Efficient Optimal Learning for Contextual Bandits

Miroslav Dudik MDUDIK@YAHOO-INC.COM
Yahoo! Research, NY

Daniel Hsu DJHSU@RCI.RUTGERS.EDU
Rutgers University, NJ and University of Pennsylvania, PA

Satyen Kale SKALE@YAHOO-INC.COM
Yahoo! Research, CA

Nikos Karampatziakis NK@CS.CORNELL.EDU
Cornell University, NY

John Langford JL@YAHOO-INC.COM
Yahoo! Research, NY

Lev Reyzin LREYZIN@CC.GATECH.EDU
Georgia Institute of Technology, GA

Tong Zhang TZHANG@STAT.RUTGERS.EDU
Rutgers University, NJ

Abstract

We address the problem of learning in an online setting where the learner repeatedly observes features x, selects among K actions, and receives reward r for the action taken. We provide the first efficient algorithm with an optimal regret. Our algorithm uses an oracle which returns an optimal policy given rewards for all actions for each x. The algorithm has running time polylog(N), where N is the number of policies that we compete with. This is exponentially faster than all previous algorithms that achieve optimal regret in this setting. Our formulation also enables us to create an algorithm with regret that is additive rather than multiplicative in feedback delay as in all previous work.

1. Introduction

The contextual bandit setting consists of the following loop repeated indefinitely: 1) The world presents context as features x. 2) The learner chooses an action a. 3) The world presents a reward r for the action.

The key difference between the contextual bandit setting and standard supervised learning is that *only* the reward of the chosen action is revealed. The contextual bandit setting essentially captures the difficulty of exploration while avoiding the difficulty of credit assignment as in general reinforcement learning settings.

The contextual bandit setting is a half-way point between standard supervised learning and full-scale reinforcement learning. Many natural settings satisfy this half-way point, motivating the investigation of contextual bandit learning. For example, the problem of choosing interesting news articles or ads for users by internet companies can be naturally modeled as a contextual bandit setting. In the medical domain where discrete treatments are tested before approval, the process of deciding which patients are eligible for a treatment takes context into account.

In the i.i.d. setting, the world draws $(x, \vec{r})$ from some unknown distribution D, revealing x in Step 1 and the reward $r(a)$ of the chosen action a in Step 3. Given a set of policies $\Pi = \{\pi : X \to A\}$, the goal is to create an algorithm for Step 2 which competes with the set of policies. We compare the algorithm's cumulative

Appearing in the *Workshops of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.
Copyright 2011 by the author(s)/owner(s).

reward to the expected cumulative reward of the best policy in the set.

All existing algorithms for this setting either achieve a suboptimal regret (Langford & Zhang, 2007) or require computation linear in the number of policies (Auer et al., 2002b; Beygelzimer et al., 2010). Herein, we give the first optimization-based algorithm for the contextual bandit setting. Given an oracle optimizer for supervised cost-sensitive learning (Beygelzimer et al., 2009), our algorithm runs in time only polylog(N) while achieving regret $O(\sqrt{TK \ln N})$. We achieve this efficiency in a modular way, so any improvement in cost-sensitive learning immediately applies here.

All previous optimal approaches are *measure* based—they work by updating a measure over policies, which is linear in the number of policies. In contrast, regret guarantees scale only logarithmically in the number of policies. If not for this computational bottleneck, these regret guarantees imply that we could dramatically increase performance in contextual bandit settings using more expressive policies. We overcome the computational bottleneck using an optimization-based algorithm which works by choosing optimal policies rather than keeping track of a measure over policies.

1.1. Previous Work and Motivation

In a more difficult version of contextual bandits, an adversary chooses $(x, \vec{r})$ given knowledge of the learning algorithm (but not any random numbers). All known regret-optimal solutions in this setting are variants the EXP4 algorithm (Auer et al., 2002b). EXP4 achieves the same regret rate as our algorithm: $O\left(\sqrt{KT \ln N}\right)$, where T is the number of time steps, K is the number of available actions, and N is the number of policies.

Why not use EXP4 in the i.i.d. setting? For example, it is known that the algorithm can be modified to succeed with high probability (Beygelzimer et al., 2010), and also for VC classes when the adversary is constrained to i.i.d. sampling. There are two main benefits one may hope to gain from an i.i.d. argument.

The first is computational tractability. Even when the reward vector is fully known, regrets scale as $O\left(\sqrt{\ln N}\right)$ while computation scales as $O(N)$ in general. One attempt to get around this is the follow-the-perturbed-leader algorithm (Kalai & Vempala, 2005) which provides a computationally tractable solution in certain special-case structures. This algorithm has no mechanism for efficient application to arbitrary policy spaces, even given an optimization oracle over the policies. An efficient optimization oracle has been shown effective in transductive settings (Kakade & Kalai, 2005). Furthermore, the regret achieved there is substantially worse than for EXP4.

The second main benefit is improved rates. When the world is i.i..d., it's possible to achieve substantially lower regrets than with algorithms for the adversarial setting. For example, in supervised learning, regrets scaling as $O(\log(T))$ with a problem dependent constant are possible. When the feedback is delayed by τ rounds, lower bounds imply that the regret in the adversarial setting increases by a multiplicative $\sqrt{\tau}$ while in the i.i.d. setting, an additive regret of τ is possible.

In the i.i.d. setting, the previous-best were ϵ-greedy and epoch greedy algorithms (Langford & Zhang, 2007) whose worst-case regret scales as $O(T^{2/3})$.

There have also been many special-case analyses. For example, the context-free setting is well understood (Lai & Robbins, 1985; Auer et al., 2002a; Even-Dar et al., 2006). Similarly when rewards are linear in the features (Auer, 2002) or Gaussian (Srinivas et al., 2010), good algorithms are known.

1.2. What We Prove

In Section 3 we state the Policy_Elimination algorithm, and prove the following regret bound for it.

Theorem 4. For all distributions D over K actions and features, for all sets of N policies Π, with probability at least $1-\delta$, the regret of the Policy_Elimination algorithm (Algorithm 1) over T rounds is at most

$$16\sqrt{2TK \ln(4T^2N/\delta)}.$$

This result forms the simplest method we have of exhibiting the new analysis.

The main new element of this algorithm is a mechanism for constructing a distribution over actions via a distribution over policies which achieves both small expected regret and small variance in the estimated value of every policy. The key insight boils down to a game between an adversary and the algorithm.

The Policy_Elimination algorithm is computationally intractable and also requires the learner to have knowledge of the unlabeled data distribution. We show how to address these issues in Section 4 using an algorithm, Randomized_UCB. Namely, we prove the following.

Theorem 5. For all distributions D over K actions and features, for all sets of N policies Π, with probability at least $1-\delta$, the regret of the Randomized UCB algorithm (Algorithm 2) over T rounds is at most

$$O\left(\sqrt{TK \log\left(TN/\delta\right)} + K \log(NK/\delta)\right).$$

Randomized_UCB's analysis is substantially more complex, with a key subroutine being an application of the ellipsoid algorithm on an optimization oracle (described in Section 5). The Randomized_UCB algorithm also works with its own existing history of unlabeled data points and, unlike Policy_Elimination, does not have access to the unlabeled data distribution. Modifying the proof in this manner requires a covering argument over the distributions over policies which uses the probabilistic method. The net result is an algorithm with a similar analysis that has only a logarithmic dependence on the number of policies.

Theorem 11. In each time step t, the Randomized UCB algorithm makes at most $O(\text{poly}(t, K, \log(1/\delta), \log N))$ calls to an optimization oracle, and requires additional $O(\text{poly}(t, K, \log N))$ processing time.

Another key advantage of this style of analysis is the ability to prove tighter results than for adversarial settings. We provide one example of this for the common setting where reward feedback is delayed by τ rounds in Section 6. Here, a straightforward modification of the Policy_Elimination algorithm yields a regret only τ larger than in the delay-free setting, namely:

Theorem 12. For all distributions D over K actions and features, for all sets of N policies Π, and all delay factors τ, with probability at least $1-\delta$, the regret of the Delayed Policy Elimination algorithm is at most

$$16\sqrt{2K\ln(4T^2N/\delta)}\left(\tau+\sqrt{T}\right).$$

2. The Setting and Definitions

2.1. The Setting

Let A be the set of K actions, X be the domain of contexts x_t, let D be any joint distribution on $(x, \vec{r})$. D_X denotes the marginal distribution of D over X

We denote Π to be a finite set of policies $\{\pi : X \to A\}$, as in each policy π, predicts according to $\pi(x_t)$, where x_t is the context available in round t. N denotes $|\Pi|$. Let $\vec{r}_t \in [0,1]^K$ be the vector of rewards, where $r_t(a)$ is the reward of action a on round t.

In the i.i.d. setting, on each round $t = 1 \dots T$, the world chooses $(x_t, \vec{r}_t)$ i.i.d. according to D and reveals x_t to the learner. Then the learner, having access to Π, chooses action $a_t \in \{1, \dots, K\}$. Finally, the world reveals reward $r_t(a_t)$ (which we call r_t for short), and this game proceeds to the next round. The number of rounds T is not known in advance to the learner.

The goal of the learner is to minimize its regret to Π. This notion is defined in Equation 2.1.

2.2. Expected and Empirical Rewards

Let the expected instantaneous **reward** of a policy $\pi \in \Pi$ be denoted by

$$\eta_D(\pi) \doteq \mathop{\mathbb{E}}_{(x,\vec{r})\sim D}[r(\pi(x))].$$

The best policy $\pi_{\max} \in \Pi$ maximizes $\eta_D(\pi)$:

$$\pi_{\max} \doteq \mathop{\text{argmax}}_{\pi\in\Pi} \eta_D(\pi).$$

We can define h_t to be the **history** at time t that the learner has seen. Specifically

$$h_t = \bigcup_{t'=1\dots t} (x_{t'}, a_{t'}, r_{t'}(a_{t'}), p),$$

where p is the *a priori* probability of the algorithm having taken its chosen action a_t at time t. If we want to choose x u.a.r. from the x's in history h, we denote this by $x \sim h$.

We can also define the empirical (importance-weighted) value of a policy as:

$$\eta_t(\pi) \doteq \frac{1}{t}\sum_{(x,a,r,p)\in h_t} \frac{rI(\pi(x)=a)}{p}.$$

We denote π_t to be the best policy measured at time t

$$\pi_t \doteq \mathop{\text{argmax}}_{\pi\in\Pi} \eta_t(\pi).$$

2.3. Regret

Our is to obtain a learner that has small **regret** to the expected performance of $\pi_{\max}$ over T rounds, which is

$$\sum_{t=1}^{T} \left(\eta_D(\pi_{\max}) - r_t\right). \tag{2.1}$$

We say that the regret of the learner over T rounds is bounded by ϵ with probability at least $1-\delta$, if

$$\Pr\left[\sum_{t=1}^{T}\left(\eta_D(\pi_{\max}) - r_t(a_t)\right) \le \epsilon\right] \ge 1-\delta.$$

The probability is taken w.r.t. $(x_t, \vec{r}_t) \sim D$, as well as any internal randomness used by the learner.

We can also define notions of regret for policies π. Let

$$\Delta_D(\pi) \doteq \eta_D(\pi_{\max}) - \eta_D(\pi),$$
$$\Delta_t(\pi) \doteq \eta_t(\pi_t) - \eta_t(\pi).$$

Our algorithms work by choosing distributions over policies, which in turn then induce distributions over

actions. For any distribution P over policies Π, let $W_P(x,a)$ denote the induced conditional distribution over actions a given the context x:

$$W_P(x,a) \doteq \sum_{\pi\in\Pi:\pi(x)=a} P(\pi). \tag{2.2}$$

In general, we shall use W, W' and Z as probability distributions over the actions A, namely

$$\{W, W', Z : X \times A \to p(A)\}.$$

W' can be thought of as a dampened version of W with a minimum probability of μ, s.t. $\forall x \in X, a \in A$

$$W'(x,a) = (1 - K\mu)W(x,a) + \mu.$$

We can now define notions of regret also for probability distributions W (and W', Z, etc.). Let

$$\Delta_D(W) \doteq \eta_D(\pi_{\max}) - \eta_D(W),$$

$$\Delta_t(W) \doteq \eta_t(\pi_t) - \eta_t(W).$$

with

$$\eta_D(W) \doteq \mathop{\mathbb{E}}_{(x,\vec{r})\sim D}[\vec{r}\cdot W(x)],$$

$$\eta_t(W) \doteq \frac{1}{t}\sum_{(x,a,r,p)\in h_t} \frac{rW(x,a)}{p}.$$

3. Policy Elimination

We present Algorithm 1, Policy_Elimination, which demonstrates the basic ideas behind our approach.

The insight which allows this algorithm to work is Step 1, which finds a distribution over policies which induces low variance in the estimates of the value of all policies. We prove that this is always possible using a minimax theorem. How to find this distribution is discussed in Section 5.

Step 2 then projects this distribution over actions, together with the uniform distribution mixed in. Finally, Step 5 eliminates the policies that have been determined to be w.h.p. suboptimal.

Algorithm Analysis

The following minimax theorem considers randomized policies, i.e. maps $W : X \times A \to [0,1]$ where $W(x,a)$ is the probability of choosing action a on a context x.

Lemma 1. *Let $\mathcal{C}$ be a compact and convex set of randomized policies. Let $\mu \in (0, 1/K]$ and for any $W \in \mathcal{C}$, $W'(x,a) \doteq (1-K\mu)W(x,a) + \mu$. Then for all distributions D,*

$$\min_{W\in\mathcal{C}}\max_{Z\in\mathcal{C}} \mathop{\mathbb{E}}_{x\sim D_X}\mathop{\mathbb{E}}_{a\sim Z(x,\cdot)}\left[\frac{1}{W'(x,a)}\right] \le \frac{K}{1-K\mu} \quad .$$

Algorithm 1 Policy_Elimination(Π,δ,K,D_X)

Let $\Pi_0 = \Pi$ and history $h_0 = \emptyset$

Define: $\delta_t \doteq \delta \,/\, 4Nt^2$

Define: $b_t \doteq 2\sqrt{\dfrac{2K\ln(1/\delta_t)}{t}}$

Define: $\mu_t \doteq \min\left\{\dfrac{1}{2K}, \sqrt{\dfrac{\ln(1/\delta_t)}{2Kt}}\right\}$

For each timestep $t = 1 \ldots T$, observe x_t and do:

1. Choose distribution P_t over Π_{t-1} s.t. $\forall\ \pi \in \Pi_{t-1}$:
$$\mathop{\mathbb{E}}_{x\sim D_X}\left[\frac{1}{(1-K\mu_t)W_{P_t}(x,\pi(x)) + \mu_t}\right] \le 2K$$
2. Let $W'_t(a) = (1-K\mu_t)W_{P_t}(x_t,a) + \mu_t$ for all $a \in A$
3. Choose $a_t \sim W'_t$
4. Observe reward r_t
5. Let $\Pi_t = \left\{\pi \in \Pi_{t-1} : \eta_t(\pi) \ge \max_{\pi'\in\Pi_{t-1}} \eta_t(\pi') - 2b_t\right\}$
6. Let $h_t = h_{t-1} \cup (x_t, a_t, r_t, W'_t(a_t))$

Proof. Let $f(W,Z) \doteq \mathbb{E}_{x\sim D_X}\mathbb{E}_{a\sim Z(x,\cdot)}[1/W'(x,a)]$ denote the inner expression of the minimax problem. Note that $f(W,Z)$ is: everywhere defined, linear in Z, and convex in W. Hence, by Sion's Minimax Theorem,

$$\min_{W\in\mathcal{C}}\max_{Z\in\mathcal{C}} f(W,Z) = \max_{Z\in\mathcal{C}}\min_{W\in\mathcal{C}} f(W,Z) \quad .$$

The right-hand side can be further upper-bounded by $\max_{Z\in\mathcal{C}} f(Z,Z)$, which is upper-bounded by

$$\begin{aligned} f(Z,Z) &= \mathop{\mathbb{E}}_{x\sim D_X}\sum_{a\in A}\left[\frac{Z(x,a)}{Z'(x,a)}\right] \\ &\le \mathop{\mathbb{E}}_{x\sim D_X}\sum_{a\in A}\left[\frac{Z(x,a)}{(1-K\mu)Z(x,a)}\right] = \frac{K}{1-K\mu} \end{aligned}$$

□

Corollary 2. *The set of distributions satisfying constraints of Step 1 is non-empty.*

Lemma 1 and Corollary 2 establish existence of a distribution P_t in Step 1. As we will see below, the constraints in Step 1 ensure low variance of the policy value estimator $\eta_h(\pi)$ for all $\pi \in \Pi_{t-1}$. The small variance is in turn used to ensure accuracy of policy elimination in Step 5 as quantified as follows:

Lemma 3. *W.p. at least $1-\delta$, for all t: 1) $\pi_{\max} \in \Pi_t$ and 2) $\eta_D(\pi_{\max}) - \eta_D(\pi) \le 4b_t$ for all $\pi \in \Pi_t$*

Proof. We will show that for any policy $\pi \in \Pi_{t-1}$, the probability that $\eta_t(\pi)$ deviates from $\eta_D(\pi)$ by more that b_t is at most δ_t. Taking the union bound over all policies and all time steps we find that w.p. $\geq 1-\delta$,

$$|\eta_t(\pi) - \eta_D(\pi)| \leq b_t \tag{3.1}$$

for all t and all $\pi \in \Pi_{t-1}$. Then by triangle inequality the lemma follows.

It remains to show Eq. (3.1). We fix the policy $\pi \in \Pi$ time t, and show that the deviation bound is violated with probability at most δ_t. Our argument rests on Freedman's inequality. Let

$$y_t = \frac{r_t \mathbb{I}(\pi(x_t) = a_t)}{W'_t(a_t)} \quad .$$

Let $\mathbb{E}_t$ denote the conditional expectation $\mathbb{E}[\cdot \,|\, h_{t-1}]$. To use Freedman's inequality, we need to bound the range of y_t and its conditional second moment $\mathbb{E}_t[y_t^2]$. Since $r_t \in [0,1]$ and $W'_t(a_t) \geq \mu_t$, we have $0 \leq y_t \leq 1/\mu_t \doteq R_t$. Next,

$$\begin{aligned}
\mathbb{E}_t[y_t^2] &= \mathop{\mathbb{E}}_{(x_t,\vec{r}_t)\sim D} \mathop{\mathbb{E}}_{a_t \sim W'_t} \left[y_t^2\right] \\
&= \mathop{\mathbb{E}}_{(x_t,\vec{r}_t)\sim D} \mathop{\mathbb{E}}_{a_t \sim W'_t} \left[\frac{r_t^2 \mathbb{I}(\pi(x_t) = a_t)}{W'_t(a_t)^2}\right] \\
&\leq \mathop{\mathbb{E}}_{(x_t,\vec{r}_t)\sim D} \left[\frac{W'_t(\pi(x_t))}{W'_t(\pi(x_t))^2}\right] \leq 2K \quad . \qquad (3.2)
\end{aligned}$$

Eq. (3.2) follows from r_t being bounded and the constraints in Step 1. Hence, $\sum_{t'=1}^{t} \mathbb{E}_{t'}[y_{t'}^2] \leq 2Kt \doteq V_t$.

Since $K \geq 2$ and $(\ln t)/t$ is decreasing for $t \geq 3$, we obtain that μ_t is non-decreasing. Let t_0 be the first t such that $\mu_t < 1/2K$. Note that $b_t \geq 4K\mu_t$, so for $t < t_0$, $b_t \geq 2$. and $\Pi_t = \Pi$. Hence, the deviation bound holds for $t < t_0$.

Let $t \geq t_0$. For $t' \leq t$, by the monotonicity of μ_t

$$R_{t'} = 1/\mu_{t'} \leq 1/\mu_t = \sqrt{\frac{2Kt}{\ln(1/\delta_t)}} = \sqrt{\frac{V_t}{\ln(1/\delta_t)}} \quad .$$

We can now apply a version of Freedman's inequality (Beygelzimer et al., 2010, Theorem 1),

$$\Pr\left[|\eta_t(\pi) - \eta_D(\pi)| \geq b_t\right] \leq \delta_t \quad .$$

The union bound over π and t yields Eq. (3.1). □

This immediately implies the following regret bound

$$\begin{aligned}
\sum_{t=1}^{T} (\eta_D(\pi_{\max}) - r_t) &\leq 8\sqrt{2K \ln \frac{4NT^2}{\delta}} \sum_{t=1}^{T} \frac{1}{\sqrt{t}} \\
&\leq 16\sqrt{2TK \ln \frac{4T^2 N}{\delta}} \qquad (3.3)
\end{aligned}$$

and gives us the following theorem.

Theorem 4. *For all distributions D over K actions and features, for all sets of N policies Π, with probability at least $1-\delta$, the regret of the Policy_Elimination algorithm (Algorithm 1) over T rounds is at most*

$$16\sqrt{2TK \ln(4T^2 N/\delta)}.$$

4. The Randomized UCB Algorithm

Policy Elimination is the simplest exhibition of the minimax argument, but it has some drawbacks:

1. The algorithm keeps explicit track of all the good policies and is thereby computationally expensive.
2. If the optimal policy is mistakenly eliminated by chance, the algorithm can never recover.
3. The algorithm requires perfect knowledge of the distribution D_X over contexts.

These difficulties are addressed by the Randomized UCB (RUCB) algorithm. The idea here is to have a UCB style algorithm where instead of choosing the highest UCB, we randomize over choices according to the value of their empirical performance. The algorithm has the following properties:

1. The optimization required by the algorithm always considers the full set of policies (*i.e.*, instead of explicitly tracking of the set of good policies), and thus it can be efficiently implemented using an ERM-type oracle. We discuss this in Section 5.
2. Suboptimal policies are implicitly used with decreasing frequency by using a non-uniform variance constraint that depends on a policy's estimated regret. A consequence of this is a bound on the value of the optimization, stated in Lemma 6.
3. The history of previously seen contexts is used as a surrogate for the distribution over contexts in the optimization. We discuss this in Subsection 4.2.

Randomized UCB has the following guarantee.

Theorem 5. *For all distributions D over K actions and features, for all sets of N policies Π, with probability at least $1-\delta$, the regret of the Randomized UCB algorithm (Algorithm 2) over T rounds is at most*

$$O\left(\sqrt{TK \log(TN/\delta)} + K \log(NK/\delta)\right).$$

Algorithm 2 RUCB(Π,δ,K)

Let $h_0 \doteq \emptyset$ be the initial history.
Define the following quantities:

$$C_t \doteq 2\log\left(\frac{Nt}{\delta}\right) \quad \text{and} \quad \mu_t \doteq \min\left\{\frac{1}{2K}, \sqrt{\frac{C_t}{2Kt}}\right\}.$$

For each timestep $t = 1 \dots T$, observe x_t and do:

1. Let P_t be a distribution over Π that approximately solves the optimization problem

$$\min_P \sum_{\pi\in\Pi} P(\pi)\Delta_{t-1}(\pi) \text{ s.t. for all distr. } Q \text{ over } \Pi: \\ \mathop{\mathbb{E}}_{\pi\sim Q}\left[\frac{1}{t-1}\sum_{i=1}^{t-1}\frac{1}{(1-K\mu_t)W_P(x_i,\pi(x_i))+\mu_t}\right] \\ \le \max\left\{4K, \frac{(t-1)\Delta_{t-1}(W_Q)^2}{180C_{t-1}}\right\} \tag{4.1}$$

   so that the objective value at P_t is within $\varepsilon_{\text{opt},t} = O(\sqrt{KC_t/t})$ of the optimal value, and so that each constraint is satisfied with slack $\le K$.

2. $\forall a \in A$, let W'_t be the following distribution on A

$$W'_t(a) \doteq (1-K\mu_t)W_{P_t}(x_t,a) + \mu_t$$

3. Choose $a_t \sim W'_t$.
4. Observe reward r_t.
5. Let $h_t \doteq h_{t-1} \cup (x, a_t, r_t, W'_t(a_t))$.

4.1. Overview of the Analysis

Central to the analysis is the following lemma that bounds the value of the optimization in each round.

Lemma 6. *If* OPT_t *is the value of the optimization problem* (4.1) *in round* t*, then*

$$\text{OPT}_t \le O\left(\sqrt{K\log(Nt/\delta)/t}\right).$$

This lemma implies that the algorithm is always able to select a distribution over the policies that focuses mostly on the policies with low estimated regret. Moreover, the variance constraints ensure that good policies never appear too bad, and that only bad policies are allowed to incur high variance in their reward estimates. Hence, minimizing the objective in (4.1) is an effective surrogate for minimizing regret.

The analysis mostly consists of analyzing the variance of the reward estimates $\eta_t(\pi)$, and showing how they relate to their actual expected rewards $\eta_D(\pi)$.

4.2. Empirical Variance Estimates

For a distribution P over policies Π and a particular policy $\pi \in \Pi$, define

$$V_{P,\pi,t} = \mathop{\mathbb{E}}_{x\sim D_X}\left[\frac{1}{(1-K\mu_t)W_P(x,\pi(x))+\mu_t}\right]$$
$$\widehat{V}_{P,\pi,t} = \frac{1}{t-1}\sum_{i=1}^{t-1}\frac{1}{(1-K\mu_t)W_P(x_i,\pi(x_i))+\mu_t}.$$

The first quantity $V_{P\pi,t}$ bounds the variance incurred by an importance-weighted estimate of reward in round t using the action distribution induced by P, and the second quantity $\widehat{V}_{P,\pi,t}$ is an empirical estimate of $V_{P,\pi,t}$ using the finite sample $\{x_1,\dots,x_{t-1}\} \subseteq X$ drawn from D_X. We show that for all distributions P and all $\pi \in \Pi$, $\widehat{V}_{P,\pi,t}$ is close to $V_{P,\pi,t}$ w.h.p.

Theorem 7. *For any* $\epsilon \in (0,1)$*, w.p.* $\ge 1-\delta$*,*

$$V_{P,\pi,t} \le (1+\epsilon)\cdot\widehat{V}_{P,\pi,t} + 7500/\epsilon^3 \cdot K$$

for all distributions P *over* Π*, all* $\pi \in \Pi$*, and all* $t \ge 16K\log(8KN/\delta)$*.*

5. Using an Oracle Directly

Assume that we have access to an arg-max oracle, which when supplied with a set of examples consisting of contexts and rewards, returns the policy that maximizes the expected reward.

Definition 1. There is an algorithm, $\mathcal{AMO}$, which when given any history $h = (X\times\mathbb{R}^k)^*$, computes

$$\mathcal{AMO}(h) \;:=\; \arg\max_{\pi\in\Pi} \mathop{\mathbb{E}}_{(x,\vec{r})\sim h}[r(\pi(x))].$$

$(x,\vec{r}) \sim h$ is used to denote a u.a.r. draw from h.

We now show that there is an algorithm running in polynomial time *independent*[1] of the number of policies, which make queries to $\mathcal{AMO}$ to compute a distribution over policies suitable for the optimization step of Algorithm 2.

This algorithm relies on the ellipsoid method. The ellipsoid method is a general technique for solving convex programs equipped with a separation oracle. A separation oracle is defined as follows:

Definition 2. Let S be a convex set in $\mathbb{R}^n$. A separation oracle for S is an algorithm that, given a point

[1] Or rather dependent only on $\log N$, the representation complexity of a policy.

$x \in \mathbb{R}^n$, either declares correctly that $x \in S$, or produces a hyperplane H such that x and S are on opposite sides of H.

For a point $x \in \mathbb{R}^n$ and $r \geq 0$, $B(x, r)$ denotes the ℓ_2 ball of radius r centered at x.

Lemma 8. *Suppose we are required to decide whether a convex set $S \subseteq \mathbb{R}^n$ is empty. Assume that we are given a separation oracle for S. Assume further that we are given two numbers R and r, s.t. $S \in B(0, R)$ and if S is non-empty, then there is a point $x^\star$ s.t. $S \supseteq B(x^\star, r)$. Then there is an iterative algorithm with at most $O(n^2 \log(\frac{R}{r}))$ iterations, each involving one call to the separation oracle and additional $O(n^2)$ processing time, that decides if S is empty or not.*

We now write a convex program whose solution is the required distribution, and show how to solve it using the ellipsoid method by giving a separation oracle for its feasible set using $\mathcal{AMO}$.

Fix a time period t. Let $\mathcal{X}_{t-1}$ be the set of all contexts seen so far, i.e. $\mathcal{X}_{t-1} = \{x_1, x_2, \ldots, x_{t-1}\}$. We embed all policies $\pi \in \Pi$ in $\mathbb{R}^{(t-1)K}$, with coordinates identified with $(x, a) \in \mathcal{X}_{t-1} \times A$. Abusing notation, a policy π is represented by the vector π with coordinate $\pi(x, a) = 1$ if $\pi(x) = a$ and 0 otherwise. Let $\mathcal{C}$ be the convex hull of all policy vectors π. Recall that a distribution P over policies corresponds to a point inside $\mathcal{C}$. Also define $\beta_t = \frac{t-1}{180 C_{t-1}}$. In the following, we use the notation $x \sim h_{t-1}$ to denote a context drawn uniformly at random from $\mathcal{X}_{t-1}$.

Consider the convex program to minimize s s.t.

$$\Delta_{t-1}(W) \leq s \tag{5.1}$$

$$W \in \mathcal{C} \tag{5.2}$$

$$\forall Z \in \mathcal{C} : \mathop{\mathbb{E}}_{x \sim h_{t-1}} \left[\sum_a \frac{Z(x,a)}{W'(x,a)} \right] \leq \max\{4K, \beta_t \Delta_{t-1}(Z)^2\}, \tag{5.3}$$

It is not difficult to see that this program is equivalent to the RUCB optimization problem (4.1), up to finding an explicit distribution over policies which corresponds to the optimal solution.

For a fixed value of s, the constraints (5.1, 5.2, 5.3) define a feasibility problem, denoted by $\mathcal{A}$.

We now give a sketch of how we construct a separation oracle for the feasible region of $\mathcal{A}$. The details of the algorithm are a bit complicated due to the fact that we need to ensure that the feasible region, when non-empty, has a non-negligible volume (recall the requirements of Lemma 8). This necessitates having a small error in satisfying the constraints of the program. Modulo these details, the construction of the separation oracle essentially implies that we can solve $\mathcal{A}$.

Before giving the construction of the separation oracle, we first show that $\mathcal{AMO}$ allows us to do linear optimization over $\mathcal{C}$ efficiently:

Lemma 9. *Given a vector $w \in \mathbb{R}^{(t-1)K}$, we can compute $\arg\max_{Z \in \mathcal{C}} w \cdot Z$ using one invocation of $\mathcal{AMO}$.*

The following lemma explains how to get a separating hyperplane for violations of convex constraints:

Lemma 10. *For $x \in \mathbb{R}^n$, let $f(x)$ be a convex function of x, and define the convex set K by $K = \{x : f(x) \leq 0\}$. Suppose we have a point y such that $f(y) > 0$. Let $\nabla f(y)$ be a subgradient of f at y. Then the hyperplane $f(y) + \nabla f(y) \cdot (x - y) = 0$ separates y from K.*

Now given a candidate point W, a separation oracle can be constructed as follows. We check whether W satisfies the constraints of $\mathcal{A}$. If any constraint is violated, then we find a hyperplane separating W from all points satisfying the constraint.

1. First, for constraint (5.1), note that $\eta_{t-1}(W)$ is linear in W, and so we can compute $\max_\pi \eta_{t-1}(\pi)$ via $\mathcal{AMO}$ as in Lemma 9. We can then compute $\eta_{t-1}(W)$ and check if the constraint is satisfied. If not, then the constraint, being linear, automatically yields a separating hyperplane.

2. Next, we consider constraint (5.2). To check if $W \in \mathcal{C}$, we use the perceptron algorithm. We shift the origin to W, and run perceptron with all points $\pi \in \Pi$ as positive examples. Perceptron aims to find a hyperplane putting all policies $\pi \in \Pi$ on one side. In each iteration, we have a candidate hyperplane, and then if there is a policy π that is on the wrong side of the hyperplane, we can find it by running a linear optimization over $\mathcal{C}$ in the negative normal vector direction as in Lemma 9. If $W \notin \mathcal{C}$, then in a bounded number of iterations we obtain a separating hyperplane.

3. Finally, we consider constraint (5.3). We first rewrite $\eta_{t-1}(W)$ as $\eta_{t-1}(W) = w \cdot W$, where w is a vector defined as $w(x, a) = \frac{1}{t-1} \sum_{(x', a', r, p) \in h_t :\ x' = x, a' = a} \frac{r}{p}$. Thus, $\Delta_{t-1}(Z) = v - w \cdot Z$, where $v = \max_{\pi'} \eta_{t-1}(\pi') = \max_{\pi'} w \cdot \pi'$ which can be computed by using $\mathcal{AMO}$ once.

   Next, using the candidate point W, compute the vector u defined as $u(x, a) = \frac{n_x / t}{W'(x,a)}$, where n_x is the number of times x appears in h_{t-1}, so that $\mathbb{E}_{x \sim h_{t-1}} \left[\sum_a \frac{Z(x,a)}{W'(x,a)} \right] = u \cdot Z$. Now, the problem

reduces to finding a policy $Z \in \mathcal{C}$ which violates the constraint: $u \cdot Z \leq \max\{4K, \beta_t(w \cdot Z - v)^2\}$.

Define $f(Z) = \max\{4K, \beta_t(w \cdot Z - v)^2\} - u \cdot Z$. Note that f is convex function of Z. Finding a point Z that violates the above constraint is equivalent to solving the following (convex) program:

$$f(Z) \leq 0 \tag{5.4}$$
$$Z \in \mathcal{C} \tag{5.5}$$

To do this, we again apply the ellipsoid method. For this, we need a separation oracle for the program. A separation oracle for the constraints (5.5) can be constructed as in Step 2 above. For the constraints (5.4), if the candidate solution Z has $f(Z) > 0$, then we can construct a separating hyperplane as in Lemma 10. Thus we can solve the program by the ellipsoid method.

Suppose that after solving the program, we get a point $Z \in \mathcal{C}$ such that $f(Z) \leq 0$, i.e. W violates the constraint (5.3) for Z. Then since constraint (5.3) is convex in W, we can construct a separating hyperplane as in Lemma 10.

Working out the details carefully gives the following.

Theorem 11. *There is an iterative algorithm with at most $O(t^5K^4\log^2(\frac{tK}{\delta}))$ iterations, each involving one call to $\mathcal{AMO}$ and $O(t^2K^2)$ processing time, that either outputs an explicit distribution P over policies in Π such that W_P satisfies*

$$\forall Z \in \mathcal{C}:$$
$$\mathop{\mathbb{E}}_{x \sim h_{t-1}}\left[\sum_a \frac{Z(x,a)}{W'_P(x,a)}\right] \leq \max\{4K, \beta_t \Delta_{t-1}(Z)^2\} + 5\epsilon$$
$$\Delta_{t-1}(W) \leq s + 2\gamma,$$

where $\epsilon = \frac{8\delta}{\mu_t^2}$ and $\gamma = \frac{\delta}{\mu_t}$, or declares correctly that $\mathcal{A}$ is infeasible.

6. The Delayed Feedback Setting

In a delayed feedback setting, we observe rewards with a τ step delay according to:

1. The world presents features x_t.
2. The learner chooses an action $a_t \in \{1, ..., K\}$.
3. The world presents a reward $r_{t-\tau}$ for the action $a_{t-\tau}$ given the features $x_{t-\tau}$.

It is not difficult to suitably modify Algorithm 1 to incorporate the delay factor τ by updating the history and policy sets with the delay. We call this the Delayed Policy Elimination algorithm. The modification to the proof is minor and yields the following.

Theorem 12. *For all distributions D over K actions and features, for all sets of N policies Π, and all delay factors τ, with probability at least $1 - \delta$, the regret of the Delayed Policy Elimination algorithm is at most*

$$16\sqrt{2K\ln(4T^2N/\delta)}\left(\tau + \sqrt{T}\right).$$

References

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002a.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, Y. E. The nonstochastic multiarmed bandit problem. *SIAM Journal of Computing*, 32(1):48–77, 2002b.

Beygelzimer, A., Langford, J., and Ravikumar, P. Error correcting tournaments. In *ALT*, 2009.

Beygelzimer, A., Langford, J., Li, L., Reyzin, L., and Schapire, R. E. An optimal high probability algorithm for the contextual bandit problem. *CoRR*, abs/1002.4058, 2010.

Even-Dar, E., Mannor, S., and Mansour, Y. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 7:1079–1105, 2006.

Kakade, S. M. and Kalai, A. From batch to transductive online learning. In *NIPS*, 2005.

Kalai, A. T. and Vempala, Santosh. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3):291–307, 2005.

Lai, T. L. and Robbins, H. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

Langford, J. and Zhang, T. The epoch-greedy algorithm for contextual multi-armed bandits. In *NIPS*, 2007.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, pp. 1015–1022, 2010.