33	15 / 48	7 / 46	28 / 50	54 / 26	42 / 27	36 / 2	151 / 9	-

(a) -1/0 encoding

↓ vs →	G	R	RO	C-nu	B	B-g	ϵG	C-u	A
G	-	27 / 66	5 / 69	35 / 51	74 / 38	71 / 39	68 / 21	150 / 30	36 / 19
R	66 / 27	-	15 / 38	41 / 24	86 / 12	82 / 16	107 / 19	171 / 4	77 / 22
RO	69 / 5	38 / 15	-	58 / 13	108 / 7	104 / 9	120 / 3	174 / 3	89 / 4
C-nu	51 / 35	24 / 41	13 / 58	-	83 / 27	72 / 29	95 / 25	169 / 6	63 / 31
B	38 / 74	12 / 86	7 / 108	27 / 83	-	21 / 34	58 / 49	130 / 14	37 / 67
B-g	39 / 71	16 / 82	9 / 104	29 / 72	34 / 21	-	64 / 46	128 / 18	42 / 54
ϵG	21 / 68	19 / 107	3 / 120	25 / 95	49 / 58	46 / 64	-	121 / 28	3 / 42
C-u	30 / 150	4 / 171	3 / 174	6 / 169	14 / 130	18 / 128	28 / 121	-	22 / 150
A	19 / 36	22 / 77	4 / 89	31 / 63	67 / 37	54 / 42	42 / 3	150 / 22	-

(b) 0/1 encoding

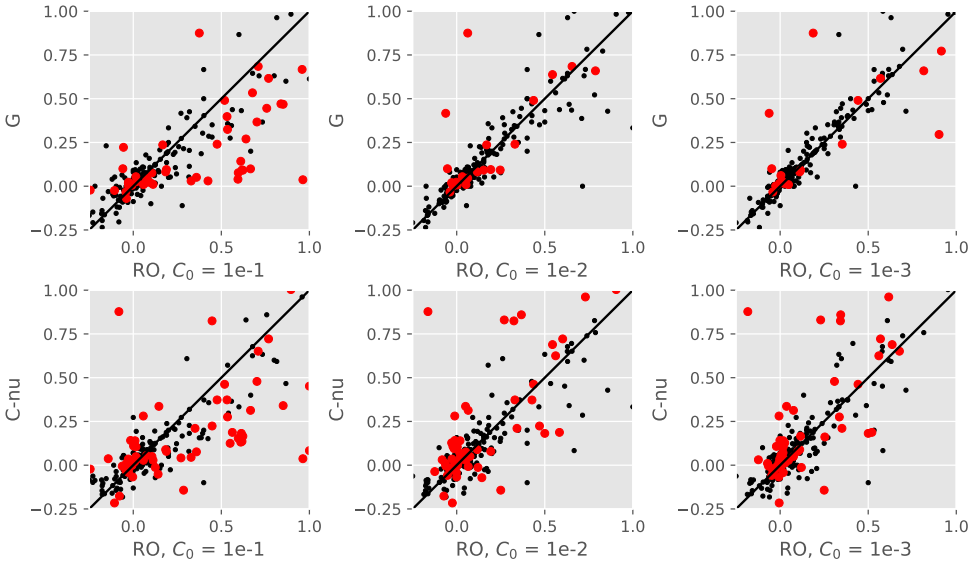


Figure 5: Comparison of RegCB-opt with Greedy (top) and Cover-NU (bottom) for different values of C_0. Hyperparameters for Greedy and Cover-NU fixed as in Table 7. Encoding fixed to -1/0. The plots consider normalized loss on held-out datasets, with red points indicating significant wins.

Table 10: *Statistically significant* wins / losses of all methods on the 5 multilabel classification datasets. Hyperparameters are fixed as given in Table 8, with no *baseline*.

↓ vs →	G	R	RO	C-nu	B	B-g	ϵG	C-u	A
G	-	2 / 2	1 / 0	3 / 1	2 / 2	3 / 2	2 / 1	4 / 1	2 / 1
R	2 / 2	-	2 / 2	2 / 0	3 / 0	3 / 0	3 / 2	5 / 0	3 / 2
RO	0 / 1	2 / 2	-	2 / 2	2 / 2	3 / 1	2 / 1	4 / 1	2 / 1
C-nu	1 / 3	0 / 2	2 / 2	-	3 / 1	3 / 1	3 / 2	5 / 0	3 / 2
B	2 / 2	0 / 3	2 / 2	1 / 3	-	1 / 1	3 / 2	5 / 0	3 / 2
B-g	2 / 3	0 / 3	1 / 3	1 / 3	1 / 1	-	2 / 3	4 / 1	2 / 3
ϵG	1 / 2	2 / 3	1 / 2	2 / 3	2 / 3	3 / 2	-	4 / 1	1 / 0
C-u	1 / 4	0 / 5	1 / 4	0 / 5	0 / 5	1 / 4	1 / 4	-	1 / 4
A	1 / 2	2 / 3	1 / 2	2 / 3	2 / 3	3 / 2	0 / 1	4 / 1	-

(a) -1/0 encoding

↓ vs →	G	R	RO	C-nu	B	B-g	ϵG	C-u	A
G	-	3 / 1	2 / 2	1 / 3	4 / 0	4 / 1	4 / 0	5 / 0	3 / 0
R	1 / 3	-	0 / 3	1 / 4	2 / 2	2 / 3	2 / 2	4 / 1	2 / 2
RO	2 / 2	3 / 0	-	1 / 2	5 / 0	4 / 0	3 / 1	5 / 0	3 / 1
C-nu	3 / 1	4 / 1	2 / 1	-	4 / 1	3 / 1	4 / 0	4 / 1	4 / 1
B	0 / 4	2 / 2	0 / 5	1 / 4	-	0 / 2	1 / 2	2 / 1	1 / 3
B-g	1 / 4	3 / 2	0 / 4	1 / 3	2 / 0	-	2 / 2	4 / 1	1 / 3
ϵG	0 / 4	2 / 2	1 / 3	0 / 4	2 / 1	2 / 2	-	4 / 1	0 / 1
C-u	0 / 5	1 / 4	0 / 5	1 / 4	1 / 2	1 / 4	1 / 4	-	0 / 5
A	0 / 3	2 / 2	1 / 3	1 / 4	3 / 1	3 / 1	1 / 0	5 / 0	-

(b) 0/1 encoding

Table 11: Progressive validation loss for RCV1 with real-valued costs. Same as Table 2, but with all methods. Hyperparameters are fixed as given in Table 8, with no *baseline*. The learning rate is optimized once on the original dataset, and we show mean and standard error based on 10 different random reshufflings of the dataset.

G	R	RO	C-nu	C-u
0.215 ± 0.010	0.408 ± 0.003	0.225 ± 0.008	0.215 ± 0.006	0.570 ± 0.023

B	B-g	ϵG	A
0.256 ± 0.006	0.251 ± 0.005	0.230 ± 0.009	0.230 ± 0.010

Table 12: Impact of reductions for Bag (left) and ϵ-greedy (right), with hyperparameters optimized and encoding fixed to -1/0 or 0/1. Extended version of Table 3. Each (row, column) entry shows the *statistically significant* wins and losses of row against column.

↓ vs →	ips	dr	iwr
ips	-	30 / 72	25 / 83
dr	72 / 30	-	30 / 57
iwr	83 / 25	57 / 30	-

↓ vs →	ips	dr	iwr
ips	-	85 / 22	16 / 148
dr	22 / 85	-	9 / 163
iwr	148 / 16	163 / 9	-

(a) -1/0 encoding

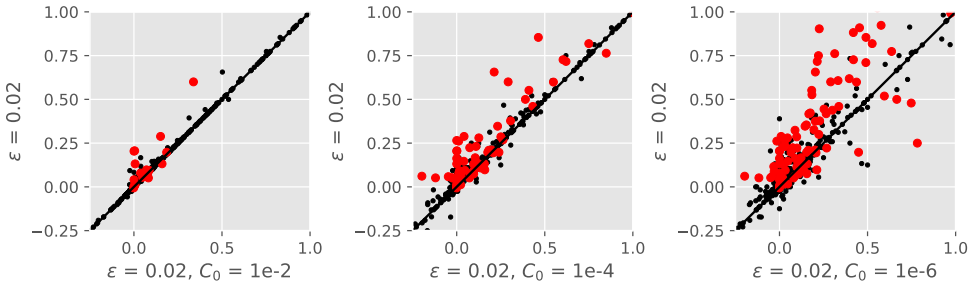
↓ vs →	ips	dr	iwr
ips	-	46 / 239	17 / 244
dr	239 / 46	-	33 / 96
iwr	244 / 17	96 / 33	-

↓ vs →	ips	dr	iwr
ips	-	40 / 135	36 / 181
dr	135 / 40	-	23 / 147
iwr	181 / 36	147 / 23	-

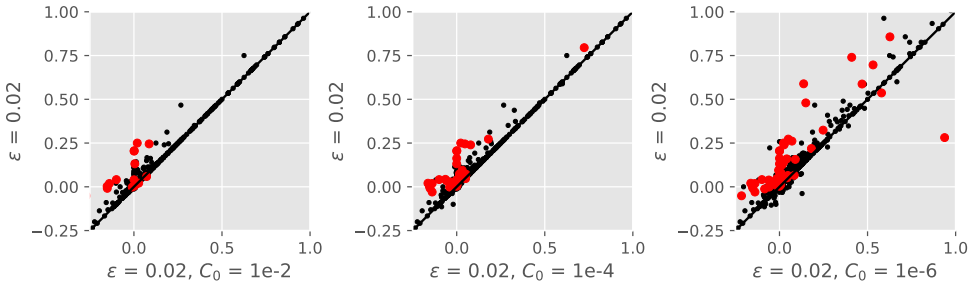
(b) 0/1 encoding

Table 13: *Statistically significant* wins / losses of all methods on held-out datasets, with -1/0 encoding and fixed hyperparameters, except for *baseline*, which is optimized on each dataset together with the learning rate. This optimization benefits Greedy and RegCB-opt in particular.

↓ vs →	G	R	RO	C-nu	B	B-g	ϵG	C-u	A
G	-	32 / 14	14 / 16	60 / 15	87 / 13	69 / 16	77 / 1	177 / 1	49 / 6
R	14 / 32	-	9 / 36	43 / 25	68 / 19	52 / 24	65 / 14	166 / 9	42 / 19
RO	16 / 14	36 / 9	-	63 / 13	89 / 10	65 / 10	83 / 3	186 / 1	61 / 5
C-nu	15 / 60	25 / 43	13 / 63	-	58 / 30	34 / 41	59 / 24	162 / 6	42 / 36
B	13 / 87	19 / 68	10 / 89	30 / 58	-	10 / 33	53 / 35	132 / 7	29 / 59
B-g	16 / 69	24 / 52	10 / 65	41 / 34	33 / 10	-	56 / 18	146 / 2	37 / 38
ϵG	1 / 77	14 / 65	3 / 83	24 / 59	35 / 53	18 / 56	-	128 / 10	3 / 43
C-u	1 / 177	9 / 166	1 / 186	6 / 162	7 / 132	2 / 146	10 / 128	-	5 / 163
A	6 / 49	19 / 42	5 / 61	36 / 42	59 / 29	38 / 37	43 / 3	163 / 5	-



(a) DR



(b) IWR

Figure 6: Improvements to ϵ-greedy from our active learning strategy. Encoding fixed to -1/0. The IWR implementation described in Section D.1 still manages to often outperform ϵ-greedy, despite only providing an approximation to Algorithm 6.

Algorithm 6 Active ϵ-greedy

π_1; ϵ; $C_0 > 0$.
`explore`(x_t):
  $A_t = \{a : \texttt{loss\_diff}(\pi_t, x_t, a) \leq \Delta_{t,C_0}\}$;
  $p_t(a) = \frac{\epsilon}{K}\,\mathbb{1}\{a \in A_t\} + (1 - \frac{\epsilon|A_t|}{K})\,\mathbb{1}\{\pi_t(x_t) = a\}$;
  return p_t;
`learn`$(x_t, a_t, \ell_t(a_t), p_t)$:
  $\hat{\ell}_t = \texttt{estimator}(x_t, a_t, \ell_t(a_t), p_t(a_t))$;
  $\hat{c}_t(a) = \begin{cases} \hat{\ell}_t(a), & \text{if } p_t(a) > 0 \\ 1, & \text{otherwise.} \end{cases}$
  $\pi_{t+1} = \texttt{csc\_oracle}(\pi_t, x_t, \hat{c}_t)$;

D Active ϵ-greedy: Practical Algorithm and Analysis

This section presents our active ϵ-greedy method, a variant of ϵ-greedy that reduces the amount of uniform exploration using techniques from active learning. Section D.1 introduces the practical algorithm, while Section D.2 provides a theoretical analysis of the method, showing that it achieves a regret of $O(T^{1/3})$ in favorable settings, compared to $O(T^{2/3})$ for vanilla ϵ-greedy.

D.1 Algorithm

The simplicity of the ϵ-greedy method described in Section A.1 often makes it the method of choice for practitioners. However, the uniform exploration over randomly selected actions can be quite inefficient and costly in practice. A natural consideration is to restrict this randomization over actions which could plausibly be selected by the optimal policy $\pi^* = \arg\min_{\pi \in \Pi} L(\pi)$, where $L(\pi) = \mathbb{E}_{(x,\ell)\sim D}[\ell(\pi(x))]$ is the expected loss of a policy π.

To achieve this, we use techniques from disagreement-based active learning [15, 17]. After observing a context x_t, for any action a, if we can find a policy π that would choose this action ($\pi(x_t) = a$) instead of the empirically best action $\pi_t(x_t)$, while achieving a small loss on past data, then there is disagreement about how good such an action is, and we allow exploring it. Otherwise, we are confident that the best policy would not choose this action, thus we avoid exploring it, and assign it a high cost. The resulting method is in Algorithm 6. Like RegCB, the method requires a known loss range $[c_{min}, c_{max}]$, and assigns a loss c_{max} to such unexplored actions (we consider the range $[0, 1]$ in Algorithm 6 for simplicity). The disagreement test we use is based on empirical loss differences, similar to the Oracular CAL active learning method [17], denoted `loss_diff`, together with a threshold:

$$\Delta_{t,C_0} = \sqrt{C_0 \frac{K \log t}{\epsilon t}} + C_0 \frac{K \log t}{\epsilon t}.$$

A practical implementation of `loss_diff` for an online setting is given below. We analyze a theoretical form of this algorithm in Section D.2, showing a formal version of the following theorem:

Theorem 1. *With high-probability, and under favorable conditions on disagreement and on the problem noise, active ϵ-greedy achieves expected regret $O(T^{1/3})$.*

Note that this data-dependent guarantee improves on worst-case guarantees achieved by the optimal algorithms in [4, 11]. In the extreme case where the loss of any suboptimal policy is bounded away from that of π^*, we show that our algorithm can achieve constant regret. While active learning algorithms suggest that data-dependent thresholds Δ_t can yield better guarantees (*e.g.*, [18], this may require more work in our setting due to open problems related to data-dependent guarantees for contextual bandits [5]. In a worst-case scenario, active ϵ-greedy behaves similarly to ϵ-greedy [24], achieving an $O(T^{2/3})$ expected regret with high probability.

Practical implementation of the disagreement test. We now present a practical way to implement the disagreement tests in the active ϵ-greedy method, in the context of online cost-sensitive classification oracles based on regression, as in Vowpal Wabbit. This corresponds to the `loss_diff` method in Algorithm 6.

Let $\hat{L}_{t-1}(\pi)$ denote the empirical loss of policy π on the (biased) sample of cost-sensitive examples collected up to time $t-1$ (see Section D.2 for details). After observing a context x_t, we want to estimate

$$\texttt{loss\_diff}(\pi_t, x_t, \bar{a}) \approx \hat{L}_{t-1}(\pi_{t,\bar{a}}) - \hat{L}_{t-1}(\pi_t),$$

for any action $\bar{a}$, where

$$\pi_t = \arg\min_{\pi} \hat{L}_{t-1}(\pi)$$
$$\pi_{t,\bar{a}} = \arg\min_{\pi:\pi(x_t)=\bar{a}} \hat{L}_{t-1}(\pi).$$

In our online setup, we take π_t to be the current online policy (as in Algorithm 6), and we estimate the loss difference by looking at how many online CSC examples of the form $\bar{c} := (\mathbb{1}\{a \neq \bar{a}\})_{a=1..K}$ are needed (or the importance weight on such an example) in order to switch prediction from $\pi_t(x_t)$ to $\bar{a}$. If we denote this importance weight by $\tau_{\bar{a}}$, then we can estimate $\hat{L}_{t-1}(\pi_{t,\bar{a}}) - \hat{L}_{t-1}(\pi_t) \approx \tau_{\bar{a}}/t$.

Computing $\tau_{\bar{a}}$ for IPS/DR. In the case of IPS/DR, we use an online CSC oracle, which is based on K regressors $f(x,a)$ in VW, each predicting the cost for an action a. Let f_t be the current regressors for policy π_t, $y_t(a) := f_t(x_t, a)$, and denote by $s_t(a)$ the *sensitivity* of regressor $f_t(\cdot, a)$ on example $(x_t, \bar{c}(a))$. This sensitivity is essentially defined to be the derivative with respect to an importance weight w of the prediction $y'(a)$ obtained from the regressor after an online update $(x_t, \bar{c}(a))$ with importance weight w. A similar quantity has been used, *e.g.*, in [18, 22, 23]. Then, the predictions on actions $\bar{a}$ and a cross when the importance weight w satisfies $y_t(\bar{a}) - s_t(\bar{a})w = y_t(a) + s_t(a)w$. Thus, the importance weight required for action $\bar{a}$ to be preferred (*i.e.*, smaller predicted loss) to action a is given by:

$$w_{\bar{a}}^{a} = \frac{y_t(\bar{a}) - y_t(a)}{s_t(\bar{a}) + s_t(a)}.$$

Action $\bar{a}$ will thus be preferred to all other actions when using an importance weight $\tau_{\bar{a}} = \max_a w_{\bar{a}}^{a}$.

Computing $\tau_{\bar{a}}$ for IWR. Although Algorithm 6 and the theoretical analysis require CSC in order to assign a loss of 1 to unexplored actions, and hence does not directly support IWR, we can consider an approximation which leverages the benefits of IWR by performing standard IWR updates as in ϵ-greedy, while exploring only on actions that pass a similar disagreement test. In this case, we estimate $\tau_{\bar{a}}$ as the importance weight on an online regression example $(x_t, 0)$ for the regressor $f_t(\cdot, \bar{a})$, needed to switch prediction to $\bar{a}$. If $s_t(\bar{a})$ is the sensitivity for such an example, we have $\tau_{\bar{a}} = (y_t(\bar{a}) - y_t^*)/s_t(\bar{a})$, where $y_t^* = \min_a y_t(a)$.

D.2 Theoretical Analysis

This section presents a theoretical analysis of the active ϵ-greedy method introduced in Section D.1. We begin by presenting the analyzed version of the algorithm together with definitions in Section D.2.1. Section D.2.2 then studies the correctness of the method, showing that with high probability, the actions chosen by the optimal policy are always explored, and that policies considered by the algorithm are always as good as those obtained under standard ϵ-greedy exploration. This section also introduces a Massart-type low-noise condition similar to the one considered in [23] for cost-sensitive classification. Finally, Section D.2.3 provides a regret analysis of the algorithm, both in the worst case and under disagreement conditions together with the Massart noise condition. In particular, a formal version of Theorem 1 is given by Theorem 8, and a more extreme but informative situation is considered in Proposition 9, where our algorithm can achieve constant regret.

D.2.1 Algorithm and definitions

We consider a version of the active ϵ-greedy strategy that is more suitable for theoretical analysis, given in Algorithm 7. This method considers exact CSC oracles, as well as a CSC oracle with one constraint on the policy ($\pi(x_t) = a$ in Eq.(9)). The threshold Δ_t is defined later in Section D.2.2. Computing it would require some knowledge about the size of the policy class, which we avoid by introducing a parameter C_0 in the practical variant. The disagreement strategy is based on the Oracular CAL active learning method of Hsu [17], which tests for disagreement using empirical error differences, and considers biased samples when no label is queried. Here, similar tests are used

Algorithm 7 active ϵ-greedy: analyzed version

Input: exploration probability ϵ.
Initialize: $\hat{Z}_0 := \emptyset$.
for $t = 1, \ldots$ **do**
Observe context x_t. Let

$$\pi_t := \arg\min_{\pi} L(\pi, \hat{Z}_{t-1})$$

$$\pi_{t,a} := \arg\min_{\pi:\pi(x_t)=a} L(\pi, \hat{Z}_{t-1}) \tag{9}$$

$$A_t := \{a : L(\pi_{t,a}, \hat{Z}_{t-1}) - L(\pi_t, \hat{Z}_{t-1}) \leq \Delta_t\} \tag{10}$$

Let

$$p_t(a) = \begin{cases} 1 - (|A_t| - 1)\epsilon/K, & \text{if } a = \pi_t(x_t) \\ \epsilon/K, & \text{if } a \in A_t \setminus \{\pi_t(x_t)\} \\ 0, & \text{otherwise.} \end{cases}$$

Play action $a_t \sim p_t$, observe $\ell_t(a_t)$ and set $\hat{Z}_t = \hat{Z}_{t-1} \cup \{(x_t, \hat{\ell}_t)\}$, where $\hat{\ell}_t$ is defined in (8).
end for

to decide which actions should be explored, in the different context of cost-sensitive classification, and the unexplored actions are assigned a loss of 1, making the empirical sample biased ($\hat{Z}_T$ in Algorithm 7).

Definitions. Define $Z_T = \{(x_t, \ell_t)\}_{t=1..T} \subset \mathcal{X} \times \mathbb{R}^K$, $\tilde{Z}_T = \{(x_t, \tilde{\ell}_t)\}_{t=1..T}$ (biased sample) and $\hat{Z}_T = \{(x_t, \hat{\ell}_t)\}_{t=1..T}$ (IPS estimate of biased sample), where $\ell_t \in [0,1]^K$ is the (unobserved) loss vector at time t and

$$\tilde{\ell}_t(a) = \begin{cases} \ell_t(a), & \text{if } a \in A_t \\ 1, & \text{o/w} \end{cases} \tag{7}$$

$$\hat{\ell}_t(a) = \begin{cases} \frac{\mathbb{1}\{a=a_t\}}{p_t(a_t)} \ell_t(a_t), & \text{if } a \in A_t \\ 1, & \text{o/w.} \end{cases} \tag{8}$$

For any set $Z \subset \mathcal{X} \times \mathbb{R}^K$ defined as above, we denote, for $\pi \in \Pi$,

$$L(\pi, Z) = \frac{1}{|Z|} \sum_{(x,c) \in Z} c(\pi(x)).$$

We then define the empirical losses $L_T(\pi) := L(\pi, Z_T)$, $\hat{L}_T(\pi) := L(\pi, \hat{Z}_T)$ and $\tilde{L}_T(\pi) := L(\pi, \tilde{Z}_T)$. Let $L(\pi) := \mathbb{E}_{(x,\ell)\sim D}[\ell(\pi(x))]$ be the expected loss of policy π, and $\pi^* := \arg\min_{\pi \in \Pi} L(\pi)$. We also define $\rho(\pi, \pi') := P_x(\pi(x) \neq \pi'(x))$, the expected disagreement between policies π and π', where P_x denotes the marginal distribution of D on contexts.

D.2.2 Correctness

We begin by stating a lemma that controls deviations of empirical loss differences, which relies on Freedman's inequality for martingales (see, *e.g.*, [20, Lemma 3]).

Lemma 2 (Deviation bounds)**.** *With probability $1 - \delta$, the following event holds: for all $\pi \in \Pi$, for all $T \geq 1$,*

$$|(\hat{L}_T(\pi) - \hat{L}_T(\pi^*)) - (\tilde{L}_T(\pi) - \tilde{L}_T(\pi^*))| \leq \sqrt{\frac{2K\rho(\pi, \pi^*)e_T}{\epsilon}} + \left(\frac{K}{\epsilon} + 1\right) e_T \tag{11}$$

$$|(L_T(\pi) - L_T(\pi^*)) - (L(\pi) - L(\pi^*))| \leq \sqrt{\rho(\pi, \pi^*)e_T} + 2e_T, \tag{12}$$

where $e_T = \log(2|\Pi|/\delta_T)/T$ and $\delta_T = \delta/(T^2 + T)$. We denote this event by $\mathcal{E}$ in what follows.

Proof. We prove the result using Freedman's inequality (see, *e.g.*, [20, Lemma 3]), which controls deviations of a sum using the conditional variance of each term in the sum and an almost sure bound on their magnitude, along with a union bound.

For (11), let $(\hat{L}_T(\pi) - \hat{L}_T(\pi^*)) - (\tilde{L}_T(\pi) - \tilde{L}_T(\pi^*)) = \frac{1}{T}\sum_{t=1}^T R_t$, with

$$R_t = \hat{\ell}_t(\pi(x_t)) - \hat{\ell}_t(\pi^*(x_t)) - (\tilde{\ell}_t(\pi(x_t)) - \tilde{\ell}_t(\pi^*(x_t))).$$

We define the σ-fields $\mathcal{F}_t := \sigma(\{x_i, \ell_i, a_i\}_{i=1}^t)$. Note that R_t is $\mathcal{F}_t$-measurable and

$$\mathbb{E}[\hat{\ell}_t(\pi(x_t)) - \hat{\ell}_t(\pi^*(x_t))|x_t, \ell_t] = \tilde{\ell}_t(\pi(x_t)) - \tilde{\ell}_t(\pi^*(x_t)),$$

so that $\mathbb{E}[R_t|\mathcal{F}_{t-1}] = \mathbb{E}[\mathbb{E}[R_t|x_t, \ell_t]|\mathcal{F}_{t-1}] = 0$. Thus, $(R_t)_{t\geq 1}$ is a martingale difference sequence adapted to the filtration $(\mathcal{F}_t)_{t\geq 1}$. We have

$$|R_t| \leq |\hat{\ell}_t(\pi(x_t)) - \hat{\ell}_t(\pi^*(x_t))| + |\tilde{\ell}_t(\pi(x_t)) - \tilde{\ell}_t(\pi^*(x_t))| \leq \frac{K}{\epsilon} + 1.$$

Note that $\mathbb{E}[\hat{\ell}_t(\pi(x_t)) - \hat{\ell}_t(\pi^*(x_t))|x_t, \ell_t] = \tilde{\ell}_t(\pi(x_t)) - \tilde{\ell}_t(\pi^*(x_t))$, so that

$$\begin{aligned}
\mathbb{E}[R_t^2|\mathcal{F}_{t-1}] &= \mathbb{E}[\mathbb{E}[R_t^2|x_t, \ell_t, A_t]|\mathcal{F}_{t-1}] \\
&\leq \mathbb{E}[\mathbb{E}[(\hat{\ell}_t(\pi(x_t)) - \hat{\ell}_t(\pi^*(x_t)))^2|x_t, \ell_t, A_t]|\mathcal{F}_{t-1}] \\
&\leq \mathbb{E}\left[\mathbb{E}\left[\frac{(\mathbb{1}\{\pi(x_t) = a_t\} - \mathbb{1}\{\pi^*(x_t) = a_t\})^2}{p_t(a_t)^2}|x_t, \ell_t, A_t\right]|\mathcal{F}_{t-1}\right] \\
&\leq \mathbb{E}\left[\mathbb{E}\left[\frac{\mathbb{1}\{\pi(x_t) \neq \pi^*(x_t)\}(\mathbb{1}\{\pi(x_t) = a_t\} + \mathbb{1}\{\pi^*(x_t) = a_t\})}{p_t(a_t)^2}|x_t, \ell_t, A_t\right]|\mathcal{F}_{t-1}\right] \\
&= \mathbb{E}\left[\frac{2K\ \mathbb{1}\{\pi(x_t) \neq \pi^*(x_t)\}}{\epsilon}|\mathcal{F}_{t-1}\right] = \frac{2K}{\epsilon}\rho(\pi, \pi^*).
\end{aligned}$$

Freedman's inequality then states that (11) holds with probability $1 - \delta_T/2|\Pi|$.

For (12), we consider a similar setup with

$$R_t = \ell_t(\pi(x_t)) - \ell_t(\pi^*(x_t)) - (L(\pi) - L(\pi^*)).$$

We have $\mathbb{E}[R_t|\mathcal{F}_{t-1}] = 0$, $|R_t| \leq 2$ and $\mathbb{E}[R_t^2|\mathcal{F}_{t-1}] \leq \rho(\pi, \pi^*)$, which yields that (12) holds with probability $1 - \delta_T/2|\Pi|$ using Freedman's inequality. A union bound on $\pi \in \Pi$ and $T \geq 1$ gives the desired result. □

Threshold. We define the threshold Δ_T used in (10) in Algorithm 7 as:

$$\Delta_T := \left(\sqrt{\frac{2K}{\epsilon}} + 1\right)\sqrt{e_{T-1}} + \left(\frac{K}{\epsilon} + 3\right)e_{T-1}. \tag{13}$$

We also define the following more precise deviation quantity for a given policy, which follows directly from the deviation bounds in Lemma 2

$$\Delta_T^*(\pi) := \left(\sqrt{\frac{2K}{\epsilon}} + 1\right)\sqrt{\rho(\pi, \pi^*)e_{T-1}} + \left(\frac{K}{\epsilon} + 3\right)e_{T-1}. \tag{14}$$

Note that we have $\Delta_T^*(\pi) \leq \Delta_T$ for any policy π.

The next lemma shows that the bias introduced in the empirical sample by assigning a loss of 1 to unexplored actions is favorable, in the sense that it will not hurt us in identifying π^*.

Lemma 3 (Favorable bias)**.** *Assume $\pi^*(x_t) \in A_t$ for all $t \leq T$. We have*

$$\tilde{L}_T(\pi) - \tilde{L}_T(\pi^*) \geq L_T(\pi) - L_T(\pi^*). \tag{15}$$

Proof. For any $t \leq T$, we have $\tilde{\ell}_t(a) \geq \ell_t(a)$, so that $\tilde{L}_T(\pi) \geq L_T(\pi)$. Separately, we have $\tilde{\ell}_t(\pi^*(x_t)) = \ell_t(\pi^*(x_t))$ for all $t \leq T$ using the definition of $\tilde{\ell}_t$ and the assumption $\pi^*(x_t) \in A_t$, hence $\tilde{L}_T(\pi^*) \geq L_T(\pi^*)$. □

We now show that with high probability, the optimal action is always explored by the algorithm.

Lemma 4. *Assume that event $\mathcal{E}$ holds. The actions given by the optimal policy are always explored for all $t \geq 1$, i.e., $\pi^*(x_t) \in A_t$ for all $t \geq 1$.*

Proof. We show by induction on $T \geq 1$ that $\pi^*(x_t) \in A_t$ for all $t = 1, \dots, T$. For the base case, we have $A_1 = [K]$ since $\hat{Z}_0 = \emptyset$ and hence empirical errors are always equal to 0, so that $\pi^*(x_1) \in A_1$. Let us now assume as the inductive hypothesis that $\pi^*(x_t) \in A_t$ for all $t \leq T-1$.

From deviation bounds, we have

$$\hat{L}_{T-1}(\pi_T) - \hat{L}_{T-1}(\pi^*) \geq \tilde{L}_{T-1}(\pi_T) - \tilde{L}_{T-1}(\pi^*) - \left(\sqrt{\frac{2K\rho(\pi, \pi^*)e_{T-1}}{\epsilon}} + (K/\epsilon + 1)e_{T-1} \right)$$

$$L_{T-1}(\pi_T) - L_{T-1}(\pi^*) \geq L(\pi_T) - L(\pi^*) - \left(\sqrt{\rho(\pi, \pi^*)e_{T-1}} + 2e_{T-1} \right).$$

Using Lemma 3 together with the inductive hypothesis, the above inequalities yield

$$\hat{L}_{T-1}(\pi_T) - \hat{L}_{T-1}(\pi^*) \geq L(\pi_T) - L(\pi^*) - \Delta^*_T(\pi_T).$$

Now consider an action $a \notin A_t$. Using the definition (10) of A_t, we have

$$\begin{aligned} \hat{L}_{T-1}(\pi_{T,a}) - \hat{L}_{T-1}(\pi^*) &= \hat{L}_{T-1}(\pi_{T,a}) - \hat{L}_{T-1}(\pi_T) + \hat{L}_{T-1}(\pi_T) - \hat{L}_{T-1}(\pi^*) \\ &> \Delta_T - \Delta^*_T(\pi_T) = 0, \end{aligned}$$

which implies $\pi^*(x_T) \neq a$, since $\hat{L}_{T-1}(\pi_{T,a})$ is the minimum of $\hat{L}_{T-1}$ over policies satisfying $\pi(x_T) = a$. This yields $\pi^*(x_T) \in A_T$, which concludes the proof. □

With the previous results, we can now prove that with high probability, discarding some of the actions from the exploration process does not hurt us in identifying good policies. In particular, π_{T+1} is about as good as it would have been with uniform ϵ-exploration all along.

Theorem 5. *Under the event $\mathcal{E}$, which holds with probability $1 - \delta$,*

$$L(\pi_{T+1}) - L(\pi^*) \leq \Delta^*_{T+1}(\pi_{T+1}).$$

In particular, $L(\pi_{T+1}) - L(\pi^*) \leq \Delta_{T+1}$.

Proof. Assume event $\mathcal{E}$ holds. Using (11-12) combined with Lemma 3 (which holds by Lemma 4), we have

$$L(\pi_{T+1}) - L(\pi^*) \leq \hat{L}_T(\pi_{T+1}) - \hat{L}_T(\pi^*) + \Delta^*_{T+1}(\pi_{T+1}) \leq \Delta^*_{T+1}(\pi_{T+1}).$$

□

Massart noise condition. We introduce a low-noise condition that will help us obtain improved regret guarantees. Similar conditions have been frequently used in supervised learning [26] and active learning [17, 18, 23] for obtaining better data-dependent guarantees. We consider the following Massart noise condition with parameter $\tau > 0$:

$$\rho(\pi, \pi^*) \leq \frac{1}{\tau}(L(\pi) - L(\pi^*)). \tag{M}$$

This condition holds when $\mathbb{E}[\min_{a \neq \pi^*(x)} \ell(a) - \ell(\pi^*(x))|x] \geq \tau$, P_x-almost surely, which is similar to the Massart condition considered in [23] in the context of active learning for cost-sensitive classification. Indeed, we have

$$\begin{aligned} L(\pi) - L(\pi^*) &= \mathbb{E}[\mathbb{1}\{\pi(x) \neq \pi^*(x)\}(\ell(\pi(x)) - \ell(\pi^*(x))] \\ &\quad + \mathbb{E}[\mathbb{1}\{\pi(x) = \pi^*(x)\}(\ell(\pi^*(x)) - \ell(\pi^*(x)))] \\ &\geq \mathbb{E}\left[\mathbb{1}\{\pi(x) \neq \pi^*(x)\} \left(\min_{a \neq \pi^*(x)} \ell(a) - \ell(\pi^*(x)) \right) \right] \\ &= \mathbb{E}[\mathbb{1}\{\pi(x) \neq \pi^*(x)\}\, \mathbb{E}[\min_{a \neq \pi^*(x)} \ell(a) - \ell(\pi^*(x))|x]] \\ &\geq \mathbb{E}[\mathbb{1}\{\pi(x) \neq \pi^*(x)\}\tau] = \tau\rho(\pi, \pi^*), \end{aligned}$$

which is precisely (M). The condition allows us to obtain a fast rate for the policies considered by our algorithm, as we now show.

Theorem 6. *Assume the Massart condition* (M) *holds with parameter τ. Under the event $\mathcal{E}$, which holds w.p. $1-\delta$,*

$$L(\pi_{T+1}) - L(\pi^*) \leq C\frac{K}{\tau\epsilon}e_T,$$

for some numeric constant C.

Proof. Using Theorem 5 and the Massart condition, we have

$$\begin{aligned}
L(\pi_{T+1}) - L(\pi^*) \leq \Delta^*_{T+1}(\pi_{T+1}) &= \left(\sqrt{\frac{2K}{\epsilon}}+1\right)\sqrt{\rho(\pi_{T+1},\pi^*)e_T} + \left(\frac{K}{\epsilon}+3\right)e_T \\
&\leq \left(\sqrt{\frac{2K}{\epsilon}}+1\right)\sqrt{(L(\pi_{T+1})-L(\pi^*))e_T/\tau} + \left(\frac{K}{\epsilon}+3\right)e_T \\
&\leq \sqrt{\frac{8Ke_T}{\tau\epsilon}(L(\pi_{T+1})-L(\pi^*))} + \frac{4Ke_T}{\epsilon}.
\end{aligned}$$

Solving the quadratic inequality in $L(\pi_{T+1}) - L(\pi^*)$ yields the result. □

D.2.3 Regret Analysis

In a worst-case scenario, the following result shows that Algorithm 7 enjoys a similar $O(T^{2/3})$ regret guarantee to the vanilla ϵ-greedy approach [24].

Theorem 7. *Conditioned on the event $\mathcal{E}$, which holds with probability $1-\delta$, the expected regret of the algorithm is*

$$\mathbb{E}[R_T|\mathcal{E}] \leq O\left(\sqrt{\frac{KT\log(T|\Pi|/\delta)}{\epsilon}} + T\epsilon\right).$$

Optimizing over the choice of ϵ yields a regret $O(T^{2/3}(K\log(T|\Pi|/\delta))^{1/3})$.

Proof. We condition on the $1-\delta$ probability event $\mathcal{E}$ that the deviation bounds of Lemma 2 hold. We have

$$\begin{aligned}
\mathbb{E}[\ell_t(a_t) - \ell_t(\pi^*(x_t))|\mathcal{F}_{t-1}] &= \mathbb{E}[\mathbb{1}\{a_t = \pi_t(x_t)\}(\ell_t(\pi_t(x_t)) - \ell_t(\pi^*(x_t)))|\mathcal{F}_{t-1}] \\
&\quad + \mathbb{E}[\mathbb{1}\{a_t \neq \pi_t(x_t)\}(\ell_t(a_t) - \ell_t(\pi^*(x_t)))|\mathcal{F}_{t-1}] \\
&\leq \mathbb{E}[\ell_t(\pi_t(x_t)) - \ell_t(\pi^*(x_t))|\mathcal{F}_{t-1}] + \mathbb{E}[\mathbb{E}[1-p_t(\pi_t(x_t))|x_t]|\mathcal{F}_{t-1}] \\
&\leq L(\pi_t) - L(\pi^*) + \epsilon.
\end{aligned}$$

Summing over t and applying Theorem 5 together with $\Delta^*_t(\pi) \leq \Delta_t$, we obtain

$$\begin{aligned}
\mathbb{E}[R_T|\mathcal{E}]\,\mathbb{E}\left[\sum_{t=1}^T \ell_t(a_t) - \ell_t(\pi^*(x_t))|\mathcal{E}\right] &\leq 1 + \sum_{t=2}^T \mathbb{E}[L(\pi_t) - L(\pi^*) + \epsilon|\mathcal{F}_{t-1},\mathcal{E}] \\
&\leq 1 + T\epsilon + \sum_{t=2}^T \Delta_t.
\end{aligned}$$

Using $\sum_{t=2}^T \sqrt{e_t} \leq O(\sqrt{T\log(8T^2|\Pi|/\delta)})$ and $\sum_{t=2}^T e_t \leq O(\log(8T^2|\Pi|/\delta)\log T)$, we obtain

$$\mathbb{E}[R_T|\mathcal{E}] \leq O\left(1 + \sqrt{\frac{KT\log(T|\Pi|/\delta)}{\epsilon}} + \frac{K\log(T|\Pi|/\delta)}{\epsilon}\log T + T\epsilon\right),$$

which yields the result. □

Disagreement definitions. In order to obtain improvements in regret guarantees over the worst case, we consider notions of disagreement that extend standard definitions from the active learning literature (*e.g.*, [15, 17, 18]) to the multiclass case. Let $B(\pi^*, r) := \{\pi \in \Pi : \rho(\pi, \pi^*) \leq r\}$ be the ball centered at π^* under the (pseudo)-metric $\rho(\cdot, \cdot)$. We define the disagreement region $DIS(r)$ and disagreement coefficient θ as follows:

$$DIS(r) := \{x : \exists \pi \in B(\pi^*, r) \quad \pi(x) \neq \pi^*(x)\}$$
$$\theta := \sup_{r>0} \frac{P(x \in DIS(r))}{r}.$$

The next result shows that under the Massart condition and with a finite disagreement coefficient θ, our algorithm achieves a regret that scales as $O(T^{1/3})$ (up to logarithmic factors), thus improving on worst-case guarantees obtained by optimal algorithms such as [3, 4, 11].

Theorem 8. *Assume the Massart condition* (M) *holds with parameter τ. Conditioning on the event $\mathcal{E}$ which holds w.p. $1 - \delta$, the algorithm has expected regret*

$$\mathbb{E}[R_T|\mathcal{E}] \leq O\left(\frac{K \log(T|\Pi|/\delta)}{\tau\epsilon} \log T + \frac{\theta}{\tau}\sqrt{\epsilon K T \log(T|\Pi|/\delta)}\right).$$

Optimizing over the choice of ϵ yields a regret

$$\mathbb{E}[R_T|\mathcal{E}] \leq O\left(\frac{1}{\tau}(\theta K \log(T|\Pi|/\delta))^{2/3}(T \log T)^{1/3}\right).$$

Proof. Assume $\mathcal{E}$ holds. Let $t \geq 2$, and assume $a \in A_t \setminus \{\pi^*(x_t)\}$. Define

$$\pi_a = \begin{cases} \pi_t, & \text{if } \pi_t(x_t) = a \\ \pi_{t,a}, & \text{if } \pi_t(x_t) \neq a, \end{cases}$$

so that we have $\pi_a(x_t) = a \neq \pi^*(x_t)$.

- If $\pi_a = \pi_t$, then $L(\pi_a) - L(\pi^*) \leq \Delta_t^*(\pi_a) \leq \Delta_t$ by Theorem 5
- If $\pi_a = \pi_{t,a}$, using deviation bounds, Lemma 4 and 3, we have

$$\begin{aligned} L(\pi_a) - L(\pi^*) &= L(\pi_{t,a}) - L(\pi^*) \\ &\leq \hat{L}_{t-1}(\pi_{t,a}) - \hat{L}_{t-1}(\pi^*) + \Delta_t^*(\pi_{t,a}) \\ &= \underbrace{\hat{L}_{t-1}(\pi_{t,a}) - \hat{L}_{t-1}(\pi_t)}_{\leq \Delta_t} + \underbrace{\hat{L}_{t-1}(\pi_t) - \hat{L}_{t-1}(\pi^*)}_{\leq 0} + \Delta_t^*(\pi_{t,a}) \\ &\leq 2\Delta_t, \end{aligned}$$

  where the last inequality uses $a \in A_t$.

By the Massart assumption, we then have $\rho(\pi_a, \pi^*) \leq 2\Delta_t/\tau$. Hence, we have $x_t \in DIS(2\Delta_t/\tau)$. We have thus shown

$$\mathbb{E}[\mathbb{E}[\mathbb{1}\{a \in A_t \setminus \{\pi^*(x_t)\}\}|x_t]|\mathcal{F}_{t-1}] \leq \mathbb{E}[P(x_t \in DIS(2\Delta_t/\tau))|\mathcal{F}_{t-1}] \leq 2\theta\Delta_t/\tau.$$

We then have

$$\begin{aligned}
\mathbb{E}[\ell_t(a_t) - \ell_t(\pi^*(x_t))|\mathcal{F}_{t-1}] &= \mathbb{E}[\mathbb{1}\{a_t = \pi_t(x_t)\}(\ell_t(\pi_t(x_t)) - \ell_t(\pi^*(x_t))) \\
&\quad + \mathbb{1}\{a_t = \pi^*(x_t) \wedge a_t \neq \pi_t(x_t)\}(\ell_t(\pi^*(x_t)) - \ell_t(\pi^*(x_t))) \\
&\quad + \sum_{a=1}^{K} \mathbb{1}\{a_t = a \wedge a \notin \{\pi_t(x_t), \pi^*(x_t)\}\}(\ell_t(a) - \ell_t(\pi^*(x_t)))|\mathcal{F}_{t-1}] \\
&\leq \mathbb{E}[\ell_t(\pi_t(x_t)) - \ell_t(\pi^*(x_t))|\mathcal{F}_{t-1}] \\
&\quad + \mathbb{E}\left[\sum_{a=1}^{K} \mathbb{E}[\mathbb{1}\{a_t = a\}\, \mathbb{1}\{a \notin \{\pi_t(x_t), \pi^*(x_t)\}\}|x_t]|\mathcal{F}_{t-1}\right] \\
&= L(\pi_t) - L(\pi^*) + \sum_{a=1}^{K} \mathbb{E}[\mathbb{E}[p_t(a)\, \mathbb{1}\{a \notin \{\pi_t(x_t), \pi^*(x_t)\}\}|x_t]|\mathcal{F}_{t-1}] \\
&\leq L(\pi_t) - L(\pi^*) + \frac{\epsilon}{K} \sum_{a=1}^{K} \mathbb{E}[\mathbb{E}[\mathbb{1}\{a \in A_t \setminus \{\pi^*(x_t)\}\}|x_t]|\mathcal{F}_{t-1}] \\
&\leq C\frac{K}{\tau\epsilon} e_{t-1} + 2\epsilon\theta\Delta_t/\tau,
\end{aligned}$$

where we used

$$\begin{aligned}
p_t(a)\, \mathbb{1}\{a \notin \{\pi_t(x_t), \pi^*(x_t)\}\} &= \frac{\epsilon}{K}\, \mathbb{1}\{a \in A_t \setminus \{\pi_t(x_t), \pi^*(x_t)\}\} \\
&\leq \frac{\epsilon}{K}\, \mathbb{1}\{a \in A_t \setminus \{\pi^*(x_t)\}\}.
\end{aligned}$$

Summing over t and taking total expectations (conditioned on $\mathcal{E}$) yields

$$\mathbb{E}[R_T|\mathcal{E}] \leq O\left(\frac{K\log(T|\Pi|/\delta)}{\tau\epsilon}\log T + \frac{\epsilon\theta}{\tau}\left(\sqrt{\frac{KT\log(T|\Pi|/\delta)}{\epsilon}} + \frac{K\log(T|\Pi|/\delta)}{\epsilon}\log(T)\right)\right),$$

and the result follows. □

Finally, we look at a simpler instructive example, which considers an extreme situation where the expected loss of any suboptimal policy is bounded away from that of the optimal policy. In this case, Algorithm 7 can achieve constant regret when the disagreement coefficient is bounded, as shown by the following result.

Proposition 9. *Assume that $L(\pi) - L(\pi^*) \geq \tau > 0$ for all $\pi \neq \pi^*$, and that $\theta < \infty$. Under the event $\mathcal{E}$, the algorithm achieves constant expected regret. In particular, the algorithm stops incurring regret for $T > T_0 := \max\{t : 2\Delta_t > \tau\}$.*

Proof. By Theorem 5 and our assumption, we have $L(\pi_t) - L(\pi^*) \leq \mathbb{1}\{\Delta_t \geq \tau\}\Delta_t$. Similarly, the assumption implies that $\rho(\pi, \pi^*) \leq \mathbb{1}\{L(\pi) - L(\pi^*) \geq \tau\}$, so that using similar arguments to the proof of Theorem 8, we have

$$\mathbb{E}[\mathbb{E}[\mathbb{1}\{a \in A_t \setminus \{\pi^*(x_t)\}\}|x_t]|\mathcal{F}_{t-1}] \leq \theta\, \mathbb{1}\{2\Delta_t \geq \tau\}.$$

Following the proof of Theorem 8, this implies that when t is such that $2\Delta_t < \tau$, then we have

$$\mathbb{E}[\ell_t(a_t) - \ell_t(\pi^*(x_t))|\mathcal{F}_{t-1}] = 0.$$

Let $T_0 := \max\{t : 2\Delta_t \geq \tau\}$. We thus have

$$\mathbb{E}[R_T|\mathcal{E}] \leq 1 + \sum_{t=2}^{T_0}(\Delta_t + \epsilon).$$

□