Efficient Optimal Learning for Contextual Bandits

Miroslav Dudik mdudik@yahoo-inc.com
Daniel Hsu djhsu@rci.rutgers.edu
Satyen Kale skale@yahoo-inc.com
Nikos Karampatziakis nk@cs.cornell.edu
John Langford jl@yahoo-inc.com
Lev Reyzin lreyzin@cc.gatech.edu
Tong Zhang tzhang@stat.rutgers.edu

Abstract

We address the problem of learning in an online setting where the learner repeatedly observes features, selects among a set of actions, and receives reward for the action taken. We provide the first efficient algorithm with an optimal regret. Our algorithm uses a cost sensitive classification learner as an oracle and has a running time polylog(N), where N is the number of classification rules among which the oracle might choose. 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 information as features x.
2. The learning algorithm chooses an action a from K possible actions.
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. For example, after always choosing the same action several times in a row, the feedback given provides almost no basis to prefer the chosen action over another action. In essence, the contextual bandit setting captures the difficulty of exploration while avoiding the difficulty of credit assignment as in more general reinforcement learning settings.

The contextual bandit setting is a half-way point between standard supervised learning and full-scale reinforcement learning where it appears possible to construct algorithms with convergence rate guarantees similar to supervised 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 contexts into account. More generally, we can imagine that in a future with personalized medicine, new treatments are essentially equivalent to new actions in a contextual bandit setting.

In the i.i.d. setting, the world draws a pair $(x, \vec{r})$ consisting of a context and a reward vector from some unknown distribution D, revealing x in Step 1, but only 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 measure our success by comparing the algorithm's cumulative reward to the expected cumulative reward of the best policy in the set. The difference of the two is called *regret.*

All existing algorithms for this setting either achieve a suboptimal regret (Langford and Zhang, 2007) or require computation linear in the number of policies (Auer et al., 2002b; Beygelzimer et al., 2011). In unstructured policy spaces, this computational complexity is the best one can hope for. On the other hand, in the case where the rewards of all actions are revealed, the problem is equivalent to cost-sensitive classification, and we know of algorithms to efficiently search the space of policies (classification rules) such as cost-sensitive logistic regression and support vector machines. In these cases, the space of classification rules is exponential in the number of features, but these problems can be efficiently solved using convex optimization.

Our goal here is to efficiently solve the contextual bandit problems for similarly large policy spaces. We do this by reducing the contextual bandit problem to cost-sensitive classification. Given a supervised cost-sensitive learning algorithm as an oracle (Beygelzimer et al., 2009), our algorithm runs in time only polylog(N) while achieving regret $O(\sqrt{TK \ln N})$, where N is the number of possible policies (classification rules), K is the number of actions (classes), and T is the number of time steps. This efficiency is achieved in a modular way, so any future improvement in cost-sensitive learning immediately applies here.

1.1 PREVIOUS WORK AND MOTIVATION

All previous regret-optimal approaches are *measure* based—they work by updating a measure over policies, an operation which is linear in the number of policies. In contrast, regret guarantees scale only logarithmically in the number of policies. If not for the 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 algorithm which works by creating cost-sensitive classification instances and calling an oracle to choose optimal policies. Actions are chosen based on the policies returned by the oracle rather than according to a measure over all policies. This is reminiscent of AdaBoost (Freund and Schapire, 1997), which creates weighted binary classification instances and calls a "weak learner" oracle to obtain classification rules. These classification rules are then combined into a final classifier with boosted accuracy. Similarly as AdaBoost converts a weak learner into a strong learner, our approach converts a cost-sensitive classification learner into an algorithm that solves the contextual bandit problem.

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 the adversarial setting are variants of 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 actions available in each time step, 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., 2011), and also for VC classes when the adversary is constrained to i.i.d. sampling. There are two central benefits that we hope to realize by directly assuming i.i.d. contexts and reward vectors.

1. Computational Tractability. Even when the reward vector is fully known, adversarial 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 and 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 efficient cost-sensitive classification oracle. An efficient cost-sensitive classification oracle has been shown effective in transductive settings (Kakade and Kalai, 2005). Aside from the drawback of requiring a transductive setting, the regret achieved there is substantially worse than for EXP4.

2. Improved Rates. When the world is not completely adversarial, it is possible to achieve substantially lower regrets than are possible with algorithms optimized for the adversarial setting. For example, in supervised learning, it is possible to obtain regrets scaling as $O(\log(T))$ with a problem dependent constant (Bartlett et al., 2007). 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, it is possible to achieve an additive regret of τ (Langford et al., 2009).

In a direct i.i.d. setting, the previous-best approach using a cost-sensitive classification oracle was given by ϵ-greedy and epoch greedy algorithms (Langford and Zhang, 2007) which have a regret scaling as $O(T^{2/3})$ in the worst case.

There have also been many special-case analyses. For example, theory of context-free setting is well understood (Lai and Robbins, 1985; Auer et al., 2002a; Even-Dar et al., 2006). Similarly, good algorithms exist when rewards are linear functions of features (Auer, 2002) or actions lie in a continuous space with the reward function sampled according to a Gaussian process (Srinivas et al., 2010).

1.2 WHAT WE PROVE

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

Theorem 4. For all distributions D over $(x, \vec{r})$ with K actions, for all sets of N policies Π, with probability at least $1-\delta$, the regret of PolicyElimination (Algorithm 1) over T rounds is at most

$$16\sqrt{2TK\ln\frac{4T^2N}{\delta}}.$$

This result can be extended to deal with VC classes, as well as other special cases. It forms the simplest method we have of exhibiting the new analysis.

The new key element of this algorithm is identification of a distribution over actions which simultaneously achieves small expected regret and allows estimating value of every policy with small variance. The existence of such a distribution is shown *nonconstructively* by a minimax argument.

PolicyElimination is computationally intractable and also requires exact knowledge of the context distribution (but not the reward distribution!). We show how to address these issues in Section 4 using an algorithm we call RandomizedUCB. Namely, we prove the following theorem.

Theorem 5. For all distributions D over $(x,\vec{r})$ with K actions, for all sets of N policies Π, with probability at least $1-\delta$, the regret of RandomizedUCB (Algorithm 2) over T rounds is at most

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

RandomizedUCB's analysis is substantially more complex, with a key subroutine being an application of the ellipsoid algorithm with a cost-sensitive classification oracle (described in Section 5). RandomizedUCB does not assume knowledge of the context distribution, and instead works with the history of contexts it has observed. Modifying the proof for this empirical distribution requires a covering argument over the distributions over policies which uses the probabilistic method. The net result is an algorithm with a similar top-level analysis as PolicyElimination, but with the running time only poly-logarithmic in the number of policies given a cost-sensitive classification oracle.

Theorem 11. In each time step t, RandomizedUCB makes at most $O(\mathrm{poly}(t,K,\log(1/\delta),\log N))$ calls to cost-sensitive classification oracle, and requires additional $O(\mathrm{poly}(t,K,\log N))$ processing time.

Apart from a tractable algorithm, our analysis can be used to derive tighter regrets than would be possible in adversarial setting. For example, in Section 6, we consider a common setting where reward feedback is delayed by τ rounds. A straightforward modification of PolicyElimination yields a regret with an *additive* term proportional to τ compared with the delay-free setting. Namely, we prove the following.

Theorem 12. For all distributions D over $(x,\vec{r})$ with K actions, for all sets of N policies Π, and all delay intervals τ, with probability at least $1-\delta$, the regret of DelayedPE (Algorithm 3) is at most

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

We start next with precise settings and definitions.

2 SETTING AND DEFINITIONS

2.1 THE SETTING

Let A be the set of K actions, let X be the domain of contexts x, and let D be an arbitrary joint distribution on $(x,\vec{r})$. We denote the marginal distribution of D over X by D_X.

We denote Π to be a finite set of policies $\{\pi : X\to A\}$, where each policy π, given a context x_t in round t, chooses the action $\pi(x_t)$. The cardinality of Π is denoted by N. 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 $t=1\ldots T$, the world chooses $(x_t,\vec{r}_t)$ i.i.d. according to D and reveals x_t to the learner. The learner, having access to Π, chooses action $a_t\in\{1,\ldots,K\}$. Then the world reveals reward $r_t(a_t)$ (which we call r_t for short) to the learner, and the interaction proceeds to the next round.

We consider two modes of accessing the set of policies Π. The first option is through the enumeration of all policies. This is impractical in general, but suffices for the illustrative purpose of our first algorithm. The second option is an oracle access, through an *argmax oracle*, corresponding to a cost-sensitive learner:

Definition 1. For a set of policies Π, an argmax oracle ($\mathcal{AMO}$ for short), is an algorithm, which for any sequence $\{(x_{t'},\vec{r}_{t'})\}_{t'=1\ldots t}$, $x_{t'}\in X$, $\vec{r}_{t'}\in\mathbb{R}^K$, computes

$$\arg\max_{\pi\in\Pi}\sum_{t'=1\ldots t} r_{t'}(\pi(x_{t'}))\ .$$

The reason why the above can be viewed as a cost-sensitive classification oracle is that vectors of rewards $\vec{r}_{t'}$ can be interpreted as negative costs and hence the policy returned by $\mathcal{AMO}$ is the optimal cost-sensitive classifier on the given data.

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 $\pi_{\max} \in \Pi$ is that which maximizes $\eta_D(\pi)$. More formally,

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

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

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

where $p_{t'}$ is the probability of the algorithm choosing action $a_{t'}$ at time t'. Note that $a_{t'}$ and $p_{t'}$ are produced by the learner while $x_{t'}, r_{t'}$ are produced by nature. We write $x \sim h$ to denote choosing x uniformly at random from the x's in history h.

Using the history of past actions and probabilities with which they were taken, we can form an unbiased estimate of the policy value for any $\pi \in \Pi$:

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

The unbiasedness follows, because $\mathbb{E}_{a\sim p} \frac{r\mathbb{I}(\pi(x)=a)}{p(a)} = \sum_a p(a) \frac{r\mathbb{I}(\pi(x)=a)}{p(a)} = r(\pi(x))$. The empirically best policy at time t is denoted

$$\pi_t \doteq \operatorname*{argmax}_{\pi \in \Pi} \eta_t(\pi).$$

2.3 REGRET

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

$$\sum_{t=1\ldots T} (\eta_D(\pi_{\max}) - r_t) \, . \qquad (2.1)$$

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

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

where the probability is taken with respect to the random pairs $(x_t, \vec{r}_t) \sim D$ for $t = 1 \ldots T$, as well as any internal randomness used by the learner.

We can also define notions of regret and empirical regret for policies π. For all $\pi \in \Pi$, let

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

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:

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

In general, we shall use W, W' and Z as conditional probability distributions over the actions A given contexts X, i.e., $W : X \times A \to [0, 1]$ such that $W(x, \cdot)$ is a probability distribution over A (and similarly for W' and Z). We shall think of W' as a smoothed version of W with a minimum action probability of μ (to be defined by the algorithm), such that

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

Conditional distributions such as W (and W', Z, etc.) correspond to randomized policies. We define notions true and empirical value and regret for them as follows:

$$\eta_D(W) \doteq \mathop{\mathbb{E}}_{(x,\vec{r})\sim D} [\vec{r} \cdot W(x)]$$
$$\eta_t(W) \doteq \frac{1}{t} \sum_{(x,a,r,p)\in h_t} \frac{rW(x,a)}{p}$$
$$\Delta_D(W) \doteq \eta_D(\pi_{\max}) - \eta_D(W)$$
$$\Delta_t(W) \doteq \eta_t(\pi_t) - \eta_t(W) \ .$$

3 POLICY ELIMINATION

The basic ideas behind our approach are demonstrated in our first algorithm: PolicyElimination (Algorithm 1).

The key step is Step 1, which finds a distribution over policies which induces low variance in the estimate of the value of all policies. Below we use minimax theorem to show that such a distribution always exists. How to find this distribution is not specified here, but in Section 5 we develop a method based on the ellipsoid algorithm. Step 2 then projects this distribution onto a distribution over actions and applies smoothing. Finally, Step 5 eliminates the policies that have been determined to be suboptimal (with high probability).

ALGORITHM ANALYSIS

We analyze PolicyElimination in several steps. First, we prove the existence of P_t in Step 1, provided that Π_{t-1} is non-empty. We recast the feasibility problem in Step 1 as a game between two players: Prover, who is trying to produce P_t, and Falsifier, who is trying to find π violating the constraints. We give more power to Falsifier and allow him to choose a distribution over π (i.e., a randomized policy) which would violate the constraints.

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

```
Let $\Pi_0 = \Pi$ and history $h_0 = \emptyset$
Define: $\delta_t \doteq \delta \,/\, 4Nt^2$
Define: $b_t \doteq 2\sqrt{\frac{2K\ln(1/\delta_t)}{t}}$
Define: $\mu_t \doteq \min\left\{\frac{1}{2K}, \sqrt{\frac{\ln(1/\delta_t)}{2Kt}}\right\}$
For each timestep $t = 1 \dots T$, observe $x_t$ and do:
    1. Choose distribution $P_t$ over $\Pi_{t-1}$ 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 $\Pi_t = \Big\{\pi \in \Pi_{t-1} :$
           $\eta_t(\pi) \ge \Big(\max_{\pi'\in\Pi_{t-1}} \eta_t(\pi')\Big) - 2b_t\Big\}$
    6. Let $h_t = h_{t-1} \cup (x_t, a_t, r_t, W'_t(a_t))$
```

Note that any policy π corresponds to a point in the space of randomized policies (viewed as functions $X \times A \to [0,1]$), with $\pi(x,a) \doteq \mathbb{I}(\pi(x) = a)$. For any distribution P over policies in Π_{t-1}, the induced randomized policy W_P then corresponds to a point in the convex hull of Π_{t-1}. Denoting the convex hull of Π_{t-1} by $\mathcal{C}$, Prover's choice by W and Falsifier's choice by Z, the feasibility of Step 1 follows by the following lemma:

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,*

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

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*: Since $W'(x,a) \ge \mu$, we obtain that $1/W'(x,a) \in [0, 1/\mu]$, hence the expectations are defined for all W and Z.
- *linear in Z*: Linearity follows from rewriting $f(W,Z)$ as

$$f(W,Z) = \mathop{\mathbb{E}}_{x\sim D_X} \sum_{a\in A} \left[\frac{Z(x,a)}{W'(x,a)}\right] .$$

- *convex in W*: Note that $1/W'(x,a)$ is convex in $W(x,a)$ by convexity of $1/(c_1 w + c_2)$ in $w \ge 0$, for $c_1 \ge 0$, $c_2 > 0$. Convexity of $f(W,Z)$ in W then follows by taking expectations over x and a.

Hence, by Theorem 14 (in Appendix B), min and max can be reversed without affecting the value:

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

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

$$\begin{aligned} f(Z,Z) &= \mathop{\mathbb{E}}_{x\sim D_X} \sum_{a\in A} \left[\frac{Z(x,a)}{Z'(x,a)}\right] \\ &\le \mathop{\mathbb{E}}_{x\sim D_X} \sum_{\substack{a\in A:\\ Z(x,a)>0}} \left[\frac{Z(x,a)}{(1-K\mu)Z(x,a)}\right] = \frac{K}{1-K\mu} \ . \qquad \square \end{aligned}$$

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

Given the existence of P_t, we will see below that the constraints in Step 1 ensure low variance of the policy value estimator $\eta_t(\pi)$ for all $\pi \in \Pi_{t-1}$. The small variance is used to ensure accuracy of policy elimination in Step 5 as quantified in the following lemma:

Lemma 3. *With probability at least $1-\delta$, for all t:*

1. *$\pi_{\max} \in \Pi_t$ (i.e., Π_t is non-empty)*
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 $2\delta_t$. Taking the union bound over all policies and all time steps we find that with probability at least $1-\delta$,

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

for all t and all $\pi \in \Pi_{t-1}$. Then:

1. By the triangle inequality, in each time step, $\eta_t(\pi) \le \eta_t(\pi_{\max}) + 2b_t$ for all $\pi \in \Pi_{t-1}$, yielding the first part of the lemma.
2. Also by the triangle inequality, if $\eta_D(\pi) < \eta_D(\pi_{\max}) - 4b_t$ for $\pi \in \Pi_{t-1}$, then $\eta_t(\pi) < \eta_t(\pi_{\max}) - 2b_t$. Hence the policy π is eliminated in Step 5, yielding the second part of the lemma.

It remains to show Eq. (3.1). We fix the policy $\pi \in \Pi$ and time t, and show that the deviation bound is violated with probability at most $2\delta_t$. Our argument rests on Freedman's inequality (see Theorem 13 in Appendix A). Let

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

i.e., $\eta_t(\pi) = (\sum_{t'=1}^{t} y_{t'})/t$. Let $\mathbb{E}_t$ denote the conditional expectation $\mathbb{E}[\ \cdot \mid h_{t-1}]$. 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 the bound

$$0 \leq y_t \leq 1/\mu_t \doteq R_t \ .$$

Next,

$$\begin{aligned}
\mathbb{E}_t[y_t^2] &= \mathop{\mathbb{E}}_{(x_t,\vec{r}_t)\sim D} \mathop{\mathbb{E}}_{a_t \sim W'_t} \left[y_t^2\right] \\
&= \mathop{\mathbb{E}}_{(x_t,\vec{r}_t)\sim D} \mathop{\mathbb{E}}_{a_t \sim W'_t} \left[\frac{r_t^2 \mathbb{I}(\pi(x_t) = a_t)}{W'_t(a_t)^2}\right] \\
&\leq \mathop{\mathbb{E}}_{(x_t,\vec{r}_t)\sim D} \left[\frac{W'_t(\pi(x_t))}{W'_t(\pi(x_t))^2}\right] && (3.2) \\
&= \mathop{\mathbb{E}}_{x_t \sim D} \left[\frac{1}{W'_t(\pi(x_t))}\right] \leq 2K \ . && (3.3)
\end{aligned}$$

where Eq. (3.2) follows by boundedness of r_t and Eq. (3.3) follows from the constraints in Step 1. Hence,

$$\sum_{t'=1\dots t} \mathbb{E}_{t'}[y_{t'}^2] \leq 2Kt \doteq V_t \ .$$

Since $(\ln t)/t$ is decreasing for $t \geq 3$, we obtain that μ_t is non-increasing (by separately analyzing $t = 1$, $t = 2$, $t \geq 3$). Let t_0 be the first t such that $\mu_t < 1/2K$. Note that $b_t \geq 4K\mu_t$, so for $t < t_0$, we have $b_t \geq 2$ 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

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

Hence, the assumptions of Theorem 13 are satisfied, and

$$\Pr\left[|\eta_t(\pi) - \eta_D(\pi)| \geq b_t\right] \leq 2\delta_t \ .$$

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

This immediately implies that the cumulative regret is bounded by

$$\begin{aligned}
\sum_{t=1\dots T} (\eta_D(\pi_{\max}) - r_t) &\leq 8\sqrt{2K \ln \frac{4NT^2}{\delta}} \sum_{t=1}^{T} \frac{1}{\sqrt{t}} \\
&\leq 16\sqrt{2TK \ln \frac{4T^2 N}{\delta}} && (3.4)
\end{aligned}$$

and gives us the following theorem.

Algorithm 2 RANDOMIZEDUCB(Π,δ,K)

```
Let $h_0 \doteq \emptyset$ be the initial history.
Define the following quantities:
    $C_t \doteq 2\log\left(\frac{Nt}{\delta}\right)$ and $\mu_t \doteq \min\left\{\frac{1}{2K}, \sqrt{\frac{C_t}{2Kt}}\right\}$.
For each timestep $t = 1 \dots T$, observe $x_t$ and do:
  1. Let $P_t$ be a distribution over $\Pi$ that approximately solves the optimization problem
         $\min_P \sum_{\pi\in\Pi} P(\pi)\Delta_{t-1}(\pi)$
         s.t. for all distributions $Q$ over $\Pi$ :
         $\mathop{\mathbb{E}}_{\pi\sim Q}\left[\frac{1}{t-1}\sum_{i=1}^{t-1}\frac{1}{(1-K\mu_t)W_P(x_i,\pi(x_i))+\mu_t}\right] \leq \max\left\{4K, \frac{(t-1)\Delta_{t-1}(W_Q)^2}{180C_{t-1}}\right\}$   (4.1)
     so that the objective value at $P_t$ is within $\varepsilon_{\mathrm{opt},t} = O(\sqrt{KC_t/t})$ of the optimal value, and so that each constraint is satisfied with slack $\leq K$.
  2. Let $W'_t$ be the distribution over $A$ given by
         $W'_t(a) \doteq (1-K\mu_t)W_{P_t}(x_t,a) + \mu_t$
     for all $a \in A$.
  3. Choose $a_t \sim W'_t$.
  4. Observe reward $r_t$.
  5. Let $h_t \doteq h_{t-1} \cup (x_t, a_t, r_t, W'_t(a_t))$.
```

Theorem 4. *For all distributions D over $(x,\vec{r})$ with K actions, for all sets of N policies Π, with probability at least $1-\delta$, the regret of* POLICYELIMINATION *(Algorithm 1) over T rounds is at most*

$$16\sqrt{2TK \ln \frac{4T^2 N}{\delta}} \ .$$

4 THE RANDOMIZED UCB ALGORITHM

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

1. The algorithm keeps explicit track of the space of good policies (like a version space), which is difficult to implement efficiently in general.

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 RandomizedUCB (or RUCB for short), an algorithm which we present and analyze in this section. Our approach is reminiscent of the UCB algorithm (Auer et al., 2002a), developed for context-free setting, which keeps an upper-confidence bound on the expected reward for each action. However, instead of choosing the highest upper confidence bound, we randomize over choices according to the value of their empirical performance. The algorithm has the following properties:

1. The optimization step required by the algorithm always considers the full set of policies (i.e., explicit tracking of the set of good policies is avoided), and thus it can be efficiently implemented using an argmax oracle. We discuss this further 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 7 below.
3. Instead of D_X, the algorithm uses the history of previously seen contexts. The effect of this approximation is quantified in Theorem 6 below.

The regret of RandomizedUCB is the following:

Theorem 5. *For all distributions D over $(x, \vec{r})$ with K actions, for all sets of N policies Π, with probability at least $1-\delta$, the regret of* RandomizedUCB *(Algorithm 2) over T rounds is at most*

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

The proof is given in Appendix D.4 (in the full version). Here, we present an overview of the analysis.

4.1 EMPIRICAL VARIANCE ESTIMATES

A key technical prerequisite for the regret analysis is the accuracy of the 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 $V_{P,\pi,t}$ is (a bound on) 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 $\{x_1,\ldots,x_{t-1}\} \subseteq X$ drawn from D_X. We show that for all distributions P and all $\pi \in \Pi$, $\widehat{V}_{P,\pi,t}$ is close to $V_{P,\pi,t}$ with high probability.

Theorem 6. *For any $\epsilon \in (0,1)$, with probability at least $1-\delta$,*

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

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

The proof appears in Appendix C (in the full version).

4.2 REGRET ANALYSIS

Central to the analysis is the following lemma that bounds the value of the optimization in each round. It is a direct corollary of Lemma 24 in Appendix D.4 (in the full version).

Lemma 7. *If OPT_t is the value of the optimization problem (4.1) in round t, then*

$$\mathrm{OPT}_t \;\le\; O\left(\sqrt{\frac{KC_{t-1}}{t-1}}\right) \;=\; O\left(\sqrt{\frac{K\log(Nt/\delta)}{t}}\right).$$

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 bulk of the analysis consists of analyzing the variance of the importance-weighted reward estimates $\eta_t(\pi)$, and showing how they relate to their actual expected rewards $\eta_D(\pi)$. The details are deferred to Appendix D (in the full version).

5 USING AN ARGMAX ORACLE

In this section, we show how to solve the optimization problem (4.1) using the argmax oracle ($\mathcal{AMO}$) for our set of policies. Namely, we describe an algorithm running in polynomial time *independent*[1] of the number of policies, which makes queries to $\mathcal{AMO}$ to compute a

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

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

We do not describe the ellipsoid algorithm here (since it is standard), but only spell out its key properties in the following lemma. For a point $x \in \mathbb{R}^n$ and $r \geq 0$, we use the notation $B(x, r)$ to denote 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 or not. We are given a separation oracle for S and two numbers R and r, such that $S \in B(0, R)$ and if S is non-empty, then there is a point $x^\star$ such that $S \supseteq B(x^\star, r)$. The ellipsoid algorithm decides correctly if S is empty or not, by executing 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.*

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 $(x, a) \in \mathcal{X}_{t-1} \times A$. With abuse of 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 $\mathcal{C}$, i.e., $W_P(x, a) = \sum_{\pi:\pi(x)=a} P(\pi)$, and that $W'(x, a) = (1 - \mu_t K)W(x, a) + \mu_t$, where μ_t is as defined in Algorithm 2. Also define $\beta_t = \frac{t-1}{180C_{t-1}}$. 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 following convex program:

$$\min\ s \text{ s.t.}$$

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

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

$$\forall Z \in \mathcal{C}:$$

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

We claim 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. This can be seen as follows. Since we require $W \in \mathcal{C}$, it can be interpreted as being equal to W_P for some distribution over policies P. The constraints (5.3) are equivalent to (4.1) by substitution $Z = W_Q$.

The above convex program can be solved by performing a binary search over s and testing feasibility of the constraints. For a fixed value of s, the feasibility problem defined by (5.1)–(5.3) is 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. We leave the details to Appendix E (in the full version). 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 $\mathcal{AMO}$.*

Proof. The sequence for $\mathcal{AMO}$ consists of $x_{t'} \in \mathcal{X}_{t-1}$ and $\vec{r}_{t'}(a) = w(x_{t'}, a)$. The lemma now follows since $w \cdot \pi = \sum_{x\in\mathcal{X}_{t-1}} w(x, \pi(x))$. □

We need another simple technical lemma which 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 consider the convex set K defined 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.*

Proof. Let $g(x) = f(y) + \nabla f(y) \cdot (x - y)$. By the convexity of f, we have $f(x) \geq g(x)$ for all x. Thus, for any $x \in K$, we have $g(x) \leq f(x) \leq 0$. Since $g(y) = f(y) > 0$, we conclude that $g(x) = 0$ 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 the perceptron algorithm with all points $\pi \in \Pi$ being positive examples. The perceptron algorithm aims to find a hyperplane putting all policies $\pi \in \Pi$ on one side. In each iteration of the perceptron algorithm, we have a candidate hyperplane (specified by its normal vector), 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 (depending on the distance of W from $\mathcal{C}$, and the maximum magnitude $\|\pi\|_2$) we obtain a separating hyperplane. In passing we also note that if $W \in \mathcal{C}$, the same technique allows us to explicitly compute an approximate convex combination of policies in Π that yields W. This is done by running the perceptron algorithm as before and stopping after the bound on the number of iterations has been reached. Then we collect all the policies we have found in the run of the perceptron algorithm, and we are guaranteed that W is close in distance to their convex hull. We can then find the closest point in the convex hull of these policies by solving a simple quadratic program.

3. Finally, we consider constraint (5.3). We rewrite $\eta_{t-1}(W)$ as $\eta_{t-1}(W) = w \cdot W$, where $w(x_{t'}, a) = r_{t'}\mathbb{I}(a = a_{t'})/W'_{t'}(a_{t'})$. 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 a 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.

   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. This completes the description of the separation oracle.

Working out the details carefully yields the following theorem, proved in Appendix E (in the full version):

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

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

where $\epsilon = \frac{8\delta}{\mu_t^2}$ and $\gamma = \frac{\delta}{\mu_t}$.

6 DELAYED FEEDBACK

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

1. The world presents features x_t.
2. The learning algorithm 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 $x_{t-\tau}$.

We deal with delay by suitably modifying Algorithm 1 to incorporate the delay τ, giving Algorithm 3.

Now we can prove the following theorem, which shows the delay has an additive effect on regret.

Theorem 12. *For all distributions D over $(x, \vec{r})$ with K actions, for all sets of N policies Π, and all delay intervals τ, with probability at least $1-\delta$, the regret of* DelayedPE *(Algorithm 3) is at most*

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

Algorithm 3 DelayedPE(Π,δ,K,D_X,τ)

```
Let $\Pi_0 = \Pi$ and history $h_0 = \emptyset$
Define: $\delta_t \doteq \delta \,/\, 4Nt^2$ and $b_t \doteq 2\sqrt{\frac{2K\ln(1/\delta_t)}{t}}$
Define: $\mu_t \doteq \min\left\{\frac{1}{2K}, \sqrt{\frac{\ln(1/\delta_t)}{2Kt}}\right\}$
For each timestep $t = 1 \ldots T$, observe $x_t$ and do:
  1. Let $t' = \max(t - \tau, 1)$.
  2. Choose distribution $P_t$ over $\Pi_{t-1}$ s.t. $\forall\, \pi \in \Pi_{t-1}$:
       $\mathop{\mathbb{E}}_{x\sim D_X}\left[\frac{1}{(1-K\mu_{t'})W_{P_t}(x,\pi(x)) + \mu_{t'}}\right] \le 2K$
  3. $\forall\, a \in A$, Let $W'_t(a) = (1-K\mu_{t'})W_{P_t}(x_t,a) + \mu_{t'}$
  4. Choose $a_t \sim W'_t$
  5. Observe reward $r_t$.
  6. Let $\Pi_t = \Big\{\pi \in \Pi_{t-1}$ :
       $\eta_h(\pi) \ge \Big(\max_{\pi'\in\Pi_{t-1}} \eta_h(\pi')\Big) - 2b_{t'}\Big\}$
  7. Let $h_t = h_{t-1} \cup (x_t, a_t, r_t, W'_t(a_t))$
```

Proof. Essentially as Theorem 4. The variance bound is unchanged because it depends only on the context distribution. Thus, it suffices to replace $\sum_{t-1}^{T}\frac{1}{\sqrt{t}}$ with $\tau + \sum_{t=\tau+1}^{T+\tau}\frac{1}{\sqrt{t-\tau}} = \tau + \sum_{t=1}^{T}\frac{1}{\sqrt{t}}$ in Eq. (3.4). $\square$

Acknowledgements

We thank Alina Beygelzimer, who helped in several formative discussions.

References

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

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

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

P. L. Bartlett, E. Hazan, and A. Rakhlin. Adaptive online gradient descent. In *NIPS*, 2007.

Alina Beygelzimer, John Langford, and Pradeep Ravikumar. Error correcting tournaments. In *ALT*, 2009.

Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual bandit algorithms with supervised learning guarantees. In *AISTATS*, 2011.

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

David A. Freedman. On tail probabilities for martingales. *Annals of Probability*, 3(1):100–118, 1975.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1): 119–139, 1997.

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

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

Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

J. Langford, A. Smola, and M. Zinkevich. Slow learners are fast. In *NIPS*, 2009.

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

Maurice Sion. On general minimax theorems. *Pacific J. Math.*, 8(1):171–176, 1958.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, 2010.

A Concentration Inequality

The following is an immediate corollary of Theorem 1 of (Beygelzimer et al., 2011). It can be viewed as a version of Freedman's Inequality (Freedman, 1975). Let $y_1, \ldots, y_T$ be a sequence of real-valued random variables. Let $\mathbb{E}_t$ denote the conditional expectation $\mathbb{E}[\,\cdot\,|\, y_1, \ldots, y_{t-1}]$ and $\mathbb{V}_t$ conditional variance.

Theorem 13 (Freedman-style Inequality)**.** *Let $V, R \in \mathbb{R}$ such that $\sum_{t=1}^{T}\mathbb{V}_t[y_t] \le V$, and for all t, $y_t - \mathbb{E}_t[y_t] \le R$. Then for any $\delta > 0$ such that $R \le \sqrt{V/\ln(2/\delta)}$, with probability at least $1-\delta$,*

$$\left|\sum_{t=1}^{T} y_t - \sum_{t=1}^{T}\mathbb{E}_t[y_t]\right| \le 2\sqrt{V\ln(2/\delta)}\ .$$

B Minimax Theorem

The following is a continuous version of Sion's Minimax Theorem (Sion, 1958, Theorem 3.4).

Theorem 14. *Let $\mathcal{W}$ and $\mathcal{Z}$ be compact and convex sets, and $f : \mathcal{W}\times\mathcal{Z} \to \mathbb{R}$ a function which for all $Z \in \mathcal{Z}$ is convex and continuous in W and for all $W \in \mathcal{W}$ is concave and continuous in Z. Then*

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