
Differentiable Meta-Learning of Bandit Policies

Craig Boutilier
Google Research

Chih-Wei Hsu
Google Research

Branislav Kveton
Google Research

Martin Mladenov
Google Research

Csaba Szepesvári
DeepMind / University of Alberta

Manzil Zaheer
Google Research

Abstract

Exploration policies in Bayesian bandits maximize the average reward over problem instances drawn from some distribution $\mathcal{P}$. In this work, we *learn* such policies for an unknown distribution $\mathcal{P}$ using samples from $\mathcal{P}$. Our approach is a form of meta-learning and exploits properties of $\mathcal{P}$ without making strong assumptions about its form. To do this, we parameterize our policies in a differentiable way and optimize them by policy gradients, an approach that is pleasantly general and easy to implement. We derive effective gradient estimators and propose novel variance reduction techniques. We also analyze and experiment with various bandit policy classes, including neural networks and a novel softmax policy. The latter has regret guarantees and is a natural starting point for our optimization. Our experiments show the versatility of our approach. We also observe that neural network policies can learn implicit biases expressed only through the sampled instances.

1 Introduction

A *stochastic bandit* [36, 9, 38] is an online learning problem where a *learning agent* sequentially pulls arms with stochastic rewards. The agent aims to maximize its expected cumulative reward over some horizon. It does not know the mean rewards of the arms a priori and learns them by pulling the arms. This induces the well-known *exploration-exploitation trade-off*: *explore*, and learn more about an arm; or *exploit*, and pull the arm with the highest estimated reward. In a clinical trial, the *arm* might be a treatment and its *reward* is the outcome of that treatment for a patient.

Bandit algorithms are typically designed to have *low regret*, worst-case or instance-dependent, for some problem class of interest to the algorithm designer [38]. While regret guarantees are reassuring, this approach often results in algorithms that are overly conservative, since they do not exploit the full properties of the problem class or objective. We explore an alternative view, which is to *learn* a bandit algorithm. Specifically, we assume that the agent has access to bandit instances sampled from an unknown distribution $\mathcal{P}$ and attempts to learn a bandit algorithm with a high *Bayes reward*, the average reward over the instances drawn from $\mathcal{P}$. In essence, we automate the learning of policies for *Bayesian bandits* [16]. Our approach can be viewed as a form of *meta-learning* [55, 56, 11, 12] with gradient ascent [25].

A classic approach to Bayesian bandits is to design Bayes optimal policies [27, 28], which take a simple form for specific priors $\mathcal{P}$. Our approach is more general, since it makes minimal assumptions about $\mathcal{P}$ and optimized policies. It is also more computationally efficient and easier to parallelize. However, we lose guarantees on Bayes optimality. Another line of work [48, 59, 49] bounds the Bayes regret of classic bandit policies. These policies also have instance-dependent regret bounds and thus are more conservative than our work, where we directly optimize the Bayes reward.

Overall, our aim is to make learning of bandit policies as straightforward as applying gradient descent to supervised learning problems. We take the following steps toward this goal. First, we carefully

formulate the problem of policy-gradient optimization of the Bayes reward of bandit policies. Second, we derive the reward gradient and propose novel baseline subtraction methods that reduce the variance of its empirical estimate. These methods are tailored to the bandit structure of our problem and are critical to making our approach practical. Third, we show how to differentiate several softmax bandit policies: Exp3, SoftElim, and neural networks with a softmax output layer. SoftElim is a new algorithm where the probability of pulling an arm is directly parameterized. We prove that its n -round regret is sublinear in n for any K -armed bandit, as in UCB1 [9] and Thompson sampling (TS) [54, 3]. However, unlike UCB1 and TS, SoftElim is easy to optimize. Finally, we evaluate our methodology empirically on a range of bandit problems, highlighting the versatility of our approach. We also show that neural network policies can learn interesting biases encoded in the prior distribution $\mathcal{P}$.

2 Setting

This section introduces Bayesian bandits [27, 16]. To ease exposition, we first introduce multi-armed bandits [36, 9, 38]. To simplify notation, we define $[n] = \{1, \dots, n\}$.

2.1 Stochastic Multi-Armed Bandit

A *stochastic multi-armed bandit* [36, 9, 38] is an online learning problem where a learning agent interacts with the environment by pulling K arms in n rounds. In round $t \in [n]$, the agent *pulls* arm $I_t \in [K]$ and observes its reward. We denote the *reward* of arm i in round t by $Y_{i,t}$ and the rewards of all arms in round t by $Y_t = (Y_{1,t}, \dots, Y_{K,t})$. We assume that the rewards are drawn i.i.d. over rounds as $Y_t \sim P$, where P is a joint probability distribution over rewards of all arms, with support $[0, 1]^K$. Since P fully characterizes the bandit problem, we call it a *problem instance*. We denote the realized rewards in all n rounds by $Y = (Y_t)_{t=1}^n \in [0, 1]^{K \times n}$.

At a high level, all agents for solving bandit problems can be viewed as policies that map the history of the agent to the pulled arm. The *history* of the agent at the beginning of round t is

$$H_t = (I_1, \dots, I_{t-1}, Y_{I_1,1}, \dots, Y_{I_{t-1},t-1}),$$

which is a vector of pulled arm indices and corresponding observations up to round t . The agent is a *randomized policy* $\pi_w(\cdot | H_t) \in \Delta_K$, where Δ_K is a K -dimensional probability simplex, $w \in \mathcal{W}$ are *policy parameters*, and $\mathcal{W}$ is the set of *feasible policy parameters*. In this notation, $\pi_w(i | H_t)$ is the probability that arm i is pulled in round t given history H_t and $I_t \sim \pi_w(\cdot | H_t)$. To simplify notation, we write π instead of π_w when the dependence on w does not have to be stressed. We also define $I_{i:j} = (I_\ell)_{\ell=i}^j$ and $I = I_{1:n}$.

The *expected n -round reward* of policy π in problem instance P is $r(n, P; \pi) = \mathbb{E}[\sum_{t=1}^n Y_{I_t,t} | P]$. Note that the expectation is over both realized rewards Y and pulled arms I . A typical goal is to maximize $r(n, P; \pi)$, which is equivalent to regret minimization. The *expected n -round regret* of policy π is defined as $R(n, P; \pi) = \mathbb{E}[\sum_{t=1}^n Y_{i_*(P),t} - Y_{I_t,t} | P]$, where $i_*(P) = \arg \max_{i \in [K]} \mu_i$ is the *optimal arm* in problem instance P with mean arm rewards $\mu = \mathbb{E}[Y_t | P]$ for any $t \in [n]$.

2.2 Bayesian Bandit

A *Bayesian bandit* [27, 16] is a bandit where a learning agent interacts with problem instances P drawn i.i.d. from a known prior distribution $\mathcal{P}$. In this work, the problem instance P is the K -armed bandit in Section 2.1. The learning agent interacts as follows. First, the problem instance is drawn as $P \sim \mathcal{P}$ and all rewards in it are sampled as $Y \sim P$. Then the agent starts interacting with instance P for n rounds, starting from round 1. The agent does not know P or Y .

The quality of Bayesian bandit policies is measured by their n -round Bayes reward and regret. The *n -round Bayes reward* of policy π is

$$r(n; \pi) = \mathbb{E}[r(n, P; \pi)] = \mathbb{E}[\sum_{t=1}^n Y_{I_t,t}]. \quad (1)$$

When compared to Section 2.1, the expectation is also over problem instances P . Our goal is to maximize $r(n; \pi_w)$ with respect to w . This is equivalent to minimizing the *n -round Bayes regret*

$$R(n; \pi) = \mathbb{E}[R(n, P; \pi)] = \mathbb{E}[\sum_{t=1}^n Y_{i_*(P),t} - Y_{I_t,t}], \quad (2)$$

Algorithm 1 Gradient-based optimization of bandit policies.

- 1: **Inputs:** Policy parameters $w_0 \in \mathcal{W}$, number of iterations L , learning rate α , and batch size m
 - 2: $w \leftarrow w_0$
 - 3: **for** $\ell = 1, \dots, L$ **do**
 - 4: **for** $j = 1, \dots, m$ **do**
 - 5: Sample $P^j \sim \mathcal{P}$; sample $Y^j \sim P^j$; and apply policy π_w to Y^j to get I^j
 - 6: Let $\hat{g}(n; \pi_w)$ be an estimate of $\nabla_w r(n; \pi_w)$ from $(Y^j)_{j=1}^m$ and $(I^j)_{j=1}^m$
 - 7: $w \leftarrow w + \alpha \hat{g}(n; \pi_w)$
 - 8: **Output:** Learned policy parameters w
-

where $i_*(P)$ is defined as in Section 2.1.

To clarify our setting, we give examples of a Bayesian bandit and a policy for solving it below.

Example 1 (Bernoulli bandit with a uniform beta prior). Let $\mu \in [0, 1]^K$ be a vector of arm means and $y \in [0, 1]^K$ be a vector of realized rewards. Then the reward distribution is $P(y) = \prod_{i=1}^K \text{Ber}(y_i; \mu_i)$ and the prior distribution over arm means is $\mathcal{P}(\mu) = \prod_{i=1}^K \text{Beta}(\mu_i; 1, 1)$.

Example 2 (ε -greedy policy). The algorithm is parameterized by $w \in [0, 1]$ and works as follows. It pulls the best empirical arm with probability $1 - w$ and a random arm with probability w . That is,

$$\pi_w(i \mid H_t) = (1 - w) \mathbb{1}\left\{i = \arg \max_{j \in [K]} \hat{\mu}_{j,t-1}\right\} + w/K,$$

where $\hat{\mu}_{i,t}$ is the empirical mean of arm i after t rounds.

The bandit policy π_w in Example 2 and our objective in (1) showcase the two-level character of our optimization problem. At the lower level, π_w adapts to an unknown problem instance P , because it is a function of the history H_t that depends on P . At the higher level, we optimize (1) through w , so that π_w adapts to P as efficiently as possible, on average over $P \sim \mathcal{P}$.

3 Bandit Policy Optimization

We propose a general and data-dependent algorithm for learning bandit policies. The key idea is to maximize the Bayes reward $r(n; \pi)$ by gradient ascent on sampled problem instances from $\mathcal{P}$. Our proposed algorithm, which we call GradBand, is presented in Algorithm 1. GradBand is initialized with policy parameters w_0 . In each iteration, parameters of the last policy w are updated by gradient ascent using $\hat{g}(n; \pi_w)$, an empirical estimate of the reward gradient at the last policy, $\nabla_w r(n; \pi_w)$. To compute $\hat{g}(n; \pi_w)$, we run the policy π_w on m sampled problem instances from $\mathcal{P}$. We denote the j -th sampled instance by P^j , its realized rewards by $Y^j \in [0, 1]^{K \times n}$, and the corresponding pulled arms by $I^j \in [K]^n$.

The per-iteration time complexity of GradBand is $O(Kmn)$, because in each iteration we sample m problem instances from $\mathcal{P}$ with horizon n and K arms, and run a policy in each of them. Technically speaking, $\hat{g}(n; \pi_w)$ depends only on Y^j and I^j , and I^j depends on P_j only through Y^j . Therefore, GradBand does not need to know the instances P_j or their distribution $\mathcal{P}$ to compute the gradient.

GradBand is general, data-dependent, and directly optimizes the Bayes reward $r(n; \pi)$. However, since $r(n; \pi)$ is a complex function of the adaptive bandit policy π and environment, it is difficult to provide meaningful guarantees on optimizing it. This is one reason why existing bandit papers analyze theoretically manageable regret upper bounds and not the regret. We provide the first such guarantee below.

Theorem 1. Consider a 2-armed Gaussian bandit where the reward of arm i in round t is $Y_{i,t} \sim \mathcal{N}(\mu_i, 1)$. Consider an explore-then-commit policy π_h with parameter $h \in \mathcal{W} = [1, \lfloor n/2 \rfloor]$ where each arm is explored $\bar{h} = \lfloor h \rfloor + Z$ times for $Z \sim \text{Ber}(h - \lfloor h \rfloor)$. Then for any prior distribution $\mathcal{P}$ over arm means $\mu \in \mathbb{R}^2$, we have that $r(n; \pi_h)$ is concave in h .

The claim is proved in Appendix A. The key insight is that $r(n; \pi_h)$ of the explore-then-commit policy in a 2-armed Gaussian bandit has a closed form, differentiable with respect to h . The randomization

in Theorem 1 is only needed to extend the policy to continuous exploration horizons h . In this case, GradBand enjoys the same convergence guarantees as gradient descent for convex functions.

GradBand is a meta-algorithm. To fully exploit its power, we must specify the policy π and be able to compute its empirical gradient $\hat{g}(n; \pi)$. In Section 4, we derive the gradient and show how to reduce its variance. In Section 5, we study several differentiable bandit policies. Before we proceed, we relate our objective and algorithm design to prior work.

Stochastic multi-armed bandits: Our objective, the maximization of $\mathbb{E} [\sum_{t=1}^n Y_{I_t, t}]$, differs from maximizing $\mathbb{E} [\sum_{t=1}^n Y_{I_t, t} | P]$ in any problem instance P , which is standard in bandits [36, 9, 38]. The latter objective is more demanding, since it requires optimizing equally for likely and unlikely instances $P \sim \mathcal{P}$. Our objective is more appropriate when $\mathcal{P}$ can be estimated from data and the average reward is preferred to guarding against worst-case failures.

Bayesian bandits: Early works on Bayesian bandits [27, 16, 28] derived Bayes optimal policies that required specific conjugate priors $\mathcal{P}$. We do not make any such assumptions on $\mathcal{P}$. However, we do lose Bayes optimality guarantees, as the optimal policy may not have the same form as π . Since GradBand differentiates policies, it can be computationally costly. Nevertheless, it is less costly and easier to parallelize than the computation of typical Bayes optimal policies (Section 6.2).

Reinforcement learning: Learning of policy π is also an instance of *reinforcement learning* (RL) [51], where the *state* in round t is history H_t , the *action* is the pulled arm I_t , and the *reward* is the reward of the pulled arm $Y_{I_t, t}$. The main challenge is that the number of dimensions in H_t increases linearly with round t . So any RL method that solves this problem must introduce some structure to deal with the *curse of dimensionality*. Since it is not clear what the shape of the value function might be, we opt for optimizing parametric bandit policies (Section 5) by policy gradients [60]. The main novelty in our application of policy gradients are baseline subtraction techniques that are tailored to the bandit structure of our problem.

4 Reward Gradient

Using relatively standard techniques, we show that the reward gradient has the following form.

Theorem 2. *Let $b_t : [K]^{t-1} \times [0, 1]^{K \times n} \rightarrow \mathbb{R}$ be any function of previous $t - 1$ pulled arms and all realized rewards, for any round $t \in [n]$. Then*

$$\nabla_w r(n; \pi_w) = \sum_{t=1}^n \mathbb{E} \left[\nabla_w \log \pi_w(I_t | H_t) \left(\sum_{s=t}^n Y_{I_s, s} - b_t(I_{1:t-1}, Y) \right) \right].$$

The claim is proved in Appendix B. The collection of functions $b = (b_t)_{t=1}^n$ in Theorem 2 is known as a *baseline* [60, 53] and its purpose is to reduce the variance of empirical gradient estimates. The baseline does not change the gradient, as long as it does not depend on future actions taken by policy π . In our case, this means that b_t can be any function of all past pulls $I_{1:t-1}$, all realized rewards Y , problem instance P , and policy parameters w ; although we do not make the last two explicit. The empirical gradient, for m sampled instances in GradBand, is

$$\hat{g}(n; \pi_w) = \frac{1}{m} \sum_{j=1}^m \sum_{t=1}^n \nabla_w \log \pi_w(I_t^j | H_t^j) \left(\sum_{s=t}^n Y_{I_s^j, s}^j - b_t(I_{1:t-1}^j, Y^j) \right), \quad (3)$$

where j indexes the j -th random experiment in GradBand.

Existing variance minimizing techniques (Section 7) are hard to apply to our problem because our state space H_t is at least exponential in n ; and the shape of the value function, the future regret as a function H_t , is unknown and likely hard to approximate. Therefore, we propose our own baselines.

Our first baseline is *no baseline* $b_t^{\text{NONE}}(I_{1:t-1}, Y) = 0$. Not surprisingly, b^{NONE} performs poorly, even when learning bandit policies at short horizons (Section 6.2).

Our second baseline is $b_t^{\text{OPT}}(I_{1:t-1}, Y) = \sum_{s=t}^n Y_{i_*(P), s}$, where $i_*(P)$ is the best arm in instance P , as defined in (2). This baseline is suitable for bandit policies with regret guarantees. Specifically, if the policy has a sublinear regret with a high probability for any P , $\sum_{t=1}^n Y_{i_*(P), t} - Y_{I_t, t} = o(n)$ and thus $\sum_{s=t}^n b_t(I_{1:t-1}, Y) - Y_{I_s, s} = o(n)$ for any $s \in [n]$; both with a high probability for any P .

One limitation of b^{OPT} is that the best arm may be unknown, for instance if GradBand was only given sampled realized rewards Y^j but not sampled instances P^j . This motivates our third baseline, which is the reward of an independent run of policy π . Let $(J_t)_{t=1}^n$ be the pulled arms in an independent run of π . Then $b_t^{\text{SELF}}(I_{1:t-1}, Y) = \sum_{s=t}^n Y_{J_s, s}$. Similarly to b^{OPT} , b^{SELF} is suitable for any policy that concentrates on a single arm over time. Unlike b^{OPT} , it does not rely on knowing the best arm.

5 Differentiable Bandit Algorithms

We assume that $\nabla_w \log \pi_w(I_t | H_t)$ in Theorem 2 exists, so that the bandit policy is differentiable. Unfortunately, it seems that only a few bandit policies satisfy this assumption. For instance, UCB algorithms [9, 20, 1] are not differentiable because $\pi_w(i | H_t) \in \{0, 1\}$ is a step function. While Thompson sampling [54, 3, 4] is randomized, $\pi_w(i | H_t)$ is induced by a hard maximization over randomized posterior mean rewards. Therefore, a unique gradient may not exist. Even if it does, $\pi_w(i | H_t)$ does not have a closed form and thus is hard to differentiate computationally efficiently.

In the rest of this section, we introduce three softmax designs that can be differentiated analytically and derive a gradient for each of them. To simplify notation, we use $\pi_{i,t} = \pi_w(i | H_t)$. We note that the ε -greedy policy [52] and Boltzmann exploration [52, 18] are also differentiable, although we do not study them here.

5.1 Algorithm Exp3

Exp3 [8] is a non-stochastic bandit algorithm that pulls arm i in round t with probability

$$\pi_{i,t} = w/K + (1-w) \exp[\eta S_{i,t}] / \sum_{j=1}^K \exp[\eta S_{j,t}], \quad (4)$$

where $S_{i,t} = \sum_{\ell=1}^{t-1} \mathbb{1}\{I_\ell = i\} \pi_{i,\ell}^{-1} Y_{i,\ell}$ is the inverse propensity score estimate of the cumulative reward of arm i in the first $t-1$ rounds, η is a learning rate, and w is a parameter that guarantees sufficient exploration. When rewards are $[0, 1]$, Exp3 attains $O(\sqrt{Kn})$ regret when $\eta = w/K$ and $w = \min\{1, \sqrt{K \log K} / \sqrt{(e-1)n}\}$. In this work, we optimize w by GradBand. When η is set as above, we get the following gradient.

Lemma 3. Define $\pi_{i,t}$ as in (4). Let $\eta = w/K$, $V_{i,t} = \exp[w S_{i,t}/K]$, and $V_t = \sum_{j=1}^K V_{j,t}$. Then

$$\nabla_w \log \pi_{i,t} = \frac{1}{\pi_{i,t}} \left[\frac{V_{i,t}}{V_t} \left[(1-w) \left[\frac{S_{i,t}}{K} - \sum_{j=1}^K \frac{V_{j,t}}{V_t} \frac{S_{j,t}}{K} \right] - 1 \right] + \frac{1}{K} \right].$$

The lemma is proved in Appendix B. Although Exp3 is differentiable, it is conservative in stochastic problems, even after we optimize w (Section 6). Thus we propose a new algorithm SoftElim.

5.2 Algorithm SoftElim

Our bandit algorithm works as follows. Let $\hat{\mu}_{i,t}$ be the empirical mean of arm i after t rounds and $T_{i,t}$ be the number of pulls of arm i after t rounds. Each arm is initially pulled once. Then, in round $t > K$, arm i is pulled with probability

$$\pi_{i,t} = \exp[-S_{i,t}/w^2] / \sum_{j=1}^K \exp[-S_{j,t}/w^2], \quad (5)$$

where $S_{i,t} = 2(\max_{j \in [K]} \hat{\mu}_{j,t-1} - \hat{\mu}_{i,t-1})^2 T_{i,t-1}$ is the *score* of arm i in round t and $w > 0$ is a tunable *exploration parameter*. Since $S_{i,t} \geq 0$ and $\pi_{i,t} \propto \exp[-S_{i,t}/w^2]$, higher values of w lead to more exploration. Also note that $\exp[-S_{i,t}/w^2] \in [0, 1]$. Therefore, our algorithm can be viewed as “soft” elimination [7] of arms with high scores and we call it SoftElim.

The key idea in the design of SoftElim is that an arm is unlikely to be pulled if it has been pulled “often” and its empirical mean is low relatively to the highest empirical mean. This follows from the definition of score $S_{i,t}$. More interestingly, when a suboptimal arm has been pulled “often” and has the highest empirical mean, the optimal arm is pulled proportionally to how much its empirical mean deviates from the actual mean. This latter property implies sufficient optimism and leads to a regret analysis (Appendix C).

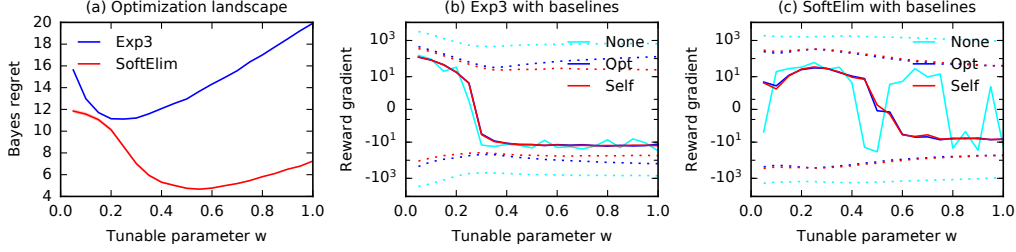


Figure 1: The Bayes regret of Exp3 and SoftElim, and the corresponding reward gradients. In the last two plots, the solid lines are estimated reward gradients for $m = 10\,000$ in (3) and the dotted lines mark high-probability regions for $m = 1$.

Since $\log \pi_{i,t} = -w^{-2}S_{i,t} - \log \sum_{j=1}^K \exp[-S_{j,t}/w^2]$, we have

$$\nabla_w \log \pi_{i,t} = 2w^{-3} \left(S_{i,t} - \sum_{j=1}^K S_{j,t} \exp[-S_{j,t}/w^2] / \sum_{j=1}^K \exp[-S_{j,t}/w^2] \right).$$

Therefore, SoftElim can be easily differentiated and optimized by GradBand. SoftElim also has a sublinear regret in any problem instance, as we show below.

Theorem 4. *Let P be any K -armed bandit problem where arm 1 is optimal, that is $\mu_1 > \max_{i>1} \mu_i$. Let $\Delta_i = \mu_1 - \mu_i$ and $w = \sqrt{8}$. Then $R(n, P; \pi_w) \leq \sum_{i=2}^K (2e + 1) (16\Delta_i^{-1} \log n + \Delta_i) + 5\Delta_i$.*

The proof of Theorem 4 is sketched in Appendix C.1 and presented in Appendix C.2. The value of $w = \sqrt{8}$ is obtained by tuning. An analogous bound, with worse constants, can be derived for any $w \in (1, \sqrt{8}]$. This can be seen from the proof, which only requires that $\gamma = w^{-2} \in [1/8, 1)$. Finally, note that our bound depends on gaps Δ_i and $\log n$ as UCB1 [9]. Thus it is near optimal.

5.3 Recurrent Neural Network

Now we take designs (4) and (5) a step further. In particular, as both are softmaxes on hand-crafted features, which allow theoretical analysis, we attempt to *learn the features* using a *recurrent neural network (RNN)*. The RNN works as follows. In round t , it takes arm I_t and reward $Y_{I_t,t}$ as inputs, updates its state s_t , and outputs the probability $\pi_{i,t+1}$ of pulling each arm i for round $t + 1$. That is,

$$s_t = \text{RNN}_\Phi(s_{t-1}, (I_t, Y_{I_t,t})), \quad \pi_{i,t+1} = \exp[v_i^\top s_t] / \sum_{j=1}^K \exp[v_j^\top s_t].$$

The optimized parameters $w = (\Phi, \{v_i\}_{i=1}^K)$ are the RNN parameters Φ and per-arm parameters v_i . The aim for the RNN is to learn to track suitable sufficient statistics through its internal state s_t . That state is initialized at $s_0 = 0$. Our RNN is an LSTM [30] with a d -dimensional latent state. We assume that the rewards are Bernoulli. The details of our implementation are in Appendix E.

6 Experiments

To demonstrate the generality and efficacy of our approach to learning bandit policies, we conduct four experiments. In Section 6.1, we study the reward gradient and its variance in a simple problem. In Section 6.2, we optimize Exp3 and SoftElim policies. In Section 6.3, we study the sensitivity of our approach to model and algorithm parameters. In Section 6.4, we optimize RNN policies.

The performance of policies is measured by the Bayes regret instead of the Bayes reward, since it directly indicates how near-optimal a policy is. Note that optimizing either optimizes the other. The regret is estimated from 1 000 i.i.d. sampled problem instances from $\mathcal{P}$, which are independent of those used by GradBand. The shaded areas in plots represent standard errors of our estimates. Our experiments are implemented in TensorFlow and PyTorch, on 112 cores and with 392 GB RAM.

6.1 Reward Gradient

Our first experiment is on a Bayesian bandit with $K = 2$ arms, Bernoulli reward distributions, and a horizon of $n = 200$ rounds. The prior distribution $\mathcal{P}$ is over two problem instances, $\mu = (0.6, 0.4)$

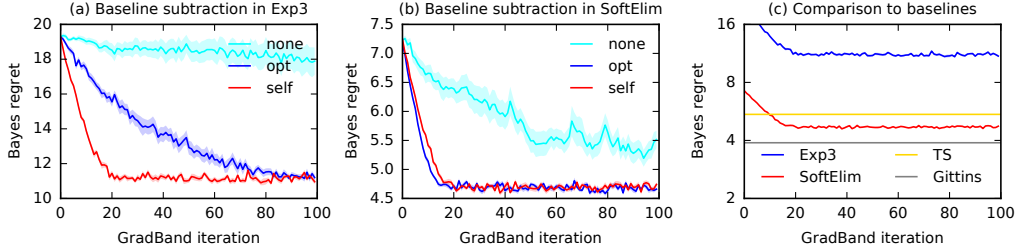


Figure 2: The Bayes regret of Exp3 and SoftElim policies, as a function of GradBand iterations. All experiments are on a Bernoulli bandit with $K = 2$ arms (Section 6.2). We report the average regret over 20 runs.

and $\mu = (0.4, 0.6)$, both with probability 0.5. The bandit policy π_w tries to identify the optimal arm, and thus to distinguish the instances, without knowing the instance that it interacts with. In this case, maximizing (1) with respect to w is akin to achieving this as fast as statistically possible, without incurring high regret on average. The symmetry of the problem instances is unimportant.

The Bayes regret of Exp3 and SoftElim, as a function of their parameter w , is shown in Figure 1a. Both are unimodal in w and suitable for optimization by GradBand. SoftElim has a lower regret than Exp3 for all w . In fact, the minimum regret of Exp3 is higher than that of SoftElim without tuning ($w = 1$). The reward gradients of Exp3 and SoftElim are reported in Figures 1b and 1c, respectively. We observe that baselines b^{OPT} and b^{SELF} lead to orders of magnitude lower variance than no baseline b^{NONE} . The variance of SoftElim gradients with b^{OPT} and b^{SELF} is comparable, while the variance of Exp3 gradients with b^{SELF} is two orders of magnitude lower for higher values of w .

6.2 Policy Optimization

In the second experiment, we apply Exp3 and SoftElim to the problem in Section 6.1. The policies are optimized by GradBand with $w_0 = 1$, $L = 100$ iterations, learning rate $\alpha = c^{-1}L^{-\frac{1}{2}}$, and batch size $m = 1000$. The constant c is chosen automatically so that $\|\hat{g}(n; \pi_{w_0})\| \leq c$ holds with a high probability, to avoid manual tuning of the learning rate in our experiments.

In Figure 2a, we optimize Exp3 with all baselines. With b^{SELF} , GradBand learns a near-optimal policy in fewer than 20 iterations. This is consistent with Figure 1b, where b^{SELF} has the lowest variance. In Figure 2b, we optimize SoftElim; and the performance with b^{OPT} and b^{SELF} is comparable. This is consistent with Figure 1c, where the variances of b^{OPT} and b^{SELF} are comparable. We conclude that b^{SELF} is the best baseline overall and use it in all remaining experiments.

To evaluate the quality of our policies, we compare them to four well-known bandit policies: UCB1 [9], Bernoulli TS [3] with Beta(1, 1) prior, UCB-V [6], and the Gittins index [27]. These benchmarks are ideal points of comparison: (i) UCB1 is arguably the most popular multi-armed bandit algorithm for $[0, 1]$ rewards. (ii) Bernoulli TS is near-optimal in Bernoulli bandits, which we use often in our experiments. We use randomized Bernoulli rounding [3] to apply TS to $[0, 1]$ rewards. (iii) UCB-V adapts the sub-Gaussian parameter of reward distributions based on observed rewards. This is similar to optimizing w in SoftElim. (iv) The Gittins index is the optimal solution to our problem, under the assumption that the mean arm rewards are drawn i.i.d. from Beta(1, 1). Finally, we also compare to the Dopamine [14] implementation of DQN [45] where the state is a concatenation of the following statistics for each arm: the number of observed ones, the number of observed zeros, the logarithm of both counts incremented by 1, the empirical mean, and a constant bias term.

The Bayes regret of our benchmarks is 9.95 ± 0.03 (UCB1), 5.47 ± 0.05 (TS), 15.79 ± 0.03 (UCB-V), 3.89 ± 0.07 (Gittins index), and 16.81 ± 1.05 (DQN). The regret of SoftElim is 4.74 ± 0.03 , and falls between those of TS and the Gittins index. The regret of Exp3 is 10.96 ± 0.16 , which is not competitive. The regret of our policies and most competitive baselines is shown in Figure 2c. We conclude that tuned SoftElim outperforms a strong baseline, TS; and performs almost as well as the Gittins index. We note that the Gittins index provides the optimal solution in limited settings, like Bernoulli bandits, but even there it is computationally costly. For instance, our computation of the Gittins index for horizon $n = 200$ took almost two days. In comparison, tuning of SoftElim by GradBand takes about 20 seconds.

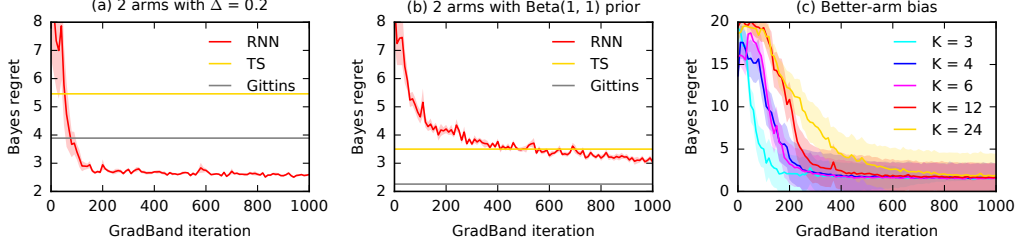


Figure 3: The Bayes regret of RNN policies, as a function of GradBand iterations. We report the average over 10 runs in the first two plots and the median in the last. The median excludes a few failed runs, which would skew the average.

Now we discuss failures of some benchmarks. UCB-V fails because its variance optimism induces too much initial exploration. This is harmful for somewhat short learning horizons in our experiments. DQN policies are unstable and require significant tuning to learn policies that outperform random actions; and still perform poorly. This is in stark contrast with the simplicity of GradBand, which learns near-optimal policies using gradient ascent. In the remaining experiments, we only discuss the most competitive benchmarks, the Gittins index and TS. We report the results for all benchmark bandit algorithms in Table 1 (Appendix D).

6.3 Supplementary Experiments

In Appendix D, we conduct additional experiments to validate our policies. We focus on two areas. In Appendix D.1, we study more challenging bandit problems. In particular, we increase horizons to $n = 1\,000$ rounds, increase the number of arms to $K = 10$, vary $\mathcal{P}$, and also experiment with beta noise. We confirm all findings from Figure 2c. We also find that tuned SoftElim can outperform the Gittins index by a large margin, in the instances where the Gittins index needs to be approximated. In Appendix D.2, we study the robustness of tuned SoftElim policies. We focus on two aspects of this problem. First, we show that the quality of the policies does not deteriorate significantly as we increase the horizon n and decrease the batch size m in GradBand. Second, we study the robustness of the policies to prior misspecification.

6.4 RNN Policies

Our preliminary experiments showed that learning of RNN policies (Section 5.3) over long horizons ($n = 200$) is challenging if we use our variance reduction baselines (Section 6.1) alone. To mitigate this, we propose the use of *curriculum learning* [15] to further reduce variance. The key idea is to apply GradBand successively to problems with increasing horizons. In this experiment, we consider a simple instance of this idea with two horizons: $n' = 20$ and $n = 200$. First, we optimize the RNN policy using GradBand at horizon n' . Then we take the learned policy and use it as the initial policy for GradBand optimization at horizon n . The number of GradBand iterations is $L = 1\,000$. We did not make any attempt to optimize this scheme.

Results from the second optimization phase are reported in Figure 3. Figure 3a shows learning of an RNN policy for the problem in Section 6.2. The RNN policy outperforms both TS and the Gittins index. This does not contradict theory, as the Gittins index is not Bayes optimal in this problem. In Figure 3b, we consider a variant of this problem where arm means are drawn i.i.d. from Beta(1, 1). The Gittins index is Bayes optimal in this problem, and so our learned RNN policy naturally does not outperform it. Nevertheless, it has a lower regret than TS.

In the final experiment, we have a K -armed Bayesian bandit with Bernoulli rewards. The prior $\mathcal{P}$ is over two problem instances, $\mu = (0.6, 0.9, 0.7, 0.7, \dots, 0.7)$ and $\mu = (0.2, 0.7, 0.9, 0.7, \dots, 0.7)$, which are equally likely. This problem has an interesting structure. The problem instance, and thus the optimal arm, can be easily identified by pulling arm 1. Arms 4 and beyond are *distractors*. Our RNN policies do not learn this structure; but they learn another strategy specialized to this problem. The strategy pulls only arms 2 or 3, since these are the only arms that can be optimal. Thus, the RNN successfully learns to ignore the distractors. As a result, the Bayes regret of our policies (Figure 3c) does not increase with K . This would not happen with classic bandit algorithms.

7 Related Work

The regret of bandit algorithms can be reduced by tuning [58, 40, 32, 31]. None of these works used policy gradients, neural network policies, or even the sequential structure of n -round rewards. Duan et al. [24] optimized a similar policy to Section 5.3 using an existing optimizer. This work does not formalize the objective clearly, relates it to Bayesian bandits, or studies policies that are provably sound (Theorem 1 and Section 5.2). Silver et al. [50] applied policy gradients to a continuous bandit problem with a quadratic cost function. Since their cost is convex in arms, this exploration problem is easier than with discrete arms.

Policy gradients in RL were proposed by Williams [60], including the idea of baseline subtraction. Other early works on this topic are Sutton et al. [53] and Baxter and Bartlett [13]. Policy gradients tend to have a high variance and reducing it is an important research area [29, 46, 64, 23, 39]. Our baselines differ from those in RL, in particular because our number of states H_t is not small. The baseline b^{OPT} uses the fact that we have a bandit problem, and thus the best arm in hindsight. Both b^{OPT} and b^{SELF} use the fact that we have access to all rewards, even of arms not pulled by policy π .

Our approach is an instance of meta-learning [55, 56], where we learn from a sample of tasks to perform well on tasks drawn from the same distribution [11, 12]. Meta-learning has been applied successfully in deep reinforcement learning (RL) [25, 26, 44]. Sequential multitask learning [17] was studied in multi-armed bandits by Azar et al. [10] and in contextual bandits by Deshmukh et al. [22]. In comparison, our setting is offline. A general template for sequential meta-learning was presented in Ortega et al. [47]. This work is conceptual and does not study policy gradients.

Maillard [41] proposed `SoftElim` with $w = 1$ and bounded the number of pulls of a suboptimal arm in Theorem 1.10. The bound has a large $O(K\Delta^{-4})$ constant, which does not seem easy to eliminate. We introduce w and have a tighter analysis (Theorem 4) with a $O(1)$ constant, for $w = 8$. Also note that `SoftElim` is not very competitive with TS without tuning. Therefore, this approach have not received much attention in the past, and this is the first work that makes it practical. The design of `SoftElim` resembles Boltzmann exploration [52, 18] and Exp3 (Section 5.1). The key difference is in how $S_{i,t}$ is chosen. In Exp3 and Boltzmann exploration, $S_{i,t}$ only depends on the history of arm i . In `SoftElim`, $S_{i,t}$ depends on all arms and makes `SoftElim` sufficiently optimistic.

8 Conclusions

We take first steps towards understanding policy-gradient optimization of bandit policies. Our work addresses two main challenges of this problem. First, we derive the reward gradient of optimized policies and show how to estimate it efficiently from a sample. Second, we propose differentiable bandit policies that can outperform state-of-the-art baselines after optimization. Our experiments highlight the simplicity and generality of our approach. We also show that neural network policies can learn interesting biases.

Our goal was to clearly show the benefits of learning to explore over the state of the art. Therefore, we focus on non-contextual problems, where the optimal policy can sometimes be computed and Thompson sampling is the state of the art. Our results can be extended to structured problems. An insight that permits the generalization of `SoftElim` is that $S_{i,t}$ in (5) is a ratio of two quantities, the squared empirical suboptimality gap of arm i and the variance of the mean reward estimate of arm i , $1/T_{i,t-1}$. Such quantities can be computed in linear bandits, for instance; and we hope to extend our work to even more general structures [34, 33, 19, 21, 57, 62]. Recently, Kveton et al. [35] and Yang and Toni [61] extended this work to contextual bandits, and Kveton et al. [35] and Min et al. [43] proposed differentiable Thompson sampling; thus resolving the open question in Section 5.

We leave open several questions of interest. First, we observe that the variance of empirical reward gradients can be high, especially in RNN policies. So any progress in variance reduction would be of a great importance. Second, except for Theorem 1, we are unaware of other bandit policy-instance pairs where the Bayes reward is concave in optimized parameters; and thus gradient ascent leads to optimal solutions. Our empirical results (Figure 1a) suggest that this phenomenon may be common. Finally, we believe that convergence guarantees for softmax exploration policies will be established in the future, based on recent advances in analyzing policy gradients in RL [2, 42].

Broader Impact

We develop bandit-style exploration algorithms that can be easily trained on historical data. This fundamentally alters the current prevalent approach to bandit algorithm design, which is by theory. Bandit algorithms are often deployed in user-facing domains. However, we do not propose any new domain. So our impact on users should be limited to those that already exist. Beyond this, we are not aware of any societal consequences of our work, such as on welfare, fairness, or privacy.

Acknowledgments and Disclosure of Funding

Csaba Szepesvári gratefully acknowledges funding from the Canada CIFAR AI Chairs Program, Amii, and NSERC.

References

- [1] Yasin Abbasi-Yadkori, David Pal, and Csaba Szepesvari. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems 24*, pages 2312–2320, 2011.
- [2] Alekh Agarwal, Sham Kakade, Jason Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in Markov decision processes. *CoRR*, abs/1908.00261, 2019. URL <http://arxiv.org/abs/1908.00261>.
- [3] Shipra Agrawal and Navin Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *Proceeding of the 25th Annual Conference on Learning Theory*, pages 39.1–39.26, 2012.
- [4] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *Proceedings of the 30th International Conference on Machine Learning*, pages 127–135, 2013.
- [5] V. M. Aleksandrov, V. I. Sysoyev, and V. V. Shemeneva. Stochastic optimization. *Engineering Cybernetics*, 5:11–16, 1968.
- [6] Jean-Yves Audibert, Remi Munos, and Csaba Szepesvari. Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19): 1876–1902, 2009.
- [7] Peter Auer and Ronald Ortner. UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010.
- [8] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 322–331, 1995.
- [9] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.
- [10] Mohammad Gheshlaghi Azar, Alessandro Lazaric, and Emma Brunskill. Sequential transfer in multi-armed bandit with finite set of models. In *Advances in Neural Information Processing Systems 26*, pages 2220–2228, 2013.
- [11] Jonathan Baxter. Theoretical models of learning to learn. In *Learning to Learn*, pages 71–94. Springer, 1998.
- [12] Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.
- [13] Jonathan Baxter and Peter Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

- [14] Marc Bellemare, Pablo Castro, Carles Gelada, Saurabh Kumar, and Subhodeep Moitra. Dopamine. <https://github.com/google/dopamine>, 2018.
- [15] Yoshua Bengio, Jerome Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*, pages 41–48, 2009.
- [16] Donald Berry and Bert Fristedt. *Bandit Problems: Sequential Allocation of Experiments*. Springer, 1985.
- [17] Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.
- [18] Nicolo Cesa-Bianchi, Claudio Gentile, Gabor Lugosi, and Gergely Neu. Boltzmann exploration done right. In *Advances in Neural Information Processing Systems 30*, pages 6284–6293, 2017.
- [19] Richard Combes, Stefan Magureanu, and Alexandre Proutiere. Minimal exploration in structured stochastic bandits. In *Advances in Neural Information Processing Systems 30*, pages 1763–1771, 2017.
- [20] Varsha Dani, Thomas Hayes, and Sham Kakade. Stochastic linear optimization under bandit feedback. In *Proceedings of the 21st Annual Conference on Learning Theory*, pages 355–366, 2008.
- [21] Remy Degenne, Han Shao, and Wouter Koolen. Structure adaptive algorithms for stochastic bandits. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.
- [22] Aniket Anand Deshmukh, Urun Dogan, and Clayton Scott. Multi-task learning for contextual bandits. In *Advances in Neural Information Processing Systems 30*, pages 4848–4856, 2017.
- [23] Travis Dick. Policy gradient reinforcement learning without regret. Master’s thesis, University of Alberta, 2015.
- [24] Yan Duan, John Schulman, Xi Chen, Peter Bartlett, Ilya Sutskever, and Pieter Abbeel. RL²: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779, 2016. URL <http://arxiv.org/abs/1611.02779>.
- [25] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135, 2017.
- [26] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems 31*, pages 9537–9548, 2018.
- [27] John Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41:148–177, 1979.
- [28] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-Armed Bandit Allocation Indices*. John Wiley & Sons, 2011.
- [29] Evan Greensmith, Peter Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530, 2004.
- [30] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [31] Chih-Wei Hsu, Branislav Kveton, Ofer Meshi, Martin Mladenov, and Csaba Szepesvari. Empirical Bayes regret minimization. *CoRR*, abs/1904.02664, 2019. URL <http://arxiv.org/abs/1904.02664>.
- [32] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *CoRR*, abs/1402.6028, 2014. URL <http://arxiv.org/abs/1402.6028>.
- [33] Branislav Kveton, Csaba Szepesvari, Zheng Wen, and Azin Ashkan. Cascading bandits: Learning to rank in the cascade model. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

- [34] Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvari. Tight regret bounds for stochastic combinatorial semi-bandits. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, 2015.
- [35] Branislav Kveton, Martin Mladenov, Chih-Wei Hsu, Manzil Zaheer, Csaba Szepesvari, and Craig Boutilier. Differentiable meta-learning in contextual bandits. *CoRR*, abs/2006.05094, 2020. URL <http://arxiv.org/abs/2006.05094>.
- [36] T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.
- [37] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems 20*, pages 817–824, 2008.
- [38] Tor Lattimore and Csaba Szepesvari. *Bandit Algorithms*. Cambridge University Press, 2019.
- [39] Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu. Action-dependent control variates for policy optimization via Stein’s identity. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.
- [40] Francis Maes, Louis Wehenkel, and Damien Ernst. Meta-learning of exploration/exploitation strategies: The multi-armed bandit case. In *Proceedings of the 4th International Conference on Agents and Artificial Intelligence*, pages 100–115, 2012.
- [41] Odalric-Ambrym Maillard. *APPRENTISSAGE SEQUENTIEL: Bandits, Statistique et Renforcement*. PhD thesis, University of Lille, 2011.
- [42] Jincheng Mei, Chenjun Xiao, Csaba Szepesvari, and Dale Schuurmans. On the global convergence rates of softmax policy gradient methods. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.
- [43] Seungki Min, Ciamac Moallemi, and Daniel Russo. Policy gradient optimization of Thompson sampling policies. *CoRR*, abs/2006.16507, 2020. URL <http://arxiv.org/abs/2006.16507>.
- [44] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.
- [45] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL <http://arxiv.org/abs/1312.5602>.
- [46] Remi Munos. Geometric variance reduction in Markov chains: Application to value function and gradient estimation. *Journal of Machine Learning Research*, 7:413–427, 2006.
- [47] Pedro Ortega, Jane Wang, Mark Rowland, Tim Genewein, Zeb Kurth-Nelson, Razvan Pascanu, Nicolas Heess, Joel Veness, Alexander Pritzel, Pablo Sprechmann, Siddhant Jayakumar, Tom McGrath, Kevin Miller, Mohammad Gheshlaghi Azar, Ian Osband, Neil Rabinowitz, Andras Gyorgy, Silvia Chiappa, Simon Osindero, Yee Whye Teh, Hado van Hasselt, Nando de Freitas, Matthew Botvinick, and Shane Legg. Meta-learning of sequential strategies. *CoRR*, abs/1905.03030, 2019. URL <http://arxiv.org/abs/1905.03030>.
- [48] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.
- [49] Daniel Russo and Benjamin Van Roy. An information-theoretic analysis of Thompson sampling. *Journal of Machine Learning Research*, 17(68):1–30, 2016.
- [50] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, pages 387–395, 2014.
- [51] Richard Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

- [52] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
- [53] Richard Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057–1063, 2000.
- [54] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.
- [55] Sebastian Thrun. *Explanation-Based Neural Network Learning - A Lifelong Learning Approach*. PhD thesis, University of Bonn, 1996.
- [56] Sebastian Thrun. Lifelong learning algorithms. In *Learning to Learn*, pages 181–209. Springer, 1998.
- [57] Andrea Tirinzoni, Alessandro Lazaric, and Marcello Restelli. A novel confidence-based algorithm for structured bandits. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, 2020.
- [58] Joannes Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Proceedings of the 16th European Conference on Machine Learning*, pages 437–448, 2005.
- [59] Zheng Wen, Branislav Kveton, and Azin Ashkan. Efficient learning in large-scale combinatorial semi-bandits. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.
- [60] Ronald Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.
- [61] Kaige Yang and Laura Toni. Differentiable linear bandit algorithm. *CoRR*, abs/2006.03000, 2020. URL <http://arxiv.org/abs/2006.03000>.
- [62] Tong Yu, Branislav Kveton, Zheng Wen, Ruiyi Zhang, and Ole Mengshoel. Graphical models meet bandits: A variational Thompson sampling approach. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.
- [63] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. In *Advances in Neural Information Processing Systems 31*, pages 9793–9803, 2018.
- [64] Tingting Zhao, Hirotaka Hachiya, Gang Niu, and Masashi Sugiyama. Analysis and improvement of policy gradient estimation. In *Advances in Neural Information Processing Systems 24*, pages 262–270, 2011.

A Proof of Theorem 1

We show that the n -round Bayes reward of a randomized explore-then-commit policy in 2-armed Gaussian bandits is concave in the exploration horizon of the policy.

Theorem 1. *Consider a 2-armed Gaussian bandit where the reward of arm i in round t is $Y_{i,t} \sim \mathcal{N}(\mu_i, 1)$. Consider an explore-then-commit policy π_h with parameter $h \in \mathcal{W} = [1, \lfloor n/2 \rfloor]$ where each arm is explored $\bar{h} = \lfloor h \rfloor + Z$ times for $Z \sim \text{Ber}(h - \lfloor h \rfloor)$. Then for any prior distribution $\mathcal{P}$ over arm means $\mu \in \mathbb{R}^2$, we have that $r(n; \pi_h)$ is concave in h .*

Proof. We start with the *explore-then-commit* policy [37], which is parameterized by $h \in [\lfloor n/2 \rfloor]$ and works as follows. In the first $2h$ rounds, it explores and pulls each arm h times. Let $\hat{\mu}_{i,h}$ be the average reward of arm i after h pulls. Then, if $\hat{\mu}_{1,h} > \hat{\mu}_{2,h}$, arm 1 is pulled for the remaining $n - 2h$ rounds. Otherwise arm 2 is pulled.

Fix any problem instance $P \sim \mathcal{P}$. Without loss of generality, let arm 1 be optimal, that is $\mu_1 > \mu_2$. Let $\Delta = \mu_1 - \mu_2$. The key observation is that the expected n -round reward in problem instance P has a closed form

$$r(n, P; \pi_h) = \mu_1 n - \Delta [h + \mathbb{P}(\hat{\mu}_{1,h} < \hat{\mu}_{2,h}) (n - 2h)] , \quad (6)$$

where

$$\begin{aligned} \mathbb{P}(\hat{\mu}_{1,h} < \hat{\mu}_{2,h}) &= \mathbb{P}(\hat{\mu}_{1,h} - \hat{\mu}_{2,h} < 0) = \mathbb{P}(\hat{\mu}_{1,h} - \hat{\mu}_{2,h} - \Delta < -\Delta) \\ &= \Phi\left(-\Delta\sqrt{h/2}\right) = \frac{1}{\sqrt{2\pi}} \int_{x=-\infty}^{-\Delta\sqrt{h/2}} e^{-\frac{x^2}{2}} dx \end{aligned} \quad (7)$$

is the probability of committing to a suboptimal arm after the exploration phase. The third equality is from the fact that $\hat{\mu}_{1,h} - \hat{\mu}_{2,h} - \Delta \sim \mathcal{N}(0, 2/h)$, where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution.

Our goal is to prove that $r(n, P; \pi_h)$ is concave in h . We rely on the following property of convex functions of a single parameter x . Let $f(x)$ and $g(x)$ be non-negative, decreasing, and convex in x . Then $f(x)g(x)$ is non-negative, decreasing, and convex in x . This follows from

$$\begin{aligned} (f(x)g(x))' &= f'(x)g(x) + f(x)g'(x) , \\ (f(x)g(x))'' &= f''(x)g(x) + 2f'(x)g'(x) + f(x)g''(x) . \end{aligned}$$

It is easy to see that (7) is non-negative, decreasing, and convex in h . The same is true for $n - 2h$, under our assumption that $h \in [\lfloor n/2 \rfloor]$. As a result, $\mathbb{P}(\hat{\mu}_{1,h} < \hat{\mu}_{2,h}) (n - 2h)$ is convex in h , and so is $\Delta[h + \mathbb{P}(\hat{\mu}_{1,h} < \hat{\mu}_{2,h}) (n - 2h)]$. Therefore, (6) is concave in h . Finally, the Bayes reward is concave in h because $r(n; \pi_h) = \mathbb{E}[r(n, P; \pi_h)]$.

The last remaining issue is that parameter h in the explore-then-commit policy cannot be optimized by GradBand, as it is discrete. To allow for optimization, we extend the explore-then-commit policy to continuous h by randomized rounding.

The *randomized explore-then-commit* policy is parameterized by continuous $h \in [1, \lfloor n/2 \rfloor]$. The discrete $\bar{h}$ is chosen as $\bar{h} = \lfloor h \rfloor + Z$, where $Z \sim \text{Ber}(h - \lfloor h \rfloor)$. Then we execute the original policy with $\bar{h}$. The key property of the randomized policy is that its n -round Bayes reward is a piecewise linear interpolation of that of the original policy,

$$(\lceil h \rceil - h) r(n; \pi_{\lfloor h \rfloor}) + (h - \lfloor h \rfloor) r(n; \pi_{\lceil h \rceil}) .$$

By definition, the above function is continuous and concave in h . Therefore, GradBand has the same guarantees for maximizing it as stochastic gradient descent on convex functions. $\square$

B Gradient Proofs

Theorem 2. Let $b_t : [K]^{t-1} \times [0, 1]^{K \times n} \rightarrow \mathbb{R}$ be any function of previous $t - 1$ pulled arms and all realized rewards, for any round $t \in [n]$. Then

$$\nabla_w r(n; \pi_w) = \sum_{t=1}^n \mathbb{E} \left[\nabla_w \log \pi_w(I_t | H_t) \left(\sum_{s=t}^n Y_{I_s, s} - b_t(I_{1:t-1}, Y) \right) \right].$$

Proof. The proof has two parts. First, we show that

$$\nabla_w r(n; \pi_w) = \sum_{t=1}^n \mathbb{E} \left[\nabla_w \log \pi_w(I_t | H_t) \sum_{s=t}^n Y_{I_s, s} \right]. \quad (8)$$

The n -round Bayes reward can be expressed as $r(n; \pi_w) = \mathbb{E} [\mathbb{E} [\sum_{t=1}^n Y_{I_t, t} | Y]]$, where the outer expectation is over instances P and their realized rewards Y , which are independent of w . Thus

$$\nabla_w r(n; \pi_w) = \mathbb{E} \left[\sum_{t=1}^n \nabla_w \mathbb{E} [Y_{I_t, t} | Y] \right].$$

In the inner expectation, only the pulled arms are random. Therefore, for any $t \in [n]$, we have

$$\mathbb{E} [Y_{I_t, t} | Y] = \sum_{i_{1:t}} \mathbb{P}(I_{1:t} = i_{1:t} | Y) Y_{i_t, t}.$$

The key to our derivations is that the joint probability distribution over pulled arms in the first t rounds, conditioned on Y , decomposes by the chain rule of probabilities as

$$\mathbb{P}(I_{1:t} = i_{1:t} | Y) = \prod_{s=1}^t \mathbb{P}(I_s = i_s | I_{1:s-1} = i_{1:s-1}, Y). \quad (9)$$

Since the policy does not act based on future rewards, we have for any $s \in [n]$ that

$$\mathbb{P}(I_s = i_s | I_{1:s-1} = i_{1:s-1}, Y) = \pi_w(i_s | i_{1:s-1}, Y_{i_1, 1}, \dots, Y_{i_{s-1}, s-1}). \quad (10)$$

Finally, we use that $\nabla_w f(w) = f(w) \nabla_w \log f(w)$ holds for any non-negative differentiable f . This identity is known as the *score-function identity* [5] and is the basis of all policy-gradient methods. We apply it to $\mathbb{E} [Y_{I_t, t} | Y]$ and obtain

$$\begin{aligned} \nabla_w \mathbb{E} [Y_{I_t, t} | Y] &= \sum_{i_{1:t}} Y_{i_t, t} \nabla_w \mathbb{P}(I_{1:t} = i_{1:t} | Y) \\ &= \sum_{i_{1:t}} Y_{i_t, t} \mathbb{P}(I_{1:t} = i_{1:t} | Y) \nabla_w \log \mathbb{P}(I_{1:t} = i_{1:t} | Y) \\ &= \sum_{s=1}^t \mathbb{E} [Y_{I_t, t} \nabla_w \log \pi_w(I_s | H_s) | Y], \end{aligned}$$

where the last equality follows from (9) and (10). Now we chain all equalities to obtain the *reward gradient*

$$\nabla_w r(n; \pi_w) = \sum_{t=1}^n \sum_{s=1}^t \mathbb{E} [Y_{I_t, t} \nabla_w \log \pi_w(I_s | H_s)] = \sum_{t=1}^n \mathbb{E} \left[\nabla_w \log \pi_w(I_t | H_t) \sum_{s=t}^n Y_{I_s, s} \right].$$

This concludes the first part of the proof.

Now we argue that b_t does not change anything. Since b_t depends only on $I_{1:t-1}$ and Y ,

$$\mathbb{E} [b_t(I_{1:t-1}, Y) \nabla_w \log \pi_w(I_t | H_t)] = \mathbb{E} [b_t(I_{1:t-1}, Y) \mathbb{E} [\nabla_w \log \pi_w(I_t | H_t) | I_{1:t-1}, Y]].$$

Now note that

$$\begin{aligned}
\mathbb{E} [\nabla_w \log \pi_w(I_t \mid H_t) \mid I_{1:t-1}, Y] &= \sum_{i=1}^K \mathbb{P}(I_t = i \mid I_{1:t-1}, Y) \nabla_w \log \pi_w(i \mid H_t) \\
&= \sum_{i=1}^K \pi_w(i \mid H_t) \nabla_w \log \pi_w(i \mid H_t) \\
&= \nabla_w \sum_{i=1}^K \pi_w(i \mid H_t) = 0.
\end{aligned}$$

The last equality follows from $\sum_{i=1}^K \pi_w(i \mid H_t) = 1$, which is a constant independent of w . This concludes the proof. $\square$

Lemma 3. Define $\pi_{i,t}$ as in (4). Let $\eta = w/K$, $V_{i,t} = \exp[wS_{i,t}/K]$, and $V_t = \sum_{j=1}^K V_{j,t}$. Then

$$\nabla_w \log \pi_{i,t} = \frac{1}{\pi_{i,t}} \left[\frac{V_{i,t}}{V_t} \left[(1-w) \left[\frac{S_{i,t}}{K} - \sum_{j=1}^K \frac{V_{j,t}}{V_t} \frac{S_{j,t}}{K} \right] - 1 \right] + \frac{1}{K} \right].$$

Proof. First, we express the derivative of $\log \pi_{i,t}$ with respect to w as

$$\nabla_w \log \pi_{i,t} = \frac{1}{\pi_{i,t}} \nabla_w \pi_{i,t} = \frac{1}{\pi_{i,t}} \left[(1-w) \nabla_w \frac{V_{i,t}}{V_t} - \frac{V_{i,t}}{V_t} + \frac{1}{K} \right].$$

Conditioned on the history, $S_{i,t}$ is a constant independent of w , and thus we have

$$\begin{aligned}
\nabla_w \frac{V_{i,t}}{V_t} &= \frac{1}{V_t} \nabla_w V_{i,t} + V_{i,t} \nabla_w \frac{1}{V_t} = \frac{V_{i,t} S_{i,t}}{V_t K} - \frac{V_{i,t}}{V_t^2} \sum_{j=1}^K V_{j,t} \frac{S_{j,t}}{K} \\
&= \frac{V_{i,t}}{V_t} \left[\frac{S_{i,t}}{K} - \sum_{j=1}^K \frac{V_{j,t}}{V_t} \frac{S_{j,t}}{K} \right].
\end{aligned}$$

This concludes the proof. $\square$

C Analysis of SoftElim

We informally justify SoftElim in Appendix C.1. Then we bound its regret in Appendix C.2.

C.1 Informal Analysis

We start with an informal argument in a 2-armed bandit. Let arm 1 be optimal, that is $\mu_1 > \mu_2$. Let $\Delta = \mu_1 - \mu_2$ be the gap. Fix any round t by which arm 2 has been pulled “often”, so that we get $T_{2,t-1} = \Omega(\Delta^{-2} \log n)$ and $\hat{\mu}_{2,t-1} \leq \mu_2 + \Delta/3$ with a high probability. Let

$$\hat{\mu}_{\max,t} = \max \{\hat{\mu}_{1,t}, \hat{\mu}_{2,t}\}.$$

Now consider two cases. First, when $\hat{\mu}_{\max,t-1} = \hat{\mu}_{1,t-1}$, arm 1 is pulled with probability of at least 0.5, by the definition of $\pi_{1,t}$. Second, when $\hat{\mu}_{\max,t-1} = \hat{\mu}_{2,t-1}$, we have

$$\pi_{1,t} = \exp[-2(\hat{\mu}_{2,t-1} - \hat{\mu}_{1,t-1})^2 T_{1,t-1}] \pi_{2,t} \geq \exp[-2(\mu_1 - \hat{\mu}_{1,t-1})^2 T_{1,t-1}] \pi_{2,t},$$

where the last inequality follows from $\hat{\mu}_{1,t-1} \leq \hat{\mu}_{2,t-1} \leq \mu_2 + \Delta/3 \leq \mu_1$ and holds with a high probability. This means that arm 1 is pulled “sufficiently often” relative to arm 2, proportionally to the deviation of $\hat{\mu}_{1,t-1}$ from μ_1 . Therefore, SoftElim eventually enters a regime where arm 1 has been pulled “often”, so that $T_{1,t-1} = \Omega(\Delta^{-2} \log n)$ and $\hat{\mu}_{1,t-1} \geq \mu_1 - \Delta/3$ holds with a high probability. Then both $S_{1,t} = 0$ and $S_{2,t} = \Omega(\log n)$ hold with a high probability, and arm 2 is unlikely to be pulled.

C.2 Regret Bound

We bound the n -round regret of SoftElim below.

Theorem 4. *Let P be any K -armed bandit problem where arm 1 is optimal, that is $\mu_1 > \max_{i>1} \mu_i$. Let $\Delta_i = \mu_1 - \mu_i$ and $w = \sqrt{8}$. Then $R(n, P; \pi_w) \leq \sum_{i=2}^K (2e+1) (16\Delta_i^{-1} \log n + \Delta_i) + 5\Delta_i$.*

Proof. Each arm is initially pulled once. Therefore,

$$R(n, P; \pi_w) = \sum_{i=2}^K \Delta_i \left(\sum_{t=K+1}^n \mathbb{P}(I_t = i) + 1 \right).$$

Now we decompose the probability of pulling each arm i as

$$\begin{aligned} \sum_{t=K+1}^n \mathbb{P}(I_t = i) &= \sum_{t=K+1}^n \mathbb{P}(I_t = i, T_{i,t-1} \leq m) + \\ &\quad \sum_{t=K+1}^n \mathbb{P}(I_t = i, T_{i,t-1} > m, T_{1,t-1} \leq m) + \\ &\quad \sum_{t=K+1}^n \mathbb{P}(I_t = i, T_{i,t-1} > m, T_{1,t-1} > m), \end{aligned}$$

where m is chosen later. In the rest of the proof, we bound each above term separately. To simplify notation, use $\gamma = 1/w^2$ in instead of w^2 .

C.3 Upper Bound on Term 1

Fix suboptimal arm i . Since $T_{i,t} = T_{i,t-1} + 1$ on event $I_t = i$ and arm i is initially pulled once, we have

$$\sum_{t=K+1}^n \mathbb{P}(I_t = i, T_{i,t-1} \leq m) \leq m - 1. \quad (11)$$

C.4 Upper Bound on Term 3

Fix suboptimal arm i and round t . Let

$$E_{1,t} = \left\{ \hat{\mu}_{1,t-1} > \mu_1 - \frac{\Delta_i}{4} \right\}, \quad E_{i,t} = \left\{ \hat{\mu}_{i,t-1} < \mu_i + \frac{\Delta_i}{4} \right\},$$

be the events that empirical means of arms 1 and i , respectively, are “close” to their means. Then

$$\begin{aligned} \mathbb{P}(I_t = i, T_{i,t-1} > m, T_{1,t-1} > m) \\ \leq \mathbb{P}(I_t = i, T_{i,t-1} > m, E_{1,t}) + \mathbb{P}(\bar{E}_{1,t}, T_{1,t-1} > m) \\ \leq \mathbb{P}(I_t = i, T_{i,t-1} > m, E_{1,t}, E_{i,t}) + \mathbb{P}(\bar{E}_{1,t}, T_{1,t-1} > m) + \mathbb{P}(\bar{E}_{i,t}, T_{i,t-1} > m). \end{aligned}$$

Let $m = \lceil 16\Delta_i^{-2} \log n \rceil$. By the union bound and Hoeffding’s inequality, we get

$$\begin{aligned} \mathbb{P}(\bar{E}_{1,t}, T_{1,t-1} > m) &\leq \sum_{s=m+1}^n \mathbb{P}\left(\mu_1 - \hat{\mu}_{1,t-1} \geq \frac{\Delta_i}{4}, T_{1,t-1} = s\right) < n \exp\left[-2\frac{\Delta_i^2}{16}m\right] = n^{-1}, \\ \mathbb{P}(\bar{E}_{i,t}, T_{i,t-1} > m) &\leq \sum_{s=m+1}^n \mathbb{P}\left(\hat{\mu}_{i,t-1} - \mu_i \geq \frac{\Delta_i}{4}, T_{i,t-1} = s\right) < n \exp\left[-2\frac{\Delta_i^2}{16}m\right] = n^{-1}. \end{aligned}$$

It follows that

$$\mathbb{P}(I_t = i, T_{i,t-1} > m, T_{1,t-1} > m) \leq \mathbb{P}(I_t = i, T_{i,t-1} > m, E_{1,t}, E_{i,t}) + 2n^{-1}.$$

Now note that $\hat{\mu}_{1,t-1} - \hat{\mu}_{i,t-1} \geq \Delta_i/2$ on events $E_{1,t}$ and $E_{i,t}$. Let

$$\hat{\mu}_{\max,t-1} = \max_{i \in [K]} \hat{\mu}_{i,t-1} \quad (12)$$

be the highest empirical mean in round t . Since $\hat{\mu}_{\max,t-1} \geq \hat{\mu}_{1,t-1}$, we have $\hat{\mu}_{\max,t-1} - \hat{\mu}_{i,t-1} \geq \Delta_i/2$. Therefore, on event $T_{i,t-1} > m$, we get

$$p_{i,t} \leq \exp[-2\gamma(\hat{\mu}_{\max,t-1} - \hat{\mu}_{i,t-1})^2 T_{i,t-1}] \leq \exp\left[-2\gamma\frac{\Delta_i^2}{4}m\right] \leq n^{-8\gamma}. \quad (13)$$

Finally, we chain all inequalities over all rounds and get that term 3 is bounded as

$$\sum_{t=K+1}^n \mathbb{P}(I_t = i, T_{i,t-1} > m, T_{1,t-1} > m) \leq n^{1-8\gamma} + 2. \quad (14)$$

C.5 Upper Bound on Term 2

Fix suboptimal arm i and round t . First, we apply Hoeffding’s inequality to arm i , as in Appendix C.4, and get

$$\begin{aligned} \mathbb{P}(I_t = i, T_{i,t-1} > m, T_{1,t-1} \leq m) &\leq \mathbb{P}(I_t = i, T_{i,t-1} > m, T_{1,t-1} \leq m, E_{i,t}) + n^{-1} \\ &= \mathbb{E}[p_{i,t} \mathbb{1}\{T_{i,t-1} > m, T_{1,t-1} \leq m, E_{i,t}\}] + n^{-1}. \end{aligned}$$

Let $\hat{\mu}_{\max,t-1}$ be defined as in (12). Now we bound $p_{i,t}$ from above using $p_{1,t}$. We consider two cases. First, suppose that $\hat{\mu}_{\max,t-1} > \mu_1 - \Delta_i/4$. Then we have (13). On the other hand, when $\hat{\mu}_{\max,t-1} \leq \mu_1 - \Delta_i/4$, we have

$$p_{i,t} = \frac{\exp[-2\gamma(\hat{\mu}_{\max,t-1} - \hat{\mu}_{i,t-1})^2 T_{i,t-1}]}{\exp[-2\gamma(\hat{\mu}_{\max,t-1} - \hat{\mu}_{1,t-1})^2 T_{1,t-1}]} p_{1,t} \leq \exp[2\gamma(\mu_1 - \hat{\mu}_{1,t-1})^2 T_{1,t-1}] p_{1,t}. \quad (15)$$

It follows that

$$p_{i,t} \leq \exp[2\gamma(\mu_1 - \hat{\mu}_{1,t-1})^2 T_{1,t-1}] p_{1,t} + n^{-8\gamma},$$

and we further get that

$$\begin{aligned} \mathbb{E}[p_{i,t} \mathbb{1}\{T_{i,t-1} > m, T_{1,t-1} \leq m, E_{i,t}\}] \\ \leq \mathbb{E}[\exp[2\gamma(\mu_1 - \hat{\mu}_{1,t-1})^2 T_{1,t-1}] p_{1,t} \mathbb{1}\{T_{1,t-1} \leq m\}] + n^{-8\gamma} \\ = \mathbb{E}[\exp[2\gamma(\mu_1 - \hat{\mu}_{1,t-1})^2 T_{1,t-1}] \mathbb{1}\{I_t = 1, T_{1,t-1} \leq m\}] + n^{-8\gamma}. \end{aligned}$$

With a slight abuse of notation, let $\hat{\mu}_{1,s}$ denote the average reward of arm 1 after s pulls. Then, since $T_{1,t} = T_{1,t-1} + 1$ on event $I_t = 1$, we have

$$\sum_{t=K+1}^n \mathbb{E} [\exp[2\gamma(\mu_1 - \hat{\mu}_{1,t-1})^2 T_{1,t-1}] \mathbb{1}\{I_t = 1, T_{1,t-1} \leq m\}] \leq \sum_{s=1}^m \mathbb{E} [\exp[2\gamma(\mu_1 - \hat{\mu}_{1,s})^2 s]] .$$

Now fix the number of pulls s and note that

$$\begin{aligned} \mathbb{E} [\exp[2\gamma(\mu_1 - \hat{\mu}_{1,s})^2 s]] &\leq \sum_{\ell=0}^{\infty} \mathbb{P} \left(\frac{\ell+1}{\sqrt{s}} > |\mu_1 - \hat{\mu}_{1,s}| \geq \frac{\ell}{\sqrt{s}} \right) \exp[2\gamma(\ell+1)^2] \\ &\leq \sum_{\ell=0}^{\infty} \mathbb{P} \left(|\mu_1 - \hat{\mu}_{1,s}| \geq \frac{\ell}{\sqrt{s}} \right) \exp[2\gamma(\ell+1)^2] \\ &\leq 2 \sum_{\ell=0}^{\infty} \exp[2\gamma(\ell+1)^2 - 2\ell^2] , \end{aligned}$$

where the last step is by Hoeffding's inequality. The above sum can be easily bounded for any $\gamma < 1$. In particular, for $\gamma = 1/8$, the bound is

$$\sum_{\ell=0}^{\infty} \exp \left[\frac{(\ell+1)^2}{4} - 2\ell^2 \right] \leq e^{\frac{1}{4}} + \sum_{\ell=1}^{\infty} 2^{-\ell} \leq e .$$

Now we combine all above inequalities and get that term 2 is bounded as

$$\sum_{t=K+1}^n \mathbb{P} (I_t = i, T_{i,t-1} > m, T_{1,t-1} \leq m) \leq 2em + n^{1-8\gamma} + 1 . \quad (16)$$

Finally, we chain (11), (14), and (16); and use that $m \leq 16\Delta_i^{-2} \log n + 1$. $\square$

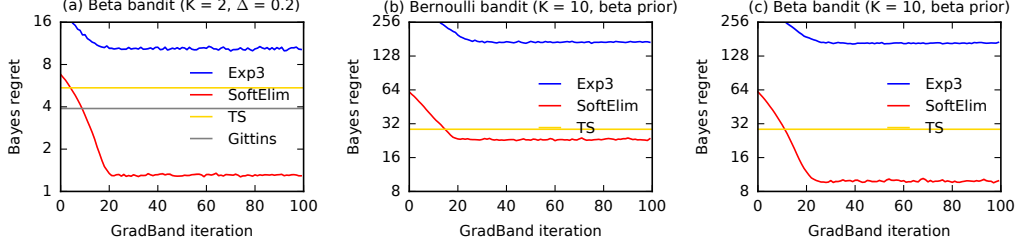


Figure 4: The Bayes regret of Exp3 and SoftElim policies, as a function of GradBand iterations. We report the average over 20 runs.

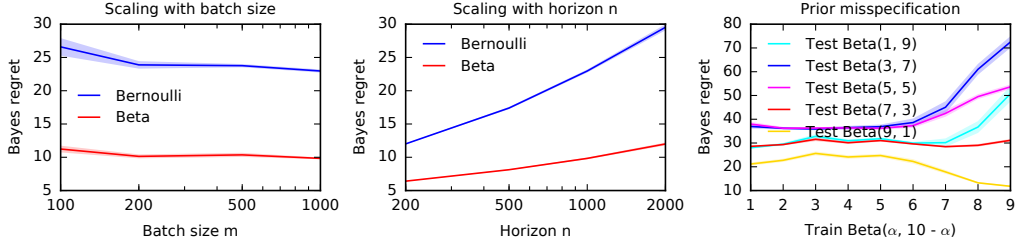


Figure 5: Robustness to batch size m , horizon n , and prior misspecification.

D Supplementary Experiments

We conduct additional experiments in this section.

D.1 More Complex Problems

Now we apply GradBand to three additional problems. The first problem is a beta bandit, where the rewards of arm i are drawn from $\text{Beta}(v\mu_i, v(1 - \mu_i))$ and $v = 4$ controls the variance of rewards. The remaining parameters of the problem are set as in the Bernoulli bandit in Section 6.2. The remaining two problems are variants of Bernoulli and beta bandits, where the number of arms is $K = 10$, their mean rewards are drawn as $\mu_i \sim \text{Beta}(1, 1)$, and the horizon is $n = 1000$. These problems are more challenging variants of our earlier problems, with 2 arms and a fixed gap.

The regret of our policies is reported in Figure 4. In all problems, the regret of SoftElim is lower than that of TS, which is a highly competitive baseline. The most significant improvements are in beta bandits, where SoftElim adapts to the lower variance of rewards. The poor performance of TS is due to the Bernoulli rounding in the algorithm (Section 6.2), which replaces low-variance beta rewards with high-variance Bernoulli rewards. As observed earlier, tuned Exp3 is not competitive.

D.2 Robustness to Model and Algorithm Parameters

This section presents three experiments, which show the robustness of SoftElim to the setting of its parameters and model misspecification. These experiments are conducted on the larger Bernoulli and beta problems in Appendix D.1.

In Figure 5a, we report the n -round regret of tuned SoftElim as a function of batch size m in GradBand. We observe that the mean regret, across all runs, is relatively stable as we decrease the batch size from 1000 to 100. The variance increases though. Setting $m = 100$ reduces the run time of GradBand ten fold, when compared to $m = 1000$ used in our earlier experiments.

In Figure 5b, we report the n -round regret of tuned SoftElim as we vary the horizon n , from 200 to 2000 rounds. We observe that the regret is roughly linear in $\log n$. This scaling is theoretically optimal. We expect it when the variance of gradients does not dominate GradBand, and thus GradBand can equally well optimize policies at shorter and longer horizons.

Figure	2c	4a	4b	4c	3a	3b
Gittins index	3.89 ± 0.07	3.89 ± 0.07	x	x	3.89 ± 0.07	2.26 ± 0.04
TS	5.47 ± 0.05	5.47 ± 0.05	28.06 ± 0.45	28.06 ± 0.45	5.47 ± 0.05	3.50 ± 0.03
UCB1	9.95 ± 0.03	9.95 ± 0.03	129.09 ± 0.60	129.09 ± 0.60	9.95 ± 0.03	8.52 ± 0.03
UCB-V	15.79 ± 0.03		289.82 ± 1.90	276.07 ± 1.65	15.79 ± 0.03	19.03 ± 0.10

Table 1: The Bayes regret of baseline bandit algorithms in Figures 2, 3 and 4. The crosses mark computationally-prohibitive experiments.

In Figure 5c, we investigate the robustness of tuned SoftElim to prior misspecification. In particular, we tune SoftElim on a Bernoulli bandit where $\mu_i \sim \text{Beta}(\alpha, 11 - \alpha)$ for $\alpha \in [10]$ and measure its n -round regret on a Bernoulli bandit with another α . We observe that the regret increases when we train and test on different problems. For instance, when we train and test on $\mu_i \sim \text{Beta}(1, 9)$, the regret is about 30. However, when we train on $\mu_i \sim \text{Beta}(9, 1)$, the regret increases to about 50. Nevertheless, we do not observe catastrophic failures, which would happen if the regret increased by an order of magnitude. We conclude that GradBand is relatively robust to prior misspecification.

E RNN Implementation

We carry out the RNN experiments using PyTorch framework. In this paper, we restrict ourselves to binary 0/1 rewards. For all experiments, our policy network is a single layer LSTM followed by LeakyRELU non-linearity and a fully connected layer. We use the fixed LSTM latent state dimension of 50, irrespective of numbers of arms. The implementation of the policy network is provided in the code snippet below:

```
1 class RecurrentPolicyNet(nn.Module):
2     def __init__(self, K=2, d=50):
3         super(RecurrentPolicyNet, self).__init__()
4         self.action_size = K # Number of arms
5         self.hidden_size = d
6         self.input_size = 2*d
7
8         self.arm_emb = nn.Embedding(K, self.hidden_size) # Number of arms
9         self.reward_emb = nn.Embedding(2, self.hidden_size) # For 0 reward
10        or 1 reward
11        self.rnn = nn.LSTMCell(input_size=self.input_size,
12                               hidden_size=self.hidden_size)
13        self.relu = nn.LeakyReLU()
14        self.linear = nn.Linear(self.hidden_size, self.action_size)
15
16        self.hprev = None
17
18    def reset(self):
19        self.hprev = None
20
21    def forward(self, action, reward):
22        arm = self.arm_emb(action)
23        rew = self.reward_emb(reward)
24
25        inp = torch.cat((arm, rew), 1)
26        h = self.rnn(inp, self.hprev)
27        self.hprev = h
28
29        h = self.relu(h[0])
30        y = self.linear(h)
31
32    return y
```

Listing 1: Policy Network

To train the policy we use the proposed GradBand algorithm as presented in Alg. 1. We used a batch-size $m = 500$ for all experiments. Along with theoretically motivated steps, we had to apply a few practical tricks:

- Instead of SGD, we used adaptive optimizers like Adam or Yogi [63].
- We used an exponential decaying learning rate schedule. We start with a learning rate of 0.001 and decay every step by a factor of 0.999.
- We used annealing over the probability to play an arm. This encourages exploration in early phase of training. In particular we used temperature $= 1/(1 - \exp(-5i/L))$, where i is current training iteration and L is the total number of training iterations.
- We applied curriculum learning as described in Section 6.4.

Our training procedure is highlighted in the code snippet below.

```
1 optimizer = torch.optim.Adam(policy.parameters(), lr=0.001)
2 scheduler = torch.optim.lr_scheduler.ExponentialLR(optimizer, 0.999)
3
4 ...
5
6 probs = rnn_policy_network(previous_action, previous_reward)
```

```

7 m = Categorical(probs/temperature)    # probability over K arms with
   temperature
8 action = m.sample()                  # select one arm
9 reward = bandit.play(action)         # receive reward
10
11 ...
12
13 loss = -m.log_prob(action) * (cumulative_reward - baseline) # Eq (3)
14 loss.backward()                     # Eq (9)
15 optimizer.step()
16 scheduler.step()
17
18 ...

```

Listing 2: Training overview