Policy Gradient Optimization of Thompson Sampling Policies*

Seungki Min
Industrial and Systems Engineering
KAIST
skmin@kaist.ac.kr

Ciamac C. Moallemi
Graduate School of Business
Columbia University
ciamac@gsb.columbia.edu

Daniel J. Russo
Graduate School of Business
Columbia University
djr2174@gsb.columbia.edu

Current Revision: August 2022

Abstract

We study the use of policy gradient algorithms to optimize over a class of generalized Thompson sampling policies. Our central insight is to view the posterior parameter sampled by Thompson sampling as a kind of pseudo-action. Policy gradient methods can then be tractably applied to search over a class of sampling policies, which determine a probability distribution over pseudo-actions (i.e., sampled parameters) as a function of observed data. We also propose and compare policy gradient estimators that are specialized to Bayesian bandit problems. Numerical experiments demonstrate that direct policy search on top of Thompson sampling automatically corrects for some of the algorithm's known shortcomings and offers meaningful improvements even in long horizon problems where standard Thompson sampling is extremely effective.

1. Introduction

In both academia and industry, Thompson sampling has emerged as a leading approach to exploration in online decision making. This is driven by the algorithm's simplicity, generality, ability to leverage rich prior information about problem, and its resilience to delayed feedback. But, like most popular bandit algorithms, it is a heuristic design based on intuitive appeal and some degree of mathematical insight. The tutorial paper by Russo et al. (2018) details numerous settings in which Thompson sampling can be grossly suboptimal. We highlight several such situations:

- Settings where the time horizon is short relative to the number of arms. As an extreme case, in the situation there is a single period remaining, the myopic policy is optimal and Thompson sampling will over explore. At another extreme, if there are many arms, it may be optimal to

*The second and third authors wish to thank Paolo Baudissone for early exploration of some the ideas in this paper.

only restrict exploration to a subset so that a good arm can be identified in time to exploit over a reasonable time frame.

- Thompson sampling does not directly consider reward noise. If there is a significant heterogeneity in the noise across arms, Thompson sampling may suboptimally pull noisy arms about which there is little hope to learn.
- In settings with correlated arms, pulling a single arm may provide information about many other arms. In these settings, there may be "free exploration" where, for example, a myopic policy might learn about all arms and the type of explicit exploration undertaken by Thompson sampling may be wasteful.

An underlying theme in the above example is the fact that Thompson sampling does not make an explicit exploration-exploitation trade off. Even in less extreme settings, Thompson sampling is generally thought to explore too aggressively.

Thompson sampling is designed for a Bayesian multi-armed bandit problem, a well-defined optimization problem that has long been approached using the tools of dynamic programming. Despite this, the literature offers no way to use computation, rather than human ingenuity, to improve on standard Thompson sampling. We propose and benchmark the use of policy gradient methods to optimize over a given family of Thompson sampling style algorithms. The proposed methods use substantial offline computation, but the resulting policies can be executed without additional real-time computation.

At first glance, it appears standard policy gradient algorithms (Williams, 1992) cannot be efficiently applied to Thompson sampling. The challenge is that traditional policy gradient methods require computation of the score function of the distribution of actions. While, in principle, Thompson sampling randomly draws an action at each decision point, the distribution over actions from which it samples is not available in closed form. Instead, efficient implementations sample a model parameter from a posterior distribution and then select the action that is optimal under this sampled parameter. Under such an implementation, it may be difficult to compute the probability of selecting each action.

Our central insight is as follows: we view the posterior parameter sampled by Thompson sampling as a kind of "pseudo-action". In our framework, a *sampling policy* maps the history of observations to a probability distribution over the parameter space. A full algorithm will, in each period, draw a sample according to the sampling policy and subsequently apply the base action that is optimal under the sampled parameter. In standard Thompson sampling, the sampling policy applies Bayes' rule, mapping any history to the associated posterior distribution over pseudo-actions (parameters). Mathematically, this is equivalent to the standard formulation which views the decision as a choice of base action. Critically, however, by viewing the decision as a choice of pseudo-action (parameter), the distribution of pseudo-actions for Thompson sampling is often available in closed-form: it is simply the posterior distribution of the parameter. This simple but powerful shift

in perspective enables us to search over Thompson sampling style policies using policy gradient methods. Indeed, we will use policy gradient to search over a class of sampling policies that are themselves parameterized by hyper-parameters we call *meta-parameters*.

A sampling policy could be parameterized in many ways. One option is to parameterize them by complex neural networks. Our experiments demonstrate that even simple modifications of standard Thompson sampling offer substantial benefit. One approach builds on Thompson sampling by viewing the statistical parameters of the Bayesian model (e.g., the prior distribution, the noise distribution, etc.) as meta-parameters. Another takes the posterior distribution used by standard Thompson sampling and reshapes it. Policy gradient methods for searching over the meta-parameters are tractable as long as (i) the sampling policies can be applied efficiently and (ii) given any history, one can efficiently calculate derivatives of the sampling distribution's log density with the respect to the meta-parameters. Typically (ii) requires that probability density function is known up to a proportionality constant.

Our work has a close conceptual connection to work on meta-learning (see e.g., Finn et al., 2017; Baxter, 2000). As is nicely articulated by Bastani et al. (2021), many companies face a large sequence of experimentation tasks, raising the question of how to effectively share information across these tasks. Consider a web company who may run thousands of A/B tests per year, giving them strong prior knowledge of the distribution of effect sizes and click through rates. Or a news article recommendation service has a new set of articles each day and needs to experiment to learn which will be popular. Each day can be viewed as its own instance of a bandit problem and the platform's goal is to do well on average across a large number of days. Bastani et al. (2021) suggest an empirical Bayesian approach, where the prior of Thompson sampling is statistically estimated from data on previous tasks. This view of meta-learning as learning a prior distribution has long been recognized. Our approach, however, will not be to apply Thompson sampling directly using some form of statistically learned prior, since Thompson sampling is not itself an optimal policy. If historical data can be used to build a simulator of this meta-bandit problem, then it is more appropriate to search over Thompson sampling like policies aiming to directly optimize the true performance metric, the average reward. This idea — shifting from learning elements of the statistical model such as the prior distribution or noise model via statistical estimation to direct optimization — may especially be powerful in settings where the statistical model is mis-specified.

Our contributions are as follows:

1. **We develop a tractable framework for policy gradient estimation for sampling policies.**

   Several recent works have explored the use of gradient based search to tune bandit algorithms (Duan et al., 2016; Boutilier et al., 2020). Relative to these works, one of our main contributions is to uncover a way to apply policy gradient methods to Thompson sampling, allowing us to fine-tune a widely used algorithm with strong theoretical guarantees. Recently, an independent and contemporaneous pre-print by Kveton et al. (2020) discovered a similar

approach to tuning Thompson sampling.

2. **We provide and analyze multiple gradient estimators for sampling policies.**

   As in the broader application of policy gradient for reinforcement learning, there are multiple possible gradient estimators possible, through different choices of reward metrics and baselines. We derive several novel policy gradient estimators that are specifically tailored to Bayesian bandit problems. We are able to compare their variance theoretically and empirically.

3. **We computationally demonstrate the benefits of our approach.**

   Through simple numerical experiments, we provide a compelling proof of concept. Policy search produces policies that correct for shortcomings of Thompson sampling in short horizon problems or problems with large discrepancies between the variances of arm rewards. Perhaps more surprisingly, policy search offers substantial improvements over Thompson sampling even in a canonical long horizon problem to which it is ideally suited. We also compare against optimistic Gittins indices (Gutin and Farias, 2016), information directed sampling (Russo and Van Roy, 2018), and Bayesian upper confidence bound algorithms (Kaufmann et al., 2012), confirming that direct policy search on top of Thompson sampling produces state of the art results for widely studied problem settings. In the future, we hope to extend the numerical experiments beyond problems with independent arms.

Future potential and current limitations. In an influential 2019 essay titled "The Bitter Lesson", pioneering artificial intelligence researcher Rich Sutton writes:

> The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available.

The design of multi-armed bandit algorithms has relied almost solely on researcher knowledge and ingenuity. We preserve design principles and theoretical guarantees from the literature by initializing a gradient based policy search algorithm at the Thompson sampling algorithm. But our method leverages computation rather than researcher ingenuity to uncover policy improvements.

Simple and targeted experiments show that gradient based policy search reliably overcomes known shortcomings of a leading bandit algorithm. But the full benefits may only become apparent once more substantial computational resources have been deployed. Lacking the data, compute infrastructure, and engineering talents available at some large companies, we leave this direction

open.

2. Model

We consider a multi-armed bandit problem in a Bayesian setting.

Rewards. Let $\mathcal{A}$ be the set of arms, possibly infinite, among which the decision maker (DM) can select at each time $t = 1, \ldots, T$. When the DM pulls an arm $a \in \mathcal{A}$ at time t, they earn a random reward $R_{a,t}$ drawn from a distribution $\mathcal{R}$ that is parameterized by *model parameters* $\theta \in \Theta$, i.e.,

$$R_{a,t}|\theta \sim \mathcal{R}(\theta, a). \tag{1}$$

We assume that the rewards $(R_{a,t})_{t\in[T]}$ are conditionally independent and identically distributed[1] given θ, for each $a \in \mathcal{A}$. We further define $\mu\colon \Theta \times \mathcal{A} \to \mathbb{R}$ to be the *mean reward function*, i.e.,

$$\mu(\theta, a) \triangleq \mathbb{E}_{R\sim\mathcal{R}(\theta,a)}\left[R\right]. \tag{2}$$

As we consider a Bayesian setting, we view θ as a (multivariate) random variable that is sampled from a *prior distribution*, which we denote by $\mathcal{P}(y_0)$, i.e.,

$$\theta \sim \mathcal{P}(y_0), \tag{3}$$

where $y_0 \in \mathcal{Y}$ are the sufficient statistics that parameterize the prior distribution.

Information set. The parameters θ are unknown to the DM, but can be inferred by observing the reward realizations sequentially revealed over time. More precisely, let H_t be the history, or information revealed up to time t (inclusive):

$$H_t \triangleq \left(A_s, R_{A_s,s}\right)_{s\in[t]}. \tag{4}$$

This includes the actions taken by the DM and the rewards realized through time t. We assume that the DM has knowledge of both the prior distribution $\mathcal{P}(y_0)$ and the functional form of reward distribution $\mathcal{R}$.

As a Bayesian learner, the DM will update their belief according to Bayes' rule whenever observing a new reward realization, and thus maintain a *posterior distribution* for θ at each time.

[1] One important case where the rewards are *not* i.i.d. is the case of contextual bandits. Here, the reward at time t is given by $R_{a,t}|\theta, x_t \sim \mathcal{R}(\theta, a, x_t)$. Here x_t is the context at time t, and this is a stochastic process that evolves independently of the DM's actions. The framework we develop here can be extended to accommodate contextual cases as well in a straightforward fashion, but in the interest of simplifying the presentation we will not be explicit about this.

Without loss of generality,[2] we assume that the posterior is represented as

$$\theta|H_t \;\sim\; \mathcal{P}(Y_t). \tag{5}$$

Here, $Y_t \in \mathcal{Y}$ is a random variable that denotes the sufficient statistics of the posterior distribution after observing the history H_t (i.e., after observing the t^{th} reward realization). We set $Y_0 = y_0$.

We will describe the randomness of our stochastic model more explicitly as follows. The DM's policy π is described by a sequence of deterministic mappings $(\pi_t)_{t\in[T]}$. Each mapping $\pi_t \colon \mathcal{H}_{t-1} \times \mathcal{E} \to \mathcal{A}$ specifies the next action A_t as a function of history $H_{t-1} \in \mathcal{H}_{t-1}$ that is revealed immediately prior to time t, and random noise $\epsilon_t \in \mathcal{E}$ that can be utilized for randomization in the choice of action. Similarly, the reward realization is described by a mapping $r \colon \Theta \times \mathcal{A} \times \Xi \to \mathbb{R}$ that specifies the next reward realization, where any randomness is generated by the noise variable $\xi_t \in \Xi$. That is,

$$A_t = \pi_t(H_{t-1}, \epsilon_t), \quad R_{a,t} = r(\theta, a, \xi_t). \tag{6}$$

We assume, without loss of generality, that the noise random variables $(\epsilon_t)_{t\in[T]}$ are independent and identically distributed, as are $(\xi_t)_{t\in[T]}$.

We define an *instance* or *episode*, denoted by I, as a random variable that encodes all uncertainties in the environment, but not the randomness in the DM's decision rule:

$$I \triangleq \left(\theta, (\xi_t)_{t\in[T]}\right). \tag{7}$$

In other words, given an instance I, we exactly know what rewards will be realized for any given action sequence $a_{1:T} \in \mathcal{A}^T$ committed by the DM. The set of all possible instances is denoted by $\mathcal{I}$.

Objective. The DM aims to earn as much reward as possible in expectation. Given the DM's decision rule π, its *expected total reward*, denoted by $\textsc{Reward}[\pi]$, is defined as

$$\textsc{Reward}[\pi] \triangleq \mathbb{E}\left[\sum_{t=1}^{T} R_{A_t,t}\right], \tag{8}$$

where the expectation is taken with respect to the randomness of instance (i.e., the randomness of the parameters θ and the reward realizations) and also any randomness of the choice of actions (if π is a randomized policy).

To better illustrate our setup, we provide an example of a canonical multi-armed bandit problem described with the notation introduced above.

Example 1 (Gaussian MAB). *Consider a finite number of arms $\mathcal{A} \triangleq \{1, \ldots, K\}$. Each arm a yields normally distributed rewards with an unknown mean θ_a and a known variance σ_a^2, where the prior*

[2] When $\mathcal{P}$ is a conjugate prior of $\mathcal{R}$ and belongs to the exponential family, the sufficient statistics Y_t will admit a compact representation. In other cases, Y_t may represent the entire history, i.e., $Y_t = H_t$.

of θ_a is given by a normal distribution $\mathcal{N}(m_{a,0}, v_{a,0}^2)$. The model parameters are given by the vector $\theta \triangleq (\theta_a)_{a\in[K]}$.

Each instance takes the form $I \triangleq \Big(\theta, (\xi_t)_{t\in[T]}\Big)$, where $(\xi_t)_{t\in[T]}$ are i.i.d. standard normal random variables that randomize the reward realizations according to

$$\mu(\theta, a) = \theta_a, \quad r(\theta, a, \xi_t) = \theta_a + \sigma_a \xi_t. \tag{9}$$

The posterior for a parameter θ_a after time t is given by a normal distribution $\mathcal{N}(m_{a,t}, v_{a,t}^2)$. Here, the sufficient statistics $m_{a,t}$ and $v_{a,t}^2$ can be computed in a closed-form according to

$$m_{a,t} = v_{a,t}^{-2} \times \left(v_{a,0}^{-2} \cdot m_{a,0} + \sigma_a^{-2} \sum_{s=1}^{t} \mathbb{I}_{\{A_s=a\}} R_{A_s,s} \right), \quad v_{a,t}^2 = \left(v_{a,0}^{-2} + \sigma_a^{-2} \sum_{s=1}^{t} \mathbb{I}_{\{A_s=a\}} \right)^{-1}.$$

As a collection of these sufficient statistics across the arms, $Y_t \triangleq (m_{a,t}, v_{a,t}^2)_{a\in[K]} \in \mathbb{R}^{2K}$, determine the posterior of the parameters θ given the history H_t.

3. Parameterized Thompson Sampling

Thompson sampling (TS) (Thompson, 1933) is a randomized policy that works as follows. At each time $t = 1, \ldots, T$: (i) the parameters $\tilde{\theta}_t$ are sampled from the posterior distribution $\mathcal{P}(Y_{t-1})$ given all information prior to time t; and (ii) an action is chosen to maximize the expected reward given these sampled parameters $\tilde{\theta}_t$. In other words,

$$\tilde{\theta}_t \sim \mathcal{P}(Y_{t-1}), \quad A_t^{\mathrm{TS}} \leftarrow \operatorname*{argmax}_{a\in\mathcal{A}} \mu(\tilde{\theta}_t, a), \tag{10}$$

where $Y_{t-1} \in \mathcal{Y}$ are, as introduced in (5), the sufficient statistics describing the posterior distribution of true parameters θ given the history H_{t-1}. As time progresses, the above procedure is repeated, while updating the posterior distribution according to Bayes' rule.

An important characteristic of TS is "probability matching". Under TS, the probability that an arm a is selected at time t is equal to the probability that the arm a is indeed the best one that Bayesian inference predicts, i.e.,

$$\mathbb{P}(A_t^{\mathrm{TS}} = a \,|\, H_{t-1}) = \mathbb{P}\left(a = \operatorname*{argmax}_{a'\in\mathcal{A}} \mu(\theta, a') \,|\, H_{t-1} \right), \quad \forall a \in \mathcal{A}. \tag{11}$$

However, probability in (11) is difficult to evaluate since it does not admit a closed-form expression in most cases and does not admit feasible policy gradient estimators.

We consider a class of variants of TS where the sampling policy in (10) is not the posterior distribution, but instead is some other distribution parameterized by *meta-parameters* $\lambda \in \Lambda \subseteq \mathbb{R}^d$. In other words, given λ, the corresponding *sampling policy* $\mathrm{TS}(\lambda)$ repeats the following at each

time t:

$$\tilde{\theta}_t \sim \mathcal{P}_\lambda(H_{t-1}), \quad A_t^{\mathrm{TS}(\lambda)} \leftarrow \operatorname*{argmax}_{a \in \mathcal{A}} \mu(\tilde{\theta}_t, a), \tag{12}$$

where $\mathcal{P}_\lambda(H_{t-1})$ is a distribution on the parameter space Θ that has arbitrary dependency on the meta-parameters λ and the history H_{t-1}. The sampling policy $\mathrm{TS}(\lambda)$ is almost identical to the naïve TS except that it samples the parameters from $\mathcal{P}_\lambda(H_{t-1})$ instead of $\mathcal{P}(Y_{t-1})$. In this way, sampling policies can be viewed as a natural generalization of TS, emitting at each time a randomized pseudo-action $\tilde{\theta}_t$ (choice of parameters) from which a base action is determined, rather than directly emitting a base action.

As we would like to employ policy gradient methods to optimize over the meta-parameters λ, we will assume that the probability density (or mass) function of the distribution $\mathcal{P}_\lambda(H_{t-1})$ is differentiable with respect to λ over its domain Λ, for any realization of history H_{t-1} almost surely. Aside from this, the distribution $\mathcal{P}_\lambda(H_{t-1})$ defining the sampling policy is allowed to be essentially arbitrary. However, in order to illustrate our ideas, consider the following example:

Example 2 (Posterior reshaping.)**.** *We adopt and generalize the idea proposed in Chapelle and Li (2011), and let the algorithm to sample the parameters from the a reshaped posterior distribution,*

$$\mathcal{P}_\lambda(H_{t-1}) = \mathcal{P}(\phi_\lambda(Y_{t-1})), \tag{13}$$

where $\mathcal{P}(\cdot)$ is the posterior distribution defined in (5)*, and $\phi_\lambda \colon \mathcal{Y} \to \mathcal{Y}$ is a differentiable mapping, parameterized by λ, that transforms one set of sufficient statistics to another.*

Posterior reshaping is motivated by several arguments. Compared to the general parameterization (12), the posterior reshaping does not parameterize "how to learn". Instead, it parameterizes how to utilize the learned results, while maintaining the Bayesian learning logic as it is. This can significantly reduce the effort required for tuning the meta-parameters. Moreover, its implementation requires a minimal effort once one has already implemented the standard TS. Indeed, when $\phi_\lambda(\cdot)$ is the identity map, posterior reshaping reduces to standard TS. Hence, under appropriate technical assumptions, a local policy gradient search starting at the identity map will be guaranteed to do no worse than standard TS.

As discussed in the introduction, the standard TS suffers from the over-exploration, for example when the time horizon is short relative to the number of arms. Posterior reshaping can naturally address this, by reducing uncertainty in the sampling distribution. As an extreme example, consider a situation where we are given a single time period (i.e., $T = 1$). The optimal policy is myopic — the optimal action is to pick the arm with the largest prior mean — which can be implemented by reshaping the posterior distribution to concentrate on the prior mean. Such posterior concentration also appears in the work of Min et al. (2019). The IRS.FH policy they suggest is posterior reshaping sampling policy.

Furthermore, it is possible that TS with the correct model parameters (e.g., the specification of

the prior distribution or the reward distribution) may not be optimal for the performance of the algorithm. Within the framework of posterior reshaping, we can find the set of model parameters that are empirically tuned for performance so as to outperform to the one with the correct values.[3]

A concrete examples of posterior reshaping in the Gaussian case can be developed as follows:

Example 3 (Posterior reshaping for Gaussian MAB). *Using the notation of Example 1, standard TS in Gaussian MAB samples the parameters $\tilde{\theta}_t = (\tilde{\theta}_{a,t})_{a\in\mathcal{A}}$ according to*

$$\tilde{\theta}_{a,t} \sim \mathcal{N}\left(\frac{m_{a,0} + \frac{v_{a,0}^2}{\sigma_a^2}\cdot S_{a,t-1}}{1+\frac{v_{a,0}^2}{\sigma_a^2}\cdot N_{a,t-1}},\ \frac{v_{a,0}^2}{1+\frac{v_{a,0}^2}{\sigma_a^2}\cdot N_{a,t-1}}\right), \tag{14}$$

for each $a = 1,\dots,K$, where the sufficient statistics are given by $S_{a,t-1} \triangleq \sum_{s=1}^{t-1}\mathbb{I}_{\{A_s=a\}}R_{A_s,s}$, and $N_{a,t-1} \triangleq \sum_{s=1}^{t-1}\mathbb{I}_{\{A_s=a\}}$.

We consider posterior reshaping with meta-parameters $\lambda \triangleq (\lambda_a^m, \lambda_a^v, \lambda_a^\sigma, \lambda_a^\gamma)_{a\in\mathcal{A}} \in \mathbb{R}^{4K}$ under which $\tilde{\theta}_{a,t}$ is sampled from

$$\tilde{\theta}_{a,t} \sim \mathcal{N}\left(\frac{\lambda_a^m + \lambda_a^\sigma\cdot S_{a,t-1}}{1+\lambda_a^\sigma\cdot N_{a,t-1}},\ \frac{\lambda_a^v(1-t/T)^{\lambda_a^\gamma}}{1+\lambda_a^\sigma\cdot N_{a,t-1}}\right). \tag{15}$$

This policy reduces to standard TS if we take $\lambda_a^m = m_{a,0}$ (prior mean), $\lambda_a^v = v_{a,0}^2$ (prior variance), $\lambda_a^\sigma = v_{a,0}^2/\sigma_a^2$ (precision ratio), and $\lambda_a^\gamma = 0$ (variance decay exponent). The amount of exploration is controlled by λ_a^v and λ_a^γ. In particular, the term $(1-t/T)^{\lambda_a^\gamma}$ diminishes exploration near the end of the horizon, where the benefit from exploration is limited.

Note that this parameterization scheme can be represented in the form of (13)*, since the sufficient statistics of the realized observations (i.e., $S_{a,t-1}$ and $N_{a,t-1}$) are uniquely determined from those of current posterior distribution (i.e., $m_{a,t-1}$ and $v_{a,t-1}^2$) and prior distribution (i.e., $m_{a,0}$ and $v_{a,0}^2$). Also note that the probability density function is differentiable with respect to λ given that $\lambda_a^v > 0$ and $\lambda_a^\sigma > 0$.*

A more complex example is as follows:

Example 4 (Deep recurrent neural network parameterization). *One might consider a recurrent neural network (RNN) structure with, at each time t, input $(A_t, R_{A_t,t})$, hidden state $\tilde{Y}_t$ and output being the sampled pseudo-action $\tilde{\theta}_t$. The network would evolve according to*

$$\tilde{Y}_t \leftarrow \phi_{\lambda_Y}^Y(\tilde{Y}_{t-1}, A_t, R_{A_t,t}), \quad \tilde{\theta}_t \sim \mathcal{P}\left(\phi_{\lambda_\theta}^\theta(\tilde{Y}_{t-1})\right).$$

Here, the hidden state $\tilde{Y}_t$ is analogous to a sufficient statistic in that it summarizes the history

[3]Even if the model is mis-specified, the policy gradient method can be applied, but we need to be careful in the choice of gradient estimator in order to avoid a bias in the gradient estimation. See the related discussion in Section 4.3.

up to an including time t. Two deep neural networks, $\phi^Y_{\lambda_Y}(\cdot)$ and $\phi^\theta_{\lambda_\theta}(\cdot)$, with weights λ_Y and λ_θ, govern the evolution of the hidden state $\tilde{Y}_t$ and the output $\tilde{\theta}_t$, respectively. The meta-parameters $\lambda \triangleq (\lambda_Y, \lambda_\theta)$ would be optimized with policy gradient methods.

Example 4 is in the spirit of the approach of Duan et al. (2016) and Boutilier et al. (2020), where RNNs were fit with policy search methods, but where the policies output actions. Here, to contrast, the RNN outputs distributional parameters which are then sampled, leveraging on top of the structure of Thompson sampling.

4. Policy Gradient for Thompson Sampling

We aim to search over the meta-parameters $\lambda \in \mathbb{R}^d$ so that the corresponding policy $\mathrm{TS}(\lambda)$ improves over the original TS significantly. For this purpose, we adopt the policy gradient framework, which applies variants of stochastic gradient ascent to optimize total expected reward. Formally, one can have in mind the iteration,

$$\lambda_{k+1} = \lambda_k + \alpha_k G(\lambda_k, w_k) \tag{16}$$

where (α_k) is a step-size sequence, (w_k) is a sequence of i.i.d. random variables, and $G(\lambda_k, w_k)$ is an unbiased gradient, i.e.,

$$\mathbb{E}\left[G(\lambda, w_k)\right] = \nabla_\lambda \textsc{Reward}\big[\mathrm{TS}(\lambda)\big] = \nabla_\lambda \mathbb{E}\left[\sum_{t=1}^{T} R_{A_t^{\mathrm{TS}(\lambda)},t}\right].$$

Typically, the w_k denote the randomness used by a stochastic simulator. For the gradient estimators we use, $w_k \triangleq \big(\epsilon_t^{(k)}, \xi_t^{(k)}\big)_{t\in[T]}$ consists of realizations of the random noise terms that determine the reward realizations and the action selection of a randomized algorithm. In deriving and comparing gradient estimators, we omit the dependence on k. It is worth noting that the iteration (16) is meant for illustrative purposes, and other first order stochastic methods, e.g., Adam (Kingma and Ba, 2014), can also be utilized.

4.1. Score Function Gradient Estimation

Most implementations of policy gradient use score function gradient estimation (Williams, 1992). However, the conventional scheme requires computing $\nabla_\lambda \log \mathbb{P}\big(A_t^{\mathrm{TS}(\lambda)} = a\big)$, which is typically intractable since there is no closed-form expression for the distribution of the chosen action.

We circumvent this issue by interpreting the sampled parameters $\tilde{\theta}_t$ as an (pseudo-)action taken by the policy at time t. One can imagine an equivalent bandit environment whose action space is set to the parameter space Θ and the decision maker earns the reward $\tilde{R}_{\tilde{\theta}_t,t} \triangleq R_{A_t,t}$ associated with the arm $A_t \triangleq \mathrm{argmax}_a\, \mu(\tilde{\theta}_t, a)$ as a result of his decision $\tilde{\theta}_t$. Assume that the sampling distribution $\mathcal{P}_\lambda(H_{t-1})$ under the any H_{t-1} has a probability density function $p_\lambda(\,\cdot\,; H_{t-1})$. Assume as well that

$p_\lambda(\,\cdot\,; H_{t-1})$ is differentiable as a function of λ. This leads to the gradient estimator

$$G \triangleq \sum_{t=1}^{T} S_t \sum_{s=t}^{T} R_{A_s^{\mathrm{TS}(\lambda)},s} \tag{17}$$

where

$$S_t \triangleq \nabla_\lambda \log p_\lambda \left(\tilde{\theta}_t \,; H_{t-1}\right) \tag{18}$$

denotes the score functions. This form of score function gradient estimator is well known to be unbiased, i.e., $\mathbb{E}[G] = \nabla_\lambda \textsc{Reward}\big[\mathrm{TS}(\lambda)\big]$, which is referred to as the policy gradient theorem in the reinforcement learning literature. Formally, unbiasedness requires technical conditions that allow for the interchange of integrals and derivatives. We refer to L'Ecuyer (1995) for appropriate conditions.

4.2. Admissible Gradient Estimators

The standard gradient estimator (17) can be very noisy due to the high variability of reward realizations and random action selections. In this section, we propose a broader, more general class of gradient estimators, and demonstrate that they remain unbiased as long as they satisfy a certain admissibility requirement. Later, we will suggest a specific list of estimators for bandit problems, and provide a theoretical comparison among them in terms of variance reduction.

General representation. With processes $M \triangleq (M_t)_{t\in[T]}$, which we call a *reward metric*, and $B \triangleq (B_t)_{t\in[T]}$, which we call a *baseline*, we define the gradient estimator $G^{M,B}$ as follows:

$$G^{M,B} \triangleq \sum_{t=1}^{T} S_t \cdot (M_t - B_t)\,. \tag{19}$$

The reward metric M_t is a random variable that accounts for the sum of rewards that the policy earns on the remaining horizon $t, t+1, \ldots, T$, and B_t is another random variable that represents some benchmark for the rewards over the same period. Note that the estimator (17) can be obtained by taking $M_t = \sum_{s=t}^{T} R_{A_s,s}$ and $B_t = 0$.

Admissibility. We state a condition on the reward metric M and the baseline B under which they induce an unbiased gradient estimator $G^{M,B}$.

Definition 1 (Admissible reward metric and baseline). *A reward metric M is admissible if for all $t \in [T]$ it is integrable and*

$$\mathbb{E}\left[M_t \,\middle|\, H_{t-1}, \tilde{\theta}_t\right] = \mathbb{E}\left[\sum_{s=t}^{T} R_{A_s,s} \,\middle|\, H_{t-1}, \tilde{\theta}_t\right]. \tag{20}$$

A baseline B is admissible if B_t is integrable and B_t and $\tilde{\theta}_t$ are conditionally independent given

H_{t-1} *for all* $t \in [T]$*, i.e.,*

$$B_t \perp\!\!\!\perp \tilde{\theta}_t \,\Big|\, H_{t-1}. \tag{21}$$

The first condition (20) ensures that a risk-neutral decision maker would not differentiate between M_t and the sum of future rewards when deciding the next action. These two measures have the same expectation given any history (which corresponds to the state in dynamic programming terms) and any pseudo-action $\tilde{\theta}_t$. The second condition (21) ensures that the decision-maker does not need to take baseline into consideration when making a decision, since the baseline is independent of the next pseudo-action $\tilde{\theta}_t$.

The above interpretation implies that the substitution of reward metric (from $\sum_{s=t}^{T} R_{A_s,s}$ to M_t) and the presence of baseline B_t do not affect the DM's decision at each time t, as long as they satisfy the admissibility conditions (20)–(21). Therefore, we can infer that the generalized gradient estimator (19) is equal in expectation to the standard one (17), which is proved formally in the following theorem.

Theorem 1 (Unbiasedness of gradient estimator)**.** *If the reward metric* M *and the baseline* B *are admissible, then* $\mathbb{E}[G^{M,B}] = \mathbb{E}[G]$.

Proof. Note that S_t is measurable with respect to $\sigma(H_{t-1}, \tilde{\theta}_t)$ and

$$\mathbb{E}\left[\, S_t \middle|\, H_{t-1}\right] = 0, \tag{22}$$

due to the property of the score function. By the condition (20), we obtain

$$\begin{aligned}
\mathbb{E}\left[S_t M_t\right] &= \mathbb{E}\left[\mathbb{E}\left(S_t M_t \,\middle|\, H_{t-1}, \tilde{\theta}_t\right)\right] \\
&= \mathbb{E}\left[S_t \times \mathbb{E}\left(M_t \,\middle|\, H_{t-1}, \tilde{\theta}_t\right)\right] \\
&= \mathbb{E}\left[S_t \times \mathbb{E}\left(\sum_{s=t}^{T} R_{A_s,s} \,\middle|\, H_{t-1}, \tilde{\theta}_t\right)\right] \\
&= \mathbb{E}\left[\mathbb{E}\left(S_t \times \sum_{s=t}^{T} R_{A_s,s} \,\middle|\, H_{t-1}, \tilde{\theta}_t\right)\right] \\
&= \mathbb{E}\left[S_t \times \sum_{s=t}^{T} R_{A_s,s}\right].
\end{aligned}$$

By the condition (21), we further obtain

$$\mathbb{E}\left[S_t B_t\right] = \mathbb{E}\left[\mathbb{E}\left(S_t B_t | H_{t-1}\right)\right] = \mathbb{E}\left[\mathbb{E}\left(S_t | H_{t-1}\right) \times \mathbb{E}\left(B_t | H_{t-1}\right)\right] = 0.$$

Combining these results, we deduce that

$$\mathbb{E}\left[G^{M,B}\right] = \mathbb{E}\left[\sum_{t=1}^{T} S_t \times (M_t - B_t)\right] = \mathbb{E}\left[\sum_{t=1}^{T} S_t \sum_{s=t}^{T} R_{A_s,s}\right] = \mathbb{E}[G],$$

which concludes the proof. ■

One particularly interesting class of reward metrics and baselines are those which are time separable:

Example 5 (Time separable reward metric and baseline)**.** *Consider*

$$M_t \triangleq \sum_{s=t}^{T} \hat{r}_s(A_{1:s}, I), \quad B_t \triangleq b_t(A_{1:t-1}, I),$$

where $\hat{r}_t \colon \mathcal{A}^t \times \mathcal{I} \to \mathbb{R}$ *and* $b_t \colon \mathcal{A}^{t-1} \times \mathcal{I} \to \mathbb{R}$ *are the deterministic functions that satisfy*

$$\mathbb{E}\left[\hat{r}_t(a_{1:t}, I) \,|\, H_{t-1}, A_{1:t-1} = a_{1:t-1}\right] = \mathbb{E}\left[r(\theta, a_t, \xi_t) \,|\, H_{t-1}, A_{1:t-1} = a_{1:t-1}\right], \quad \forall\ a_{1:t} \in \mathcal{A}^t,\ t \in [T].$$

Then, the reward metric $M \triangleq (M_t)_{t\in[T]}$ *and the baseline* $B \triangleq (B_t)_{t\in[T]}$ *are admissible.*

We remark that the baseline is allowed to be *instance-dependent*, meaning that it can depend on instance I defined in (7) that determines the realizations of rewards and the true parameters θ. This is a considerable generalization of the literature in which baselines are typically chosen as a deterministic function of state (Sutton and Barto, 2018). The use of common randomness (Glasserman and Yao, 1992) in the the baseline and the reward metric can reduce variance, especially when most variation in observed algorithm performance is driven by different realizations of the problem instance rather than differences in the choice of meta parameter.

4.3. Reward Metrics and Baselines

We suggest a specific series of reward metrics and baselines that are admissible for Bayesian bandit problems. A number of these take the time separable form of Example 5.

Reward metrics. The followings are possible choices for reward metric:

1. The observed reward $M_t^{\text{obs}} \triangleq \sum_{s=t}^{T} R_{A_s,s}$.
2. The mean reward $M_t^{\text{mean}} \triangleq \mathbb{E}\left[\sum_{s=t}^{T} R_{A_s,s} \middle| \theta, A_{t:T}\right] = \sum_{s=t}^{T} \mu(\theta, A_s)$.
3. (MAB with independent arms only) The finite-sample mean-reward estimate

$$M_t^{\text{fin}} \triangleq \sum_{s=t}^{T} \hat{\mu}_{I,t}(A_s),$$

where $\hat{\mu}_{I,t}(a) \triangleq \mathbb{E}\left[\mu(\theta, a) \,|H_T, A_s = a, \forall s = t, \ldots, T\right]$ that indicates the best estimate for the mean reward of an arm a that the DM can infer through a finite number of observations.[4]

[4]This metric is valid only when the arms and their associated priors are independent. Using the notation of (6), we further need to assume that the noise variable takes the form $\xi_t \triangleq (\xi_{a,t})_{a\in\mathcal{A}}$ where $\xi_{a,t}$ independent across a. In order for the DM to retrieve maximal information about a particular arm a, it is required to pull the arm a throughout the entire rest of the horizon (i.e., $A_s = a, \forall s = t, \ldots, T$). The metric $\hat{\mu}_{I,t}(a)$ represents the mean reward estimate

4. The posterior mean $M_t^{\text{Bayes}} \triangleq \sum_{s=t}^{T} \mathbb{E}\left[\mu(\theta, A_s) \,|H_{s-1}, A_s\right]$.

5. The state-action Q-function

$$M_t^Q \triangleq \mathbb{E}\left[\sum_{s=t}^{T} R_{A_s,s} \middle| H_{t-1}, A_t\right] = \mathbb{E}\left[\sum_{s=t}^{T} \mu(\theta, A_s) \middle| H_{t-1}, A_t\right].$$

Recall that $\mu(\theta, a)$ is the mean reward function, defined in (2), that is a deterministic function representing the expected reward of an arm a given the parameters θ.

These metrics differ in the information set on which the conditional expectation of the sum of future rewards, $\sum_{s=t}^{T} R_{A_s,s}$, is taken. The main motivation for deriving this series of metrics is "Rao–Blackwellization," i.e., integrating out some of the randomness in the future reward realizations and the future action selections. More specifically, the metric M_t^{mean} is obtained from M_t^{obs} by integrating out the randomness of immediate reward realization while maintaining the dependency on the (random) parameters θ and the (random) action sequence $A_{t:T}$. The metric M_t^{fin} is motivated from the fact that knowing the true parameters θ is as informative as having an infinite number of observations for each arm, and improves over M_t^{mean} by taking into account how much the DM can learn about θ with a finite number observations (i.e., by integrating out the uncertainties in θ that cannot be identified). Next, under the metric M_t^{Bayes}, the DM earns the expected reward given the posterior distribution at each time, which averages out the uncertainty in θ at each time step. Finally, the metric M_t^Q represents the Q-value of the given policy, i.e., the expected future reward of the policy at a given state (history) and an action (arm), which averages out the all uncertainties that arise after taking the action A_t.

We remark that these reward metrics are mostly taken from Min et al. (2019). While the reward metrics M_t^{obs} and M_t^Q are applicable for the general Markov decision processes, the other three metrics M_t^{mean}, M_t^{fin} and M_t^{Bayes} are valid only for bandit problems, and in particular, M_t^{fin} and M_t^{Bayes} are valid only in a Bayesian setting. In addition, accurate computation of M_t^Q typically requires averaging over many Monte Carlo simulations (e.g., roll-outs) which may be computationally expensive.

Baselines. We further provide a list of baselines as follows:

1. The null baseline $B_t^{\text{null}} \triangleq 0$.

2. The oracle performance $B_t^{\text{oracle}} \triangleq M_t^\star$ where $M_t^\star$ is the reward (measured with the corresponding reward metric) that the action sequence $A_t^\star = \text{argmax}_{a \in \mathcal{A}} \mu(\theta, a)$ achieves in the *same instance*. For example, in a combination with M_t^{mean}, we obtain $B_t^{\text{oracle}} = \sum_{s=t}^{T} \max_{a \in \mathcal{A}} \mu(\theta, a)$.

3. The self-play baseline $B_t^{\text{self}} \triangleq \tilde{M}_t$ where $\tilde{M}_t$ is the reward (measured with the corresponding reward metric) that an independent run of the same algorithm achieves in the same instance.

that the DM will have in this scenario, which also has a dependency on the instance I.

For example, in a combination with M_t^{mean}, we obtain $B_t^{\text{self}} = \sum_{s=t}^{T} \mu(\theta, \tilde{A}_s)$ where $\tilde{A}_{1:T}$ is the action sequence taken in the independent run.

4. The value function $B_t^V \triangleq \mathbb{E}\left[\sum_{s=t}^{T} R_{A_s,s} \middle| H_{t-1}\right]$.

As proven in Theorem 1, each of these baselines can be used in a combination with any of the reward metrics listed above. The baseline B_t^{oracle} is an instance-dependent measure that represents the performance of the omniscient policy that knows the values of true parameters θ. Given that M_t^{mean} is chosen as a coupled reward metric, the gap $B_t^{\text{oracle}} - M_t^{\text{mean}}$ reduces to the "regret" which is a measure of suboptimality that has been widely used in bandit studies. This choice of baseline is natural when we expect adaptive algorithm to have small average regret. It can be less effective in problems with a short time horizon, where the reward earned by an oracle is not an attainable baseline.

The baseline B_t^{self} utilizes an independent run of the same randomized policy under the same instance. The idea of self-play was adopted from Boutilier et al. (2020) while we make a generalization regarding the choice of reward metric and provide a formal proof of its validity. It effectively centers the reward metric, i.e., $\mathbb{E}[M_t - B_t^{\text{self}}] = 0$, which helps stabilize gradient estimates. In our numerical experiments, B_t^{self} shows an impressive performance across the different settings, though it effectively requires the computational effort of running twice as many simulations.

Finally, the baseline B_t^V is constructed analogously to the reward metric M_t^Q, and it represents the average performance of the given policy at the given state. In a combination with M_t^Q, the gap $M_t^Q - B_t^V$ measures the relative benefit of the chosen action compared to the average, which is also known as the advantage function. Like M_t^Q, however, this baseline does not have a closed-form expression. The baseline B_t^V can be understood as averaging the result of B_t^{self} (applied with the posterior mean reward metric) across many independent runs of the algorithm. The randomized baseline B_t^{self} has higher variance, but can be calculated at much lower computational cost.

Implementation issues. If we are equipped with a simulator that can generate instances with full information, it is straightforward to compute the reward metrics and the baselines listed above (apart from the computational efficiency). If we are running the algorithm in the real world situation, however, we may not be able to identify their values as we do not have an access to unrevealed information such as true model parameters θ. Nevertheless, in the Bayesian setting, we can overcome this issue by sampling the unobserved variables at the end of an episode: For example, after completing an episode, we can sample $\tilde{\theta} \sim \mathcal{P}(Y_T)$ as if we perform one more step of TS, and plug them into the formulas for reward metric or baseline. This is valid since $\tilde{\theta}$ is identically distributed with the true parameters θ given the observations revealed in that episode, by the virtue of posterior distribution, and therefore the resulting gradient estimates $\tilde{G}$ will also be identically distributed with the true one G.

Note that any model mis-specification can lead to a bias of the gradient estimator. More specifically, if the prior distribution or the reward distribution is mis-specified (e.g., the value of

noise variance σ_a^2 is incorrect in Example 1), the reward metrics M_t^{fin} and M_t^{Bayes} will result in biased estimates. If the mean reward function $\mu(\cdot,\cdot)$ is incorrect, furthermore, all the reward metrics other than M_t^{obs} will suffer from the bias. We expect that the users can determine whether there is an bias during the training process and adopt a more robust metric if needed.

4.4. Variance Comparison

The variance of a gradient estimator is a crucial factor for the performance of policy gradient. In this section, we provide an analysis than can provide theoretical comparisons between estimators of the form (19), including many of the estimators in Section 4.3.

To begin, note that for an admissible estimator of the form (19), we have

$$\mathbb{E}\left[G\right] = \mathbb{E}\left[G^{M,B}\right] = \mathbb{E}\left[\sum_{t=1}^{T} S_t \cdot (M_t - B_t)\right] = T \times \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} G_t^{M,B}\right] = \mathbb{E}\left[T \times G_\tau^{M,B}\right],$$

where $G_t^{M,B} \triangleq S_t \cdot (M_t - B_t)$, and $\tau \in [T]$ is a random time index that is independently and uniformly distributed. Thus, given any admissible estimator $G^{M,B}$, we can construct a related *single time* estimator $T \times G_\tau^{M,B}$ that is also unbiased. Loosely speaking, this estimator estimates the gradient based on the impact of an action taken at a single, randomly chosen decision epoch τ, rather than considering all decision epochs. Moreover, the simpler, single time estimator is more amenable to analysis.

In the next theorem, we further provide a comparison between two single time gradient estimators in terms of the variance they induce. For two square symmetric matrices A and B, we say $A \preceq B$ if and only if $B - A$ is a positive semi-definite matrix. This gives a partial ordering of symmetric matrices.

Theorem 2 (Variance reduction). *Consider two reward metric and baseline pairs $(\underline{M}_t, \underline{B}_t)$ and $(\overline{M}_t, \overline{B}_t)$ that satisfy*

$$\underline{M}_t - \underline{B}_t = \mathbb{E}\left[\overline{M}_t - \overline{B}_t \middle| \mathcal{G}_t\right], \quad \forall t \in [T], \tag{23}$$

for some $\mathcal{G}_t \supseteq \sigma(H_{t-1}, \tilde{\theta}_t)$. Let $\underline{G}_\tau$ and $\overline{G}_\tau$ be corresponding single time gradient estimators, respectively, i.e.,

$$\underline{G}_\tau \triangleq S_\tau \cdot (\underline{M}_\tau - \underline{B}_\tau), \quad \overline{G}_\tau \triangleq S_\tau \cdot (\overline{M}_\tau - \overline{B}_\tau).$$

Then, $\overline{G}_\tau$ exhibits a smaller variance than $\underline{G}_\tau$, in the sense that

$$\mathrm{Cov}[\underline{G}_\tau] \preceq \mathrm{Cov}[\overline{G}_\tau]. \tag{24}$$

Proof. Fix $t \in [T]$. By the law of total covariance and conditioning on $\mathcal{G}_t$,

$$\begin{aligned}
\mathrm{Cov}[\overline{G}_t] &= \mathrm{Cov}\left[\mathbb{E}(\overline{G}_t|\mathcal{G}_t)\right] + \mathbb{E}\left[\mathrm{Cov}\left(\overline{G}_t|\mathcal{G}_t\right)\right] \\
&\succeq \mathrm{Cov}\left[\mathbb{E}(\overline{G}_t|\mathcal{G}_t)\right] \\
&= \mathrm{Cov}\left[\mathbb{E}\left(S_t \cdot (\overline{M}_t - \overline{B}_t)\middle|\mathcal{G}_t\right)\right] \\
&= \mathrm{Cov}\left[S_t \cdot \mathbb{E}\left(\overline{M}_t - \overline{B}_t\middle|\mathcal{G}_t\right)\right] \\
&= \mathrm{Cov}\left[S_t \cdot (\underline{M}_t - \underline{B}_t)\right] \\
&= \mathrm{Cov}[\underline{G}_t],
\end{aligned} \tag{25}$$

where the inequality in the second step follows from the fact that every covariance matrix is positive semi-definite.

Now, note that

$$\mathbb{E}\left[\overline{G}_\tau\middle|\tau\right] = \mathbb{E}\left[S_\tau \cdot \mathbb{E}\left[\overline{M}_\tau - \overline{B}_\tau\middle|\mathcal{G}_\tau\right]\middle|\tau\right] = \mathbb{E}\left[S_\tau \cdot (\underline{M}_\tau - \underline{B}_\tau)|\,\tau\right] = \mathbb{E}\left[\underline{G}_\tau|\,\tau\right].$$

Then, applying the law of total covariance again, this time conditioning on τ,

$$\begin{aligned}
\mathrm{Cov}[\overline{G}_\tau] &= \mathrm{Cov}\left[\mathbb{E}(\overline{G}_\tau|\tau)\right] + \mathbb{E}\left[\mathrm{Cov}\left(\overline{G}_\tau|\tau\right)\right] \\
&= \mathrm{Cov}\left[\mathbb{E}(\underline{G}_\tau|\tau)\right] + \mathbb{E}\left[\mathrm{Cov}\left(\overline{G}_\tau|\tau\right)\right] \\
&\succeq \mathrm{Cov}\left[\mathbb{E}(\underline{G}_\tau|\tau)\right] + \mathbb{E}\left[\mathrm{Cov}\left(\underline{G}_\tau|\tau\right)\right] \\
&= \mathrm{Cov}[\underline{G}_\tau].
\end{aligned}$$

Here, the inequality follows from (25). This concludes the proof. ■

Theorem 2 provides a pairwise comparison of two single time gradient estimators (i.e., $\overline{G}_\tau$ and $\underline{G}_\tau$), when their reward metrics and baselines are related by (23). Ideally we would like a comparison between the variance of the original gradient estimators (i.e., $\mathrm{Cov}\left[\sum_{t=1}^T \underline{G}_t\right]$ and $\mathrm{Cov}\left[\sum_{t=1}^T \overline{G}_t\right]$). However, this is challenging due to the interdependence across time between the score functions and the reward metrics. Nevertheless, we believe that Theorem 2 is informative, and the ordering it implies is consistent with the numerical performance results we will see in Section 5.

Theorem 2 implies that the reward metric based on the smaller information set (i.e., through more averaging) produces a more precise gradient estimator than one based on the larger information set (i.e., with less averaging). This is the same insight that drives the Rao-Blackwell theorem.

In the development of the reward metrics in Section 4.3, we have argued that some reward metrics are motivated from the others via Rao-Blackwellization. In fact, the relationship (23) holds among the reward metrics M_t^{obs}, M_t^{mean}, M_t^{fin}, and M_t^Q (not including M_t^{Bayes}). Indeed, an application of Theorem 2 immediately yields the following ordering among the gradient estimators:

Corollary 1.

$$\mathrm{Cov}[G_\tau^{M^{\mathrm{obs}},B}] \succeq \mathrm{Cov}[G_\tau^{M^{\mathrm{mean}},B}] \succeq \mathrm{Cov}[G_\tau^{M^{\mathrm{fin}},B}] \succeq \mathrm{Cov}[G_\tau^{M^{Q},B}],$$

for any choice of baseline B from B^{null}, B^{oracle}, B^{self}, and B^{V}. Here, note that the baselines B_t^{oracle} and B_t^{self} require a coupled reward metric. We assume they are coupled to the corresponding reward metric in use in each estimator.

5. Numerical Experiments

In this section, we report the simulation results. We aim to illustrate the flexibility of our proposed framework as a meta-learning platform for bandit tasks, compare the gradient estimators with different choices of reward metric and baseline, and highlight the performance of optimized sampling policies in a comparison with the other state-of-the-art algorithms.

Setup. We examine the following bandit environments for each of which we (meta-)parameterize a naïve TS policy and optimize it using the policy gradient framework:

1. Standard Gaussian MAB ($K = 10, T = 500$), where all ten arms have the same prior distribution and the same noise variance. This is a typical setting that has been considered in many prior studies.
2. Heteroscedastic Gaussian MAB ($K = 5, T = 50$), where five arms have very different noise variances. Since each arm requires a different amount of effort to learn its unknown mean value, it is important to incorporate information about the noise variances into the decision making, which standard TS does not do.
3. Many-arm Gaussian MAB ($K = 20, T = 20$), where the number of arms equals to the length of time horizon. In this setup, there is no hope of discovering the true optimal arm. Since standard TS does not take into account the horizon information in its arm-selection rule, it tries to explore all arms to the end of the horizon, which is very wasteful exploration.
4. Non-stationary Gaussian MAB ($K = 5, T = 1000$), where the mean reward value of each arm changes at random times without being noticed by the DM. To deal with non-stationarity, we make a simple heuristic modification to TS and utilize the policy gradient framework to optimize the meta-parameters introduced for this modification.
5. News article recommendation application ($K = 20, T = 100000$), where the algorithm at each user arrival needs to recommend one out of 20 articles. This is a classical contextual bandit problem, and we construct a simulated environment using a real-world dataset.

The first three settings are usual stationary Gaussian MAB tasks introduced in Example 1. In all of three settings, we implement TS with parameterized posterior reshaping, described in Example 3. The reason of adopting the same parameterization is to verify that our proposed framework achieves the goal of meta-learning: the policy gradient procedure finds the choice of

meta-parameters $\lambda \in \mathbb{R}^{4K}$ from Example 3 that is optimized for each of the bandit settings, resulting in the algorithm that is trained to exploit the structure in each setting and performs no worse than the standard version of TS. We highlight that the optimized behavior differs substantially across settings and at times differs substantially from TS.

The other two settings consider more realistic environments: the non-stationary Gaussian MAB is a variant of usual Gaussian MAB (Example 1) that additionally introduces some unpredictable changes in the performance of the arms, and the news article recommendation setting is a well-known real-world example of contextual bandit task. See §5.4 and §5.5 for the detailed setup. Since exact Bayesian inference is difficult to implement in these settings, we introduce simple sampling policies that utilize some approximate posterior distribution and use them as base policies to parameterize and optimize. We highlight that our policy gradient framework works effectively in these realistic settings, even under the adoption of approximate Bayesian inference framework.

Training. We implement the policy gradient algorithm based on the gradient estimator (19) with different combinations of reward metric M and baseline B. In each policy gradient iteration, we compute the batch gradient, i.e., the average gradient measured across a set of independently generated bandit instances, where the batch size (the number of instances) ranges from 800 to 5,000 across the settings. The Adam optimizer (Kingma and Ba, 2014) is used to perform the gradient ascent steps and its learning rate is properly chosen in each experiment.

To facilitate an accurate comparison between the gradient estimators, the estimators share all the randomness in the instance generation and the random action selection. That is, in the notation of (6), the same realizations of noise variables (ϵ_t) and (ξ_t) are used for the simulation of different policy gradient estimators.

Evaluation. As a suboptimality measure of a bandit algorithm, we use the (Bayesian) regret defined as follows:

$$\textsc{Regret}(\pi) \triangleq \mathbb{E}\left[\sum_{t=1}^{T} \max_{a\in\mathcal{A}}\{\mu(\theta, a)\} - \mu(\theta, A_t)\right], \tag{26}$$

which is measured via sample average approximation in our simulation.[5] When computing the gradient estimator during the training process, we obtain as a side product the regret that the algorithm incurs in each training batch, and we report this trajectory of regret as a learning curve of the policy gradient optimization. We naturally expect that the regret decreases as training proceeds. Finally, we measure the regret of the trained policies (and the other bandit algorithms listed below) on the evaluation batch, which is a set of instances generated independently of the training batches. As done in training, the same set of instances are used for evaluating all the policies so as to facilitate accurate comparisons among them.

Competing bandit algorithms. For the stationary Gaussian MAB settings, we use the following

[5]Strictly speaking, this formula is valid only for the stationary MABs. In the non-stationary Gaussian MAB (§5.4) and news article recommendation (§5.5) settings, we use slightly different definitions that reflect the non-stationarity of θ and the contextual information, respectively.

state-of-the-art bandit algorithms as benchmarks: the Bayesian upper confidence bound (Kaufmann et al., 2012) (BAYES-UCB, with a quantile of $1 - \frac{1}{t}$), information-directed sampling (Russo and Van Roy, 2018) (IDS), and the optimistic Gittins index[6] (Gutin and Farias, 2016) (OGI). We compare the performance of the trained TS policies with these algorithms. For the other settings, we use as benchmarks the other variants of TS or UCB algorithms adapted to the task.

Implementation. All the code is written in Python, and the training module is implemented using Tensorflow to utilize the automatic gradient calculation and the Adam optimizer. We use 64-bit floating point for computation of gradient estimator.

5.1. Gaussian MAB in a Standard Setting ($K = 10, T = 500$)

We first report the result for Gaussian MAB with 10 arms and 500 time periods. More specifically, we are given ten independent arms with identical prior distributions: for each arm $a = 1, \ldots, K$ and time $t = 1, \ldots, T$, we assume that

$$\theta_a \sim \mathcal{N}(0, 1^2), \quad R_{a,t}|\theta_a \sim \mathcal{N}(\theta_a, 1^2). \tag{27}$$

This setup has been also examined in the prior literature (Gutin and Farias, 2016; Russo and Van Roy, 2018).

For policy gradient optimization of TS with parameterized posterior reshaping, we adopt the various combinations of reward metric M and baseline B for the gradient estimator $G^{M,B}$. The initial values for the meta-parameters λ are chosen in the way that the corresponding policy is identical to the standard TS. The training batch size is set to 5,000 and the learning rate for Adam optimizer is set to 0.01.

Figure 1 shows the learning curves obtained in our simulated training, and Table 1 reports the performance of the trained TS policies as well as the other algorithms being compared. In every combination of reward metric and baseline, we observe a steady improvement in performance over the course of the training process (starting from the standard TS). The training performance largely depends on the choice of baseline: with baseline B^{oracle} or B^{self} the algorithm shows an impressive progress, catching the state-of-the-art algorithms within 300 policy gradient iterations and ending up with policies that improve over the standard TS by 23% in terms of regret.

[6]There are two free parameters in OGI. We use a one-step look-ahead and a discount factor of $\gamma_t = 1 - \frac{1}{t}$, which was the primary focus of Gutin and Farias (2016).

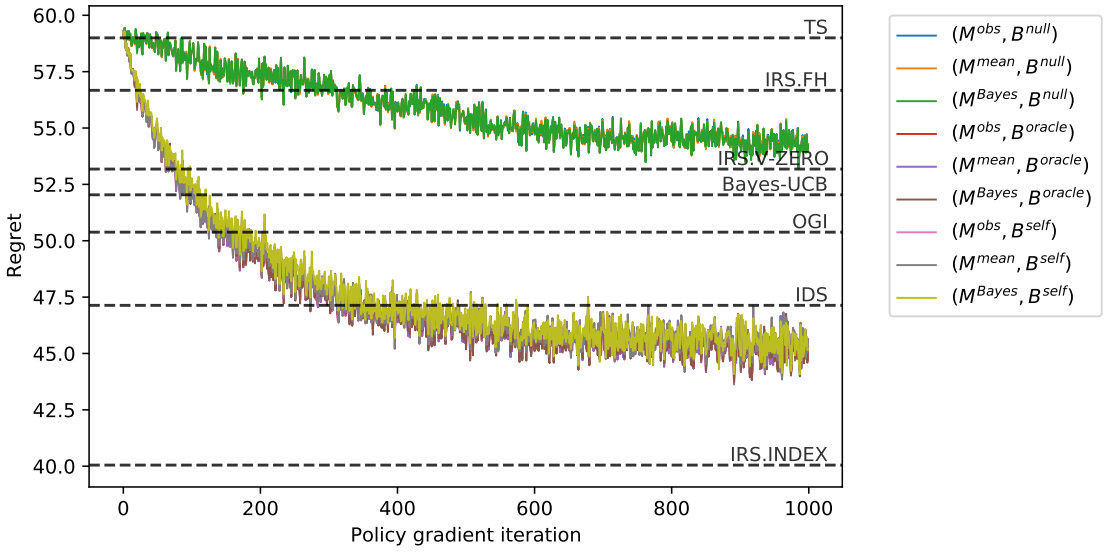

Figure 1: Learning curves in the standard Gaussian MAB task ($K = 10, T = 500$). Thompson sampling is parameterized with 40 meta-parameters that are being optimized using policy gradient with batch size 5,000 and learning rate 0.01. Each curve corresponds to a specific choice of reward metric M and baseline B in gradient estimation (19). The k^{th} data point in each curve reports the average regret on the k^{th} training batch. The dashed horizontal lines represent the performance of benchmark policies measured with 20,000 instances (also reported in Table 1).

Algorithm	Reward metric	Baseline	$\mathrm{Tr}(\mathrm{Cov}[G^{M,B}])$ ($\times 10^6$)	Regret (s.e.)
Trained TS	M^{obs}	B^{null}	3417.37	54.416 (0.208)
	M^{mean}	B^{null}	3410.17	54.251 (0.197)
	M^{Bayes}	B^{null}	3402.82	54.454 (0.207)
	M^{obs}	B^{oracle}	7.84	45.699 (0.306)
	M^{mean}	B^{oracle}	7.84	**45.360** (0.299)
	M^{Bayes}	B^{oracle}	7.97	45.335 (0.306)
	M^{obs}	B^{self}	4.50	45.913 (0.306)
	M^{mean}	B^{self}	4.50	45.409 (0.295)
	M^{Bayes}	B^{self}	6.51	46.331 (0.318)
Naïve TS	–	–	–	58.999 (0.191)
Bayes-UCB	–	–	–	52.038 (0.186)
OGI	–	–	–	50.381 (0.348)
IDS	–	–	–	47.135 (0.335)
IRS.FH	–	–	–	56.672 (0.180)
IRS.V-Zero	–	–	–	53.179 (0.187)
IRS.Index	–	–	–	**40.048** (0.251)

Table 1: Performance of the policies in the standard Gaussian MAB task ($K = 10, T = 500$). Trained TS policies use the meta-parameters that are obtained at the end of training procedure, i.e., the ones found after performing 1,000 policy gradient iterations (Figure 1). The performance is measured in regret, defined in (26), and computed via sample average approximation over 20,000 independent instances, and reported with the standard error. The best results are emphasized with bold letters.

5.2. Gaussian MAB with Heteroscedastic Arms ($K = 5, T = 50$)

We now explore a different configuration of the Gaussian MAB under which the naïve TS performs particularly poorly. We consider five arms that have significantly different noise variances: For each arm $a = 1, \ldots, 5$ and time $t = 1, \ldots, 50$, we assume that

$$\theta_a \sim \mathcal{N}(0, 1^2), \quad R_{a,t}|\theta_a \sim \mathcal{N}(\theta_a, \sigma_a^2), \quad \text{where} \quad \sigma_{1:5}^2 \triangleq (0.1,\ 0.4,\ 1,\ 4,\ 10). \tag{28}$$

Note that it is crucial for the algorithms to consider the heterogeneity in the reward variances since the variance σ_a^2 determines how much the decision maker can learn about the unknown mean reward θ_a within a finite number of observations: in order for the posterior distribution to concentrate so as to have the standard deviation of 0.1, for example, a single observations is enough for arm 1 whereas 100 and 10,000 observations are required for arm 3 and arm 5, respectively. This is especially important when the time horizon is short, as in this case.

We use the training batches of size 1,000 for gradient estimation, and the Adam optimizer with learning rate of 0.05 for policy gradient, and the evaluation batch of size 10,000 for evaluation. While every combination of reward metric and baseline shows a very stable progress throughout the policy gradient procedure, as shown in Figure 2, we observe that the baseline B^{self} works slightly

better than the baseline B^{oracle}, and so does B^{oracle} than B^{null}.

The evaluation results are shown in Table 2. We immediately observe that naïve TS and Bayes-UCB particularly perform poorly as they make decisions based only on the posterior at each moment without incorporating the noise variances into consideration. As the results show, by optimizing posterior reshaping parameters, we can make TS to trade off exploitation and exploration much more precisely, so that we can achieve a surprising improvement over TS by 35%–40%.

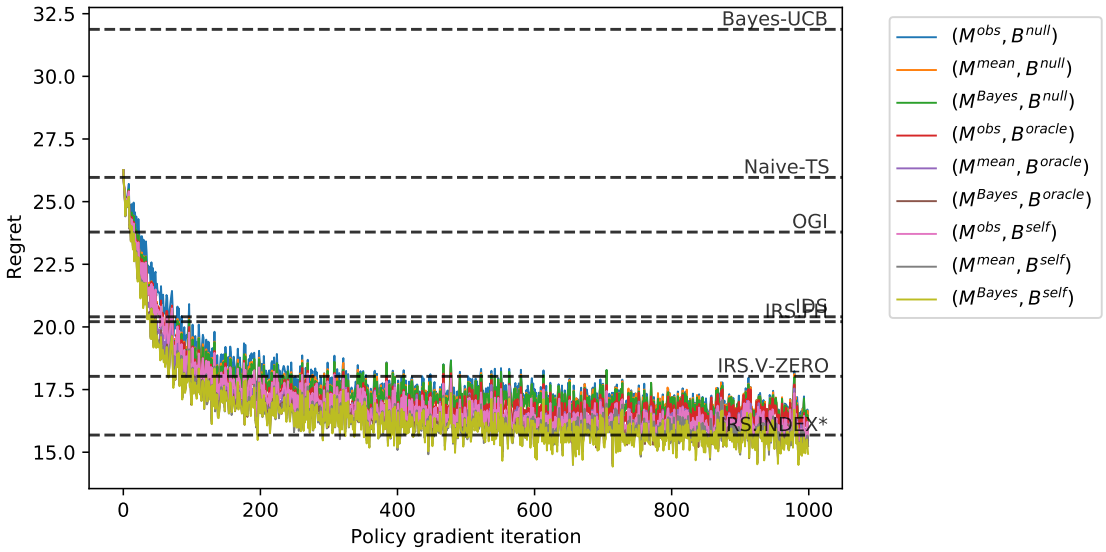

Figure 2: Learning curves in the heteroscedastic Gaussian MAB task ($K = 5, T = 50, \sigma^2_{1:5} = (0.1, 0.4, 1, 4, 10)$). Thompson sampling is parametrized with 20 meta-parameters that are being optimized using policy gradient with batch size 1,000 and learning rate 0.05. Each curve corresponds to a specific choice of reward metric M and baseline B in gradient estimation (19). The k^{th} data point in each curve reports the average regret on the k^{th} training batch. The dashed horizontal lines represent the performance of benchmark policies measured with 10,000 instances (also reported in Table 2).

Algorithm	Reward metric	Baseline	$\mathbf{Cov}[G^{M,B}]$ ($\times 10^4$)	Regret (s.e.)
Trained TS	M^{obs}	B^{null}	133.85	16.654 (0.200)
	M^{mean}	B^{null}	75.33	16.723 (0.201)
	M^{Bayes}	B^{null}	66.20	16.546 (0.200)
	M^{obs}	B^{oracle}	78.76	16.315 (0.201)
	M^{mean}	B^{oracle}	22.73	15.864 (0.201)
	M^{Bayes}	B^{oracle}	9.75	15.693 (0.194)
	M^{obs}	B^{self}	59.97	15.885 (0.199)
	M^{mean}	B^{self}	49.15	15.417 (0.194)
	M^{Bayes}	B^{self}	38.99	**15.313** (0.194)
Naive-TS	–	–	–	25.967 (0.158)
Bayes-UCB	–	–	–	31.875 (0.256)
OGI	–	–	–	23.785 (0.227)
IDS	–	–	–	20.405 (0.205)
IRS.FH	–	–	–	20.209 (0.169)
IRS.V-Zero	–	–	–	18.027 (0.175)
IRS.Index*	–	–	–	**15.685** (0.200)

Table 2: Performance of the policies in the heteroscedastic Gaussian MAB task ($K = 5, T = 50, \sigma^2_{1:5} = (0.1, 0.4, 1, 4, 10)$). Trained TS policies use the meta-parameters that are obtained at the end of training procedure, i.e., the ones found after performing 1,000 policy gradient iterations (Figure 2). The performance is measured in regret, defined in (26), and computed via sample average approximation over 10,000 independent instances, and reported with the standard error. The best results are emphasized with bold letters.

5.3. Gaussian MAB with an Excessive Number of Arms ($K = 20, T = 20$)

We investigate Gaussian MAB with an excessive number of arms, i.e., too many arms compared to the length of time horizon. More specifically, we consider 20 arms and 20 time periods: For each arm $a = 1, \dots, 20$ and time $t = 1, \dots, 20$, we assume that

$$\theta_a \sim \mathcal{N}(0, 1^2), \quad R_{a,t}|\theta_a \sim \mathcal{N}(\theta_a, 1^2). \tag{29}$$

This setup is motivated from Russo and Van Roy (2022) in which the authors posit an extreme example where TS faces an infinite number of arms with identical priors. In such an example, TS keeps pulling a new arm throughout the entire process, since with zero probability the same arm gets the largest sampled mean $\tilde{\theta}_{a,t}$ more than once, and as a result, TS does not utilize any information obtained from the past pulls, always earning the prior mean $\mathbb{E}[\theta_a]$ in expectation at each time. We aim to see whether TS can resolve this over-exploration issue if optimized via policy gradient.

As in the previous setup, we use the training batches of size 1,000 and the learning rate of 0.05 for Adam optimizer, and the evaluation batch of size 10,000 for evaluation.

The simulation results are reported in Figure 3 and Table 3. As expected, the naïve TS exhibits an extremely poor performance in this setup. At the end of the training process, we observe that all trained algorithms have almost identical performance regardless of the choice of reward metric and baseline in their gradient estimation. During the initial phase of training (i.e., during the first 300 iterations), in contrast, we can observe that the baseline B^{null} performs better than the baseline B^{oracle}. This is in contrast with the heteroscedastic noise example. The intuitive reason is that, in the current setup, the oracle performance does not provide a (nearly) attainable benchmark for an adaptive algorithm.

Figure 4 visualizes how the parameterized TS gets improved over the course of training. It shows the distribution of pulls that each arm gets, measured at the beginning, middle, and end of the training. At the beginning, since the initial values for meta-parameters are chosen to yield the standard version of TS, it allocates the pulls evenly across the arms, i.e., one pull per one arm in average. As training proceeds, we can observe that the distribution becomes more skewed, i.e., the algorithm effectively ignored some arms as it realizes that it is wasteful to explore all of the arms. The set of ignored arms are randomly determined during the course of policy gradient optimization: while not reported here, across the choices of reward metric and baseline, the shape of the distribution looks alike but the ordering of arms in the distribution is observed to be different.

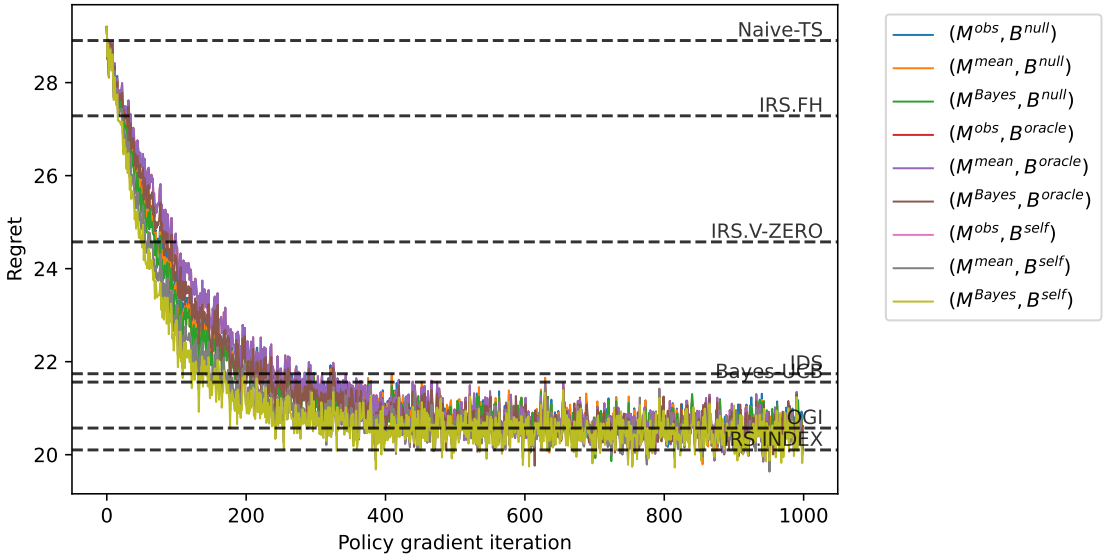

Figure 3: Learning curves in the many-arm Gaussian MAB task ($K = 20, T = 20$). Thompson sampling is parameterized with 80 meta-parameters that are being optimized using policy gradient with batch size 1,000 and learning rate 0.05. Each curve corresponds to a specific choice of reward metric M and baseline B in gradient estimation (19). The k^{th} data point in each curve reports the average regret on the k^{th} training batch. The dashed horizontal lines represent the performance of benchmark policies measured with 10,000 instances (also reported in Table 3).

Algorithm	Reward metric	Baseline	$\mathbf{Cov}[G^{M,B}]$ ($\times 10^4$)	Regret (s.e.)
Trained TS	M^{obs}	B^{null}	14.99	20.442 (0.125)
	M^{mean}	B^{null}	13.00	20.568 (0.126)
	M^{Bayes}	B^{null}	10.03	20.539 (0.127)
	M^{obs}	B^{oracle}	40.80	20.570 (0.126)
	M^{mean}	B^{oracle}	40.80	20.286 (0.125)
	M^{Bayes}	B^{oracle}	30.73	20.510 (0.125)
	M^{obs}	B^{self}	8.78	20.432 (0.126)
	M^{mean}	B^{self}	8.78	**20.210** (0.125)
	M^{Bayes}	B^{self}	6.07	20.375 (0.126)
Naive-TS	–	–	–	28.905 (0.095)
Bayes-UCB	–	–	–	21.561 (0.123)
OGI	–	–	–	20.573 (0.125)
IDS	–	–	–	21.741 (0.119)
IRS.FH	–	–	–	27.286 (0.098)
IRS.V-Zero	–	–	–	24.575 (0.105)
IRS.Index	–	–	–	**20.103** (0.125)

Table 3: Performance of the policies in the many-arm Gaussian MAB task ($K = 20, T = 20$). Trained TS policies use the meta-parameters that are obtained at the end of training procedure, i.e., the ones found after performing 1,000 policy gradient iterations (Figure 3). The performance is measured in regret, defined in (26), and computed via sample average approximation over 10,000 independent instances, and reported with the standard error. The best results are emphasized with bold letters.

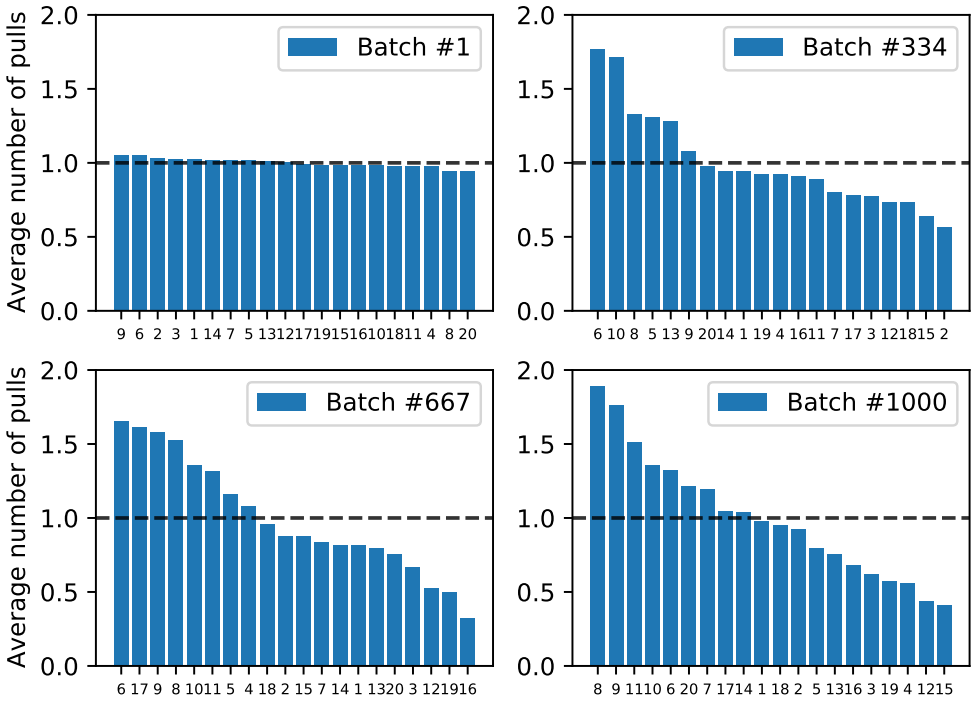

Figure 4: Average arm-selection counts in the many-arm Gaussian MAB task ($K = 20, T = 20$). The plots describe the average behavior of the policy observed during the k^{th} batched training, for $k \in \{1, 334, 667, 1000\}$: a bar in each plot reports how many times in average the policy selects a particular arm across 1,000 instances contained in the k^{th} training batch. The x-axis represents the arm indices rearranged in the decreasing order of the average number of pulls.

5.4. Gaussian MAB with Non-stationary Arms

We examine a variant of Gaussian MAB environment, also known as an abruptly changing environment (Garivier and Moulines, 2011), in which the reward distribution of each arm changes stochastically over time. We consider an instance with five arms and 1,000 time periods ($K = 5$, $T = 1000$) and assume that the reward distribution of arm a at time t is given by a normal distribution with a time-varying mean $\theta_{a,t}$ and a unit variance. As a stochastic process, the sequence of mean reward values $(\theta_{a,t})_{t\in[T]}$ is generated independently per arm as follows:

$$\theta_{a,t} = \begin{cases} \sim \mathcal{N}(0, 1^2) & \text{with probability } \frac{1}{50} \text{ or if } t = 1, \\ \theta_{a,t-1} & \text{with probability } 1 - \frac{1}{50} \end{cases}, \quad R_{a,t}|\theta_{a,t} \sim \mathcal{N}(\theta_{a,t}, 1^2).$$

In words, the mean reward value $\theta_{a,t}$ is being redrawn from the standard normal distribution at random times, which we call *changepoints*. These changepoints are unknown to the DM and determined independently from all the other variables such as the past reward realizations and the DM's past actions. The distance between two changepoints (i.e., the run length) follows a geometric distribution whose average is 50 time periods.

The exact version of TS is difficult to implement in this setting because it needs to keep tracing

the joint posterior distribution of the current mean reward and the most recent changepoint. We instead adopt *discounted Thompson sampling* (Raj and Kalyani, 2017), a simple heuristic modification that leverages the idea of exponential moving average. The discounted TS does not explicitly infer the changepoints but constructs an approximate posterior distribution of current mean reward $\theta_{a,t}$. More specifically, the sufficient statistics $S_{a,t-1}$ and $N_{a,t-1}$ in (15) are replaced with their time-discounted version, $\hat{S}_{a,t-1}$ and $\hat{N}_{a,t-1}$:

$$\hat{S}_{a,t} \triangleq \sum_{s=1}^{t} (\lambda^{\gamma})^{t-s} \times \mathbb{I}_{\{A_s=a\}} R_{A_s,s}, \quad \hat{N}_{a,t} \triangleq \sum_{s=1}^{t} (\lambda^{\gamma})^{t-s} \times \mathbb{I}_{\{A_s=a\}}, \tag{30}$$

where $\lambda^{\gamma} \in (0,1)$ is a discount factor that determines how quickly the algorithm forgets old observations. It then can take into account the fact that recent observations are more informative to infer the current mean reward.

Combining this discounting scheme with the parameterization for Gaussian MAB described in Example 3, we implement and train a TS policy that samples the parameters $\tilde{\theta}_{a,t}$ from

$$\tilde{\theta}_{a,t} \sim \mathcal{N}\left(\frac{\lambda_a^m + \lambda_a^{\sigma} \cdot \hat{S}_{a,t-1}}{1 + \lambda_a^{\sigma} \cdot \hat{N}_{a,t-1}}, \ \frac{\lambda_a^v}{1 + \lambda_a^{\sigma} \cdot \hat{N}_{a,t-1}} \right).$$

This involves $3K + 1$ $(= 16)$ meta-parameters, $((\lambda_a^m, \lambda_a^{\sigma}, \lambda_a^v)_{a \in \mathcal{A}}, \lambda^{\gamma})$, which we aim to optimize through our policy gradient framework. Note that the likelihood function of this reshaped posterior distribution is differentiable with respect to λ^{γ}. We initialize these meta-parameters as $\lambda_a^m = 0$, $\lambda_a^{\sigma} = \lambda_a^v = 1$, and $\lambda^{\gamma} = \exp(-1/50)$, and optimize their values using the training batches of size 1,000 and the learning rate of 0.05 for Adam optimizer.

To use as a benchmark, we also implement sliding-window TS (Trovo et al., 2020) that is similar to discounted TS but utilizes naïve moving average:

$$\hat{S}_{a,t} \triangleq \sum_{s=\max(1,t-\tau)}^{t} \mathbb{I}_{\{A_s=a\}} R_{A_s,s}, \quad \hat{N}_{a,t} \triangleq \sum_{s=\max(1,t-\tau)}^{t} \mathbb{I}_{\{A_s=a\}},$$

where $\tau \in \mathbb{Z}$ is the length of sliding-window. This sliding-window TS discards old observations collected before τ periods ago whenever making a decision. Since these formulas are not differentiable with respect to τ, our policy gradient framework cannot be used to optimize the value of τ. In what follows, we simulate the sliding-window TS policies with the different values of τ and use the best performing one as a benchmark.

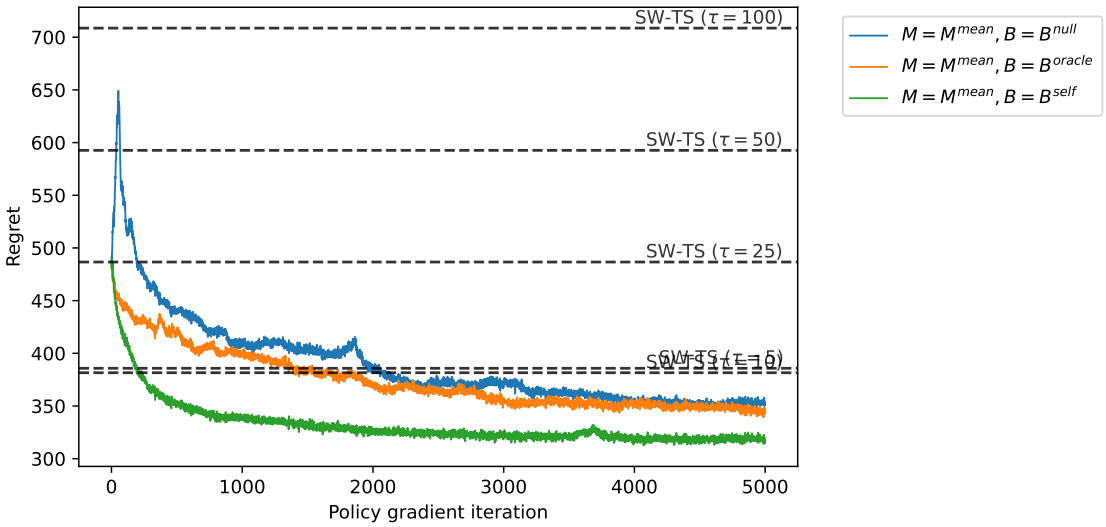

Figure 5: Learning curves in the non-stationary Gaussian MAB task ($K = 5, T = 1000$). Discounted Thompson sampling (Raj and Kalyani, 2017) is used as a base policy and parameterized with 16 meta-parameters that are being optimized using policy gradient with batch size 1,000 and learning rate 0.05. Each curve corresponds to a specific choice of reward metric M and baseline B, and the k^{th} data point reports the average regret on the k^{th} training batch. The dashed horizontal lines represent the performance of sliding-window TS with the different values of window length τ, evaluated with 10,000 instances.

Figure 5 reports the training results. In a comparison with the best sliding-window TS (with the window length $\tau = 10$), our parameterized TS starts to outperform it after 200 training steps under the choice of M^{mean} and B^{self}, and achieves 16% reduction in regret at the end of training procedure. Although the algorithms trained with the other baselines also outperform the best sliding-window TS (achieving 7% and 9% reduction in regret), training with the null-baseline B^{null} experiences some degradation at the beginning of training procedure, suffering from a large variance in the gradient estimation.

This experiment shows that the policy gradient optimization framework is very effective to deal with the non-stationarity issue, which possibly exists in every real-world bandit task. The level of adaptivity (controlled via discounting factor) and the level of exploration (controlled via posterior reshaping parameters) are jointly optimized at the same time without any help of complex analysis. Looking at a different perspective, we can make an analogy to the many-arm Gaussian MAB setting in the sense that learning in a highly non-stationary environment is effectively similar to a situation where the arms have very short lifetimes. The policy gradient procedure (implicitly) finds their effective lifetime length and discovers a learning strategy optimized to that length, where the effectiveness of our method regarding the latter component was verified in the many-arm Gaussian case: this can explain the additional 16% improvement made by the trained TS over the best sliding-window TS.

5.5. News Article Recommendation

Our last numerical experiment demonstrates the use of our framework in the news article recommendation application that has been considered as a representative instance of contextual bandit problem (Li et al., 2010). We consider an online platform who provides personalized recommendations to users who sequentially arrive over time. Upon each user arrival, the platform is given some information about the user such as browsing history (i.e., the user feature) and decides which article to recommend based on the user feature and the past observations. The user either clicks or ignores the recommended article. The platform aims to maximize the total number of user clicks while learning users' preferences on the given set of articles.

Dataset. This numerical experiment is conducted based on a real-world dataset. The dataset of our interest contains user click log for news articles displayed in the Featured Tab of the Today Module on Yahoo! Front Page during the first ten days in May 2009. Each record in the dataset describes one user arrival: a timestamp, a six-dimensional feature vector describing the user, article IDs available at the moment, the article ID actually suggested to the user, and a binary value indicating whether the user clicked the suggested article or not.[7] The dataset contains approximately four million records on each day, and the number of available articles at a time is around 20.

Simulator. We first build a model-based simulator to train and evaluate recommendation algorithms. A logistic model is assumed to describe the click probability: i.e., the likelihood that a user at time t clicks an article a is given by

$$\mathbb{P}\left[R_{a,t}=1 \middle| \mathbf{x}_t, \boldsymbol{\beta}_a\right] = \frac{1}{1+\exp\left(-\mathbf{x}_t^\top \boldsymbol{\beta}_a\right)} = 1 - \mathbb{P}\left[R_{a,t}=0 \middle| \mathbf{x}_t, \boldsymbol{\beta}_a\right],$$

where $\mathbf{x}_t \in \mathbb{R}^6$ is an observable feature vector of the user, which is given in the dataset, and $\boldsymbol{\beta}_a \in \mathbb{R}^6$ is a latent feature vector of the article, which needs to be estimated per article. For each article a that has ever appeared in the dataset, we estimate its latent feature vector $\boldsymbol{\beta}_a$ through an offline ridge regression, i.e., $\hat{\boldsymbol{\beta}}_a \leftarrow \mathrm{argmin}_{\boldsymbol{\beta}\in\mathbb{R}^6} \left\{ \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 + \sum_{t:A_t=a} \log\left(1+\exp(-y_t \mathbf{x}_t^\top \boldsymbol{\beta})\right) \right\}$ where $y_t = +1$ if the user t actually clicked the suggested article A_t (i.e., $R_{A_t,t}=1$) and $y_t=-1$ otherwise (i.e., $R_{A_t,t}=0$).

Having estimated the feature vectors of all articles in the dataset, we build a simulator that can generate random instances as many as we want. A random instance describes a single day episode in which the platform encounters 100,000 user arrivals ($T = 100,000$) given 20 articles to recommend ($K = 20$). More specifically, an instance is generated as follows. First, K article feature vectors $(\boldsymbol{\beta}_a)_{a\in[K]}$ are independently drawn from the six-dimensional normal distribution whose mean vector and covariance matrix are those of the estimated feature vectors. These article feature vectors represent the set of available articles that the platform can recommend in this instance. Second, T user feature vectors $(\mathbf{x}_t)_{t\in[T]}$ are randomly drawn from the actual user feature

[7] The dataset also contains six-dimensional feature vectors of the articles, but we do not utilize these feature vectors in our numerical experiment. See Li et al. (2010) for the detailed specification of those feature vectors.

vectors archived in the dataset on a randomly selected day while their order is preserved. These user feature vectors represent the sequence of user arrivals in this instance. Lastly, for each article $a \in [K]$ and user $t \in [T]$, the counterfactual click event $R_{a,t}$ is generated according to the logistic model described above.

We remark that the choice of logistic model is not an essential part of this experiment. Our policy gradient framework and the recommendation algorithm that will be introduced next do not exploit this modeling assumption. In this paper, we focus on providing a concise demonstration with a possibly simplest simulator design, but one may adopt a more sophisticated generative model such as neural networks in order to reduce the gap between the reality and the simulated environment.

Parameterized linear TS. We now describe a baseline algorithm that we aim to optimize via policy gradient. This algorithm is, often referred to as linear TS (Agrawal and Goyal, 2013), a policy that at each time t draws a six-dimensional vector $\tilde{\boldsymbol{\theta}}_{a,t} \in \mathbb{R}^6$ from the multivariate normal distribution $\mathcal{N}(\mathbf{m}_{a,t-1}, \mathbf{S}_{a,t-1})$ for each article $a \in \mathcal{A}$ where

$$\mathbf{m}_{a,t} \triangleq \mathbf{S}_{a,t}\left(\sum_{s \le t: A_s = a} \mathbf{x}_s R_{a,s} \times \frac{1}{0.05(1-0.05)}\right), \quad \mathbf{S}_{a,t} \triangleq \left(\mathbf{I} + \sum_{s \le t: A_s = a} \mathbf{x}_s \mathbf{x}_s^\top \times \frac{1}{0.05(1-0.05)}\right)^{-1},$$

and then recommends an article $A_t = \mathrm{argmax}_a\, \mathbf{x}_t^\top \tilde{\boldsymbol{\theta}}_{a,t}$. This can be understood as a TS policy developed for the setting where the reward distribution of arm a is given by $\mathcal{N}\left(\mathbf{x}_t^\top \boldsymbol{\theta}_a,\ 0.05(1-0.05)\right)$ for some unknown vector $\boldsymbol{\theta}_a \in \mathbb{R}^6$ and the prior of $\boldsymbol{\theta}_a$ is given by $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Note that this policy does not assume the logistic model (the distribution $\mathcal{N}(\mathbf{m}_{a,t-1}, \mathbf{S}_{a,t-1})$ does not represent the posterior distribution of latent article feature vector $\boldsymbol{\beta}_a$). We simply adopt the linear TS as a baseline because it is easy to implement and parameterize, and we highlight that our policy gradient framework is still effective albeit this model misspecification.

We further introduce a parameterized version of linear TS. Similarly to the parameterization schemes considered previously, we make it sample $\tilde{\boldsymbol{\theta}}_{a,t}$ from $\mathcal{N}(\hat{\mathbf{m}}_{a,t-1}, \hat{\mathbf{S}}_{a,t-1}^{1/2}\mathrm{diag}(\boldsymbol{\lambda}^v)\hat{\mathbf{S}}_{a,t-1}^{1/2})$ where

$$\hat{\mathbf{m}}_{a,t} \triangleq \hat{\mathbf{S}}_{a,t}\left(\boldsymbol{\lambda}^m + \sum_{s \le t: A_s = a} \mathbf{x}_s R_{a,s} \times \frac{1}{\lambda^\sigma}\right), \quad \hat{\mathbf{S}}_{a,t} \triangleq \left(\mathbf{I} + \sum_{s \le t: A_s = a} \mathbf{x}_s \mathbf{x}_s^\top \times \frac{1}{\lambda^\sigma}\right)^{-1}.$$

Here, $\hat{\mathbf{S}}_{a,t-1}^{1/2}$ denotes the square-root matrix of $\hat{\mathbf{S}}_{a,t-1}$, and $\boldsymbol{\lambda}^v \in \mathbb{R}^6$, $\boldsymbol{\lambda}^m \in \mathbb{R}^6$, and $\lambda^\sigma \in \mathbb{R}$ are the meta-parameters to be optimized. Roughly speaking, $\boldsymbol{\lambda}^v$ parameterizes a (directional) multiplier on the variance of the sampling distribution, $\boldsymbol{\lambda}^m$ parameterizes the prior mean, and λ^σ parameterizes the relative precision of one observation. This reduces to the linear TS described above when we take $\boldsymbol{\lambda}^v = \mathbf{1}$, $\boldsymbol{\lambda}^m = \mathbf{0}$, and $\lambda^\sigma = 0.05(1-0.05)$.

In terms of computational efficiency, calculating $\hat{\mathbf{m}}_{a,t}$ and $\hat{\mathbf{S}}_{a,t}$ is costly because it involves computationally heavy matrix operations such as inversion. To reduce this cost, we let the algorithm

to compute $\hat{\mathbf{m}}_{a,t}$ and $\hat{\mathbf{S}}_{a,t}$ occasionally, once in every hundred user arrivals: e.g., once it has computed $\hat{\mathbf{m}}_{a,500}$ and $\hat{\mathbf{S}}_{a,500}$ after the 500th user arrival, then for the next hundred user arrivals, the parameters $\tilde{\boldsymbol{\theta}}_{a,501}, \ldots, \tilde{\boldsymbol{\theta}}_{a,600}$ are independently drawn from $\mathcal{N}(\hat{\mathbf{m}}_{a,500}, \hat{\mathbf{S}}_{a,500}^{1/2}\mathrm{diag}(\boldsymbol{\lambda}^v)\hat{\mathbf{S}}_{a,5000}^{1/2})$.

Simulation result. We train this parameterized linear TS through the simulated environment. The reward metric M^{mean}, the baseline M^{self}, the batch size of 800, and the learning rate of 0.001 are used in this training. As shown in Figure 6, the trained TS policy outperforms the naïve linear TS policy by 40% in terms of average regret (0.4% point increase in terms of click-through-rate). This result demonstrates that our suggested policy gradient framework can be very effective in developing powerful TS policies in complex real-world applications.

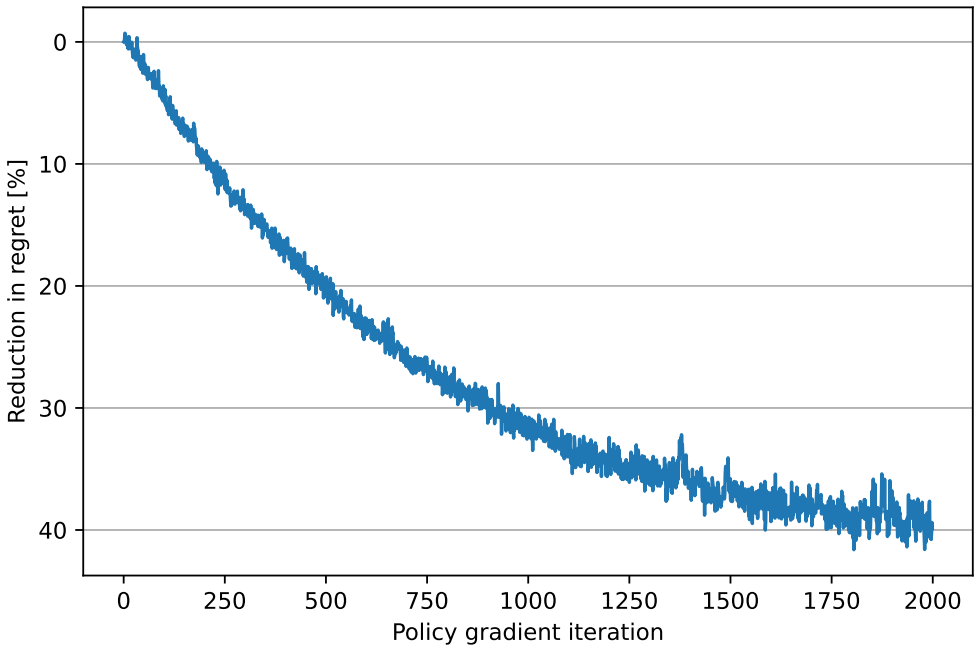

Figure 6: Learning curve in the news article recommendation application. A simulated environment is constructed using a real dataset (Yahoo! Front Page Today Module). Linear Thompson sampling (Agrawal and Goyal, 2013) is used as a base policy and parameterized with 13 meta-parameters. The curve shows the performance improvement over the naïve linear TS, represented in terms of regret reduction, with the choice of reward metric M^{mean}, baseline B^{self}, batch size 800, and learning rate 0.001. One random instance contains a random stream of 100,000 user arrival events and a random set of 20 news articles that can be recommended.

6. Conclusion

In this paper, we have provided a tractable framework for policy gradient optimization of Thompson sampling-based policies. Our main insight is the concept of the "pseudo-action". This simple shift of perspective enables tractable gradient estimation. Building on this, we develop a framework for unbiased policy gradient estimators, based on a choice of admissible reward metric and baseline. Within this framework, we identify several novel policy gradient estimators that are specialized to

our setting, and we are able to compare them theoretically and numerically.

Our work opens up a new direction for the practical solution of bandit problems. Policies can be developed with good baseline performance, based on what we already know about bandits (e.g., Bayesian learning logic), but with a high-dimensional parameter space that can now be optimized using stochastic gradient descent. Our approach provides a specific direction through which computational resources can be leveraged to gain practical policy improvements, and which may show increasing benefits as more resources are deployed.

References

S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135, 2013.

H. Bastani, D. Simchi-Levi, and R. Zhu. Meta dynamic pricing: Transfer learning across experiments. *Management Science*, 68(3):1865–1881, 2021.

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

C. Boutilier, C.-W. Hsu, B. Kveton, M. Mladenov, C. Szepesvari, and M. Zaheer. Differentiable bandit exploration. *arXiv preprint arXiv:2002.06772*, 2020.

O. Chapelle and L. Li. An empirical evaluation of Thompson sampling. In *Advances in Neural Information Processing Systems*, pages 2249–2257, 2011.

Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.

A. Garivier and E. Moulines. On upper-confidence bound policies for non-stationary bandit problems. In *Algorithmic Learning Theory*, pages 174–188, 2011.

P. Glasserman and D. D. Yao. Some guidelines and guarantees for common random numbers. *Management Science*, 38(6):884–908, 1992.

E. Gutin and V. Farias. Optimistic Gittins indices. In *Advances in Neural Information Processing Systems*, pages 3153–3161, 2016.

E. Kaufmann, O. Cappé, and A. Garivier. On Bayesian upper confidence bounds for bandit problems. In *Artificial Intelligence and Statistics*, pages 592–600, 2012.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

B. Kveton, M. Mladenov, C.-W. Hsu, M. Zaheer, C. Szepesvari, and C. Boutilier. Meta-learning bandit policies by gradient ascent. *arXiv preprint arXiv:2006.05094*, 2020.

P. L'Ecuyer. Note: On the interchange of derivative and expectation for likelihood ratio derivative estimators. *Management Science*, 41(4):738–747, 1995.

L. Li, W. Chu, J. Langford, and R.E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670, 2010.

S. Min, C. Maglaras, and C. C. Moallemi. Thompson sampling with information relaxation penalties. In *Advances in Neural Information Processing Systems*, 2019.

V. Raj and S. Kalyani. Taming non-stationary bandits: A bayesian approach. *arXiv preprint arXiv:1707.09727*, 2017.

D. Russo and B. Van Roy. Learning to optimize via information-directed sampling. *Operations Research*, 66 (1):230–252, 2018.

D. Russo and B. Van Roy. Satisficing in time-sensitive bandit learning. *Mathematics of Operations Research*, 2022.

D. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen. A tutorial on Thompson sampling. *Foundations and Trends in Machine Learning*, 11(1):1–96, 2018.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

W. 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.

F. Trovo, S. Paladino, M. Restelli, and N. Gatti. Sliding-window Thompson sampling for non-stationary settings. *Journal of Artificial Intelligence Research*, 68:311–364, 2020.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.