No DiCE: An Investigation of the Bias-Variance Tradeoff in Meta-Gradients

Risto Vuorio[*] University of Oxford
Jacob Beck University of Oxford
Gregory Farquhar DeepMind
Jakob Foerster University of Oxford

Shimon Whiteson University of Oxford

Abstract

Meta-gradients provide a general approach for optimizing the meta-parameters of reinforcement learning (RL) algorithms. Estimation of meta-gradients is central to the performance of these meta-algorithms, and has been studied in the setting of MAML-style short-horizon meta-RL problems. In this context, prior work has investigated the estimation of the Hessian of the RL objective, as well as tackling the problem of credit assignment to pre-adaptation behavior by making a *sampling correction*. However, we show that Hessian estimation, implemented for example by DiCE and its variants, always add bias and can also add variance to meta-gradient estimation. DiCE-like approaches are therefore unlikely to lie on Pareto frontier of the bias-variance tradeoff and should not be pursued in the context of meta-gradients for RL. Meanwhile, the sampling correction has not been studied in the important long-horizon setting, where the inner optimization trajectories must be truncated for computational tractability. We study the bias and variance tradeoff induced by truncated backpropagation in combination with a weighted sampling correction. While prior work has implicitly chosen points in this bias-variance space, we disentangle the sources of bias and variance and present an empirical study which relates existing estimators to each other.

1 Introduction

Recently, meta-gradients have been used for adapting the hyperparameters of state-of-the-art policy gradient RL algorithms online (Flennerhag et al., 2021), as well as for learning black-box RL algorithms from scratch (Oh et al., 2020). The estimation of meta-gradients is central to the performance of these algorithms, but has received little focused study. Deriving unbiased estimators of meta-gradients is more involved than for standard policy gradients because the meta-parameters affect the data distribution used for computing the meta-gradient. Al-Shedivat et al. (2017) present an unbiased meta-gradient estimator that correctly accounts for the changes in the data distribution, by making a *sampling correction* which is added to the direct meta-gradient. However, their derivation and experiments are limited to MAML-style meta-RL algorithms (Finn et al., 2017), which learn the initialization of a policy such that it may achieve good performance after a small number of policy gradient updates. Further work studying meta-gradient estimation for meta-RL is also largely restricted to this context.

Computing the meta-gradient requires differentiating through the agent's parameter update, which may itself include an estimated policy gradient. The correct meta-gradient of a standard policy gradient update can be computed by straightforward backpropagation of a surrogate loss, as is typical in RL. However, Foerster et al. (2018) and a number of subsequent works (Rothfuss et al., 2018; Liu et al., 2019; Farquhar et al., 2019; Mao et al., 2019; Tang et al., 2021) have claimed that an estimate of the *expected* policy Hessian should be included in the meta-gradient estimator, and provide methods for calculating such an estimate or reducing its variance. In this work, we present the surprising result that using an estimate of the expected Hessian in the meta-gradient estimator

[*]risto.vuorio@cs.ox.ac.uk

actually *adds* bias, which can degrade performance. Such a term could only be used if the agent made an *expected* policy gradient update, rather than the sampled update which must be taken using finite data. This focus on estimating expected Hessians may have distracted the community from the real challenges in meta-gradient estimation that arise in the long-horizon setting, where the sampling correction has not been addressed.

In the short-horizon setting, where the sampling correction has been previously explored, the meta-learning algorithm can afford to consider the whole optimization trajectory of the inner learning problem. Due to computational constraints, in a long-horizon setting, the inner loop must be truncated to a small fraction of the full optimization trajectory. As a result, neither the direct meta-gradient nor the sampling correction may be calculated exactly. It is therefore unclear how the truncated sampling correction affects the bias and variance of the meta-gradient estimation. Previous works (Xu et al., 2018; 2020; Oh et al., 2020) have simply opted for truncated meta-gradient estimators without sampling correction, without mention of this potential source of bias. While impressive results have been achieved with these estimators, they are biased due to both truncation and the lack of sampling correction. In this paper, we study the bias-variance tradeoff arising from these two approximations, and how bias and variance may be traded off by varying the truncation horizon and the weight given to the sampling correction. We find that while the sampling correction does not uniformly reduce the bias of the truncated estimators, it nevertheless results in convergence to better local optima. The cost of using the sampling correction is increased variance.

Our contributions can be summarized as follows. (1) We show mathematically why using the expected policy Hessian estimator is in error, and demonstrate that the resulting bias harms performance in practice. (2) We characterize the bias-variance tradeoff due to a truncated optimization horizon and the sampling correction in an empirical study, relating existing approaches to each other and showing how to interpolate between them.

2 Background

We assume that the RL algorithm is learning a control policy in a Markov decision process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, p_0, p, r, \gamma)$, where $\mathcal{S}$ is the space of states, $\mathcal{A}$ the space of actions, $p_0(s_0)$ the distribution over initial states, $p(s_{t+1}|s_t, a_t)$ the conditional probability distribution of the next state given a state and an action, $r(s_t, a_t)$ the reward function for the transition, and $\gamma \in (0, 1]$ a discount factor. A trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots, s_H, a_H) \in \mathcal{T}$, is a sequence of states and actions of length H sampled from the dynamics defined by the MDP and the policy, where the horizon H may be infinite. $\mathcal{T}$ is the space of trajectories. The probability of sampling τ is given by

$$p(\tau;\theta) = p_0(s_0) \prod_{t=0}^{H} \pi(a_t|s_t;\theta) p(s_{t+1}|s_t, a_t), \tag{1}$$

where $\pi(a_t|s_t;\theta)$ is the policy parametrized by $\theta \in \mathbb{R}^{d_\theta}$.

The objective of an RL algorithm is to train a policy, that maximizes the expected discounted sum of rewards when interacting with the environment

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t) \right]. \tag{2}$$

For brevity we write $\mathcal{R}(\tau) = \sum_{t=0}^{H} \gamma^t r(s_t, a_t)$, where s_t and a_t come from τ. In our derivations we use the Monte Carlo return $\mathcal{R}(\tau)$ for simplicity, but in practical algorithms it is usually replaced by an estimated return or advantage to reduce variance.

A policy gradient algorithm updates the parameter by gradient ascent on the policy gradient given by

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[\sum_{t=0}^{H} \nabla_\theta \log \pi(a_t|s_t;\theta) \mathcal{R}(\tau) \right]. \tag{3}$$

Typically, the expectation in equation 3 cannot be evaluated exactly and is instead approximated from samples by

$$\nabla_\theta J(\theta, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{H} \nabla_\theta \log \pi(a_t|s_t; \theta) \mathcal{R}(\tau), \tag{4}$$

where $\mathcal{D} \in \mathcal{T}^N$ is a tuple of N trajectories collected with the policy parametrized by θ also known as a batch. The probability of the batch is $p(\mathcal{D}; \theta) = \prod_{\tau \in \mathcal{D}} p(\tau; \theta)$.

Meta-gradients come into play in the meta-learning problem of optimizing the parameters η of an update function $\Psi(\eta, \theta^i, \mathcal{D}^i) : \mathbb{R}^{d_\eta} \times \mathbb{R}^{d_\theta} \times \mathcal{T}^N \to \mathbb{R}^{d_\theta}$, which computes an updated set of parameters for a policy. For example, a simple update function is

$$\theta^{i+1} = \Psi_{lr}(\eta, \theta^i, \mathcal{D}^i) = \theta^i + \eta \nabla_{\theta^i} J(\theta^i, \mathcal{D}^i), \tag{5}$$

where the meta-parameter is the learning rate of a stochastic policy gradient update. In general, the meta-parameters can appear in any term in the update function. The update functions we consider in this paper use policy gradients to update the parameters but other types of update functions may be considered as well. Meta-gradients are often computed in a setting where the policy is updated K times using the update function and the meta-parameters are optimized to maximize return with each of the $K + 1$ policies. The objective for such K-step meta-gradient is given by

$$J_K(\eta) = \sum_{k=0}^{K} \mathbb{E}_{\{\mathcal{D}^i \sim p(\mathcal{D}^i|\theta^i)\}_{i=0}^{k-1}} \left[\mathbb{E}_{\tau \sim p(\tau|\theta^k)} \left[\mathcal{R}(\tau) \right] \right], \tag{6}$$

where the outer expectation is taken over data sampled with all of the policies along the update trajectory because each θ^k depends on all of the previous data through the update function. Note that by setting the returns of the trajectories for $k < K$ equal to zero, this objective generalizes the MAML-style objective, which only considers the return of the final Kth policy.

3 Related work

The estimation of meta-gradients has been studied in the context of short-horizon multi-task meta-RL algorithms. In MAML (Finn et al., 2017), the initialization of an agent is learned such that its performance after a few policy gradient updates is maximized. Al-Shedivat et al. (2017), derive an unbiased meta-gradient estimator for learning the initialization and extend it to a continuous adaptation setting. A similar derivation is followed by Stadie et al. (2018), where the improved exploratory behavior of the initial policy in the MAML-algorithm is considered additional motivation for using the unbiased estimator. Fallah et al. (2020) study the convergence properties of the unbiased meta-gradient estimator. Learning the initialization necessitates considering the full optimization horizon. In contrast, in the truncated setting which we also consider in this paper, the last batch of experience in a truncation window has no special significance over the earlier ones. Therefore, instead of optimizing the returns at the end of the optimization trajectory as is done in MAML, it makes sense to optimize the returns throughout the lifetime of the agent as done by Antoniou et al. (2018); Chen et al. (2016); Andrychowicz et al. (2016).

Many estimators of the expected Hessian have been developed and applied primarily in the MAML framework. DiCE (Foerster et al., 2018) shows that backpropagating through a stochastic policy gradient update does not result in an estimator of the expected Hessian and proposes a modified surrogate objective, which gives such estimates via backpropagation. ProMP (Rothfuss et al., 2018) further develops the Hessian estimation strategy and derives a lower-variance estimate. Loaded DiCE (Farquhar et al., 2019) derives a new expression of the DiCE objective compatible with value function approximators and introduces a hyperparameter for trading off bias and variance. Liu et al. (2019) and Mao et al. (2019) introduce control variates to reduce the variance of the expected Hessian estimators. Tang et al. (2021) propose a unifying framework for the Hessian estimation problem from the perspective of off-policy evaluation. All of these works consider applications of their Hessian estimators to the MAML-style meta-RL setting. However, we show that besides having high variance, these Hessian estimators add bias to the meta-gradient estimate.

Other parts of the learning algorithms have been optimized via meta-gradients besides the initialization. The objective function of the policy is learned by MetaGenRL (Kirsch et al., 2019) for a

DDPG-like (Lillicrap et al., 2015) algorithm. LPG (Oh et al., 2020) learns the full objective function for an actor-critic-style learner. MODAC (Veeriah et al., 2021) uses meta-gradients for option discovery. Zheng et al. (2020) learn intrinsic rewards with meta-gradients. Meta-gradients have also been used to improve the performance of deep RL algorithms in the single-task setting. Xu et al. (2018) tune the parameters of return estimates in A2C with a meta-gradient. STAC (Zahavy et al., 2020) follows a similar idea applying it to the hyperparameters of Impala (Espeholt et al., 2018). BMG (Flennerhag et al., 2021) uses the same meta-parameterization as Zahavy et al. (2020) but improves the meta-gradient estimation to achieve the state-of-the-art model-free performance on Atari. Our investigations are directed at improving meta-gradient in the truncated meta-optimization setting, which is used by these algorithms.

Outside of reinforcement learning, meta-optimization has been an active area of study as well. Wu et al. (2018) focus on the bias from a short optimization horizon. Metz et al. (2019) investigate the optimization landscape of learned optimizers in supervised learning. For a small number of meta-parameters, forward-mode gradient computation can be used to overcome the memory constraint due to the long optimization horizon Franceschi et al. (2017). These lessons from supervised learning apply to RL as well and along with our findings should be taken into consideration when using meta-gradients.

4 Review of unbiased meta-gradients

A general unbiased meta-gradient estimator for a K-step sequence of updates was first shown by Al-Shedivat et al. (2017); we present a similar derivation here for convenience. For a sequence of K parameter updates the Kth parameter is given by

$$\theta^K = \theta^0 + \sum_{i=0}^{K-1} \Psi(\eta, \theta^i, \mathcal{D}^i), \tag{7}$$

where $\mathcal{D}^i$ are data sampled from the environment. Because the $\mathcal{D}^i$ are random variables, the parameters θ^i for $i > 0$ are random variables too. They depend on the initial parameters θ^0, the meta-parameter η, and the data. The meta-gradient estimator for the objective in equation 6 on the sequence of updates in equation 7 can be derived as follows:

$$\begin{aligned}\nabla_\eta J_K(\eta) &= \nabla_\eta \sum_{k=0}^{K} \mathbb{E}_{\{\mathcal{D}^i \sim p(\mathcal{D}^i|\theta^i)\}_{i=0}^{k-1}} \left[\mathbb{E}_{\tau \sim p(\tau|\theta^k)} \left[\mathcal{R}(\tau) \right] \right] \\ &= \sum_{k=0}^{K} \mathbb{E}_{\substack{\{\mathcal{D}^i\}_{i=0}^{k-1} \\ \tau \sim p(\tau|\theta^k)}} \left[\left(\underbrace{\sum_{j=0}^{k-1} \nabla_\eta \theta^j \nabla_{\theta^j} \log p(\mathcal{D}^j|\theta^j)}_{\texttt{sampling correction}} + \underbrace{\nabla_\eta \theta^k \nabla_{\theta^k} \log p(\tau|\theta^k)}_{\texttt{direct meta-gradient}} \right) R(\tau) \right],\end{aligned} \tag{8}$$

where the $\nabla_\eta \theta^k \in \mathbb{R}^{d_\eta \times d_\theta}$ denotes the derivative of the sequence of updates resulting in policy parameters θ^k with respect to the meta-parameters. A more detailed derivation is provided in Appendix A. Following from the additive sequence of updates in equation 7, $\nabla_\eta \theta^k$ further expands as

$$\nabla_\eta \theta^k = \nabla_\eta \theta^0 + \sum_{i=0}^{k-1} \Big(\nabla_\eta \Psi(\eta, \theta^i, \mathcal{D}^i) + \nabla_\eta \theta^i \nabla_{\theta^i} \Psi(\eta, \theta^i, \mathcal{D}^i) \Big), \tag{9}$$

where $\nabla_\eta \theta^0$ can be non-zero when the initial parameters are being meta-learned, as is the case for MAML.

The $\nabla_\eta \theta^k \nabla_{\theta^k} \log p(\tau|\theta^k)$ terms in equation 8 are the product of the derivative of the kth policy parameter with respect to the meta-parameters and the standard policy gradient. The terms of the sum $\sum_{j=0}^{k-1} \nabla_\eta \theta^j \nabla_{\theta^j} \log p(\mathcal{D}^j|\theta^j)$ give the sampling correction that assigns credit from the experience collected with the updated policy directly to the earlier policies. These latter terms are missing from the original MAML derivation (Finn et al., 2017) as shown by Al-Shedivat et al. (2017). They are also omitted in most other meta-gradient algorithms (Xu et al., 2018; Oh et al., 2020; Zheng et al., 2018; Flennerhag et al., 2021).

4.1 Variance due to the sampling correction

The sampling correction terms are of the form $\nabla_\eta \theta^j \nabla_{\theta^j} \log p(\mathcal{D}^j|\theta^j) R(\tau)$, where $\log p(\mathcal{D}^j|\theta^j) = \sum_{\tau \in \mathcal{D}^j} \log p(\tau|\theta^j)$. The sum over the batch elements results from the fact that every trajectory in the batch is upstream of the reward in the unrolled computation graph. In normal RL, only the samples from a given trajectory are upstream of a particular reward. Nonetheless, a discount factor is typically used to downweight the credit assigned to temporally distant actions, reducing variance but introducing bias with respect to the undiscounted objective. Similarly, we may downweight the sampling correction terms to trade off bias and variance in meta-gradient meta-RL. We find that this is important in practice due to the counterintuitive growth of the variance of these terms with the inner-loop batch size. Note that the variance of the meta-gradient estimate will of course still be reduced by increasing the outer-loop's meta-batch size.

To enable this bias-variance tradeoff for the sampling correction terms, we use a coefficient λ. The meta-gradient estimator estimated from samples is given by

$$J_K(\eta) \approx \sum_{k=0}^{K} \frac{1}{|\mathcal{D}|} \left(\lambda \sum_{j=0}^{k-1} \nabla_\eta \theta^j \nabla_{\theta^j} \log p(\mathcal{D}^j|\theta^j) \sum_{\tau \in \mathcal{D}^k} R(\tau) + \sum_{\tau \in \mathcal{D}^k} \nabla_\eta \theta^k \nabla_{\theta^k} \log p(\tau|\theta^k) R(\tau) \right). \tag{10}$$

The sampling correction coefficient is related to the λ hyperparameter of E-MAML (Stadie et al., 2018), but we do not separately divide by the batch size, which is done in implementations of E-MAML though not motivated in the paper. Therefore, the meta-gradient estimator is unbiased when our $\lambda = 1.0$. We investigate the bias-variance tradeoff induced by this λ in Section 6.

4.2 Meta-gradient estimators for the expected policy gradient

An expression for the meta-gradient without the sampling correction, used by most meta-gradient algorithms, can be derived by substituting the stochastic policy gradient in the update function with the expected policy gradient given by equation 3. In that case, there is no dependency between the policy θ^k and the data sampled with earlier policies because when the policy is updated using the expected policy gradient, θ^k becomes a deterministic function of η and θ^0. Therefore, we no longer get the sampling correction terms that consider how η affects the distribution of data. The meta-gradient for the expected update case is given by

$$\nabla \eta J'_K(\eta) = \nabla_\eta \sum_{k=0}^{K} \mathbb{E}_{\tau \sim p(\tau;\theta^k)} \Big[R(\tau) \Big] = \sum_{k=0}^{K} \mathbb{E}_{\tau \sim p(\tau;\theta^k)} \Big[\nabla_\eta \theta^k \nabla_{\theta^k} \log p(\tau;\theta^k) R(\tau) \Big]. \tag{11}$$

Algorithms that estimate this by computing inner-loop updates and the meta-gradient from samples are biased because they ignore the sampling correction terms.

5 Estimators for the expected policy Hessian

Often, computing an unbiased meta-gradient estimator requires computing second order derivatives of the inner loop objective. As pointed out by Foerster et al. (2018), computing the second order derivatives of an expected policy gradient objective is a more involved process than the corresponding computation for a sampled policy gradient objective. In this section, we look at estimators for second order derivatives of policy gradient objectives.

Assuming that the update function uses some form of policy gradient, the term $\nabla_{\theta^k} \Psi(\eta, \theta^k, \mathcal{D}^k)$ in equation 9 includes the Hessian of the policy gradient objective. Using the stochastic policy gradient in the update function makes it a deterministic function of its inputs, for which the correct Hessian is easy to compute with reverse mode automatic differentiation offered by machine learning packages (e.g., (Paszke et al., 2019; Bradbury et al., 2018)) by backpropagating through the computation graph resulting from the sequence of updates given by equation 7. However, assuming (typically incorrectly) that the inner update uses the expected policy gradient, computing an unbiased estimator of the meta-gradient requires more work. Developing unbiased estimators of the Hessian of the expected policy gradient has been a popular research direction within meta-RL (Rothfuss et al., 2018; Liu et al., 2019; Farquhar et al., 2019; Mao et al., 2019; Tang et al., 2021). As we show next, inclusion of the Hessian of the expected policy gradient in the meta-gradient estimator is an involved process with a questionable payoff as it leads to bias in practice and requires changes to how the update function itself is computed.

The expected Hessian of the policy gradient objective in equation 2 can be derived as follows

$$\nabla_\theta^2 J(\theta) = \nabla_\theta \mathbb{E}_{\tau\sim p(\tau;\theta)}[\nabla_\theta \log \pi(\tau;\theta)\mathcal{R}(\tau)] = \nabla_\theta \int p(\tau;\theta)\nabla_\theta \log\pi(\tau;\theta)\mathcal{R}(\tau)d\tau \tag{12}$$

$$= \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[\left(\nabla_\theta \log\pi(\tau;\theta)\nabla_\theta\log\pi(\tau;\theta)^\top + \nabla_\theta^2\log\pi(\tau;\theta)\right)\mathcal{R}(\tau)\right]. \tag{13}$$

Estimating the expectation from samples we are left with the two terms

$$\nabla_\theta^2 J(\theta) \approx \frac{1}{|\mathcal{D}|}\sum_{\tau\in\mathcal{D}}\left(\nabla_\theta \log\pi(\tau;\theta)\nabla_\theta\log\pi(\tau;\theta)^\top + \nabla_\theta^2\log\pi(\tau;\theta)\right)\mathcal{R}(\tau). \tag{14}$$

In contrast, when the update function uses the stochastic policy gradient estimate given by 4, the Hessian is the following

$$\nabla_\theta^2 J(\theta,\mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{\tau\in\mathcal{D}}\nabla_\theta^2\log\pi(\tau;\theta)\mathcal{R}(\tau). \tag{15}$$

The term $\nabla_\theta \log\pi(\tau;\theta)\nabla_\theta\log\pi(\tau;\theta)^\top\mathcal{R}(\tau) \in \mathbb{R}^{d_\theta\times d_\theta}$ does not appear in this stochastic Hessian. Although including estimates of the expected Hessian is motivated by removing bias, in the typical case where the update function uses a stochastic policy gradient, it actually adds bias because the true stochastic Hessian does not have the extra term. It is also known to have high variance (Rothfuss et al., 2018). Consequently, we argue that developing meta-gradient algorithms based on them is not the best use of resources. We will illustrate the bias and variance resulting from using this estimator in practice in Section 7.

6 Sampling corrections in a truncated optimization setting

Meta-gradients can be used in practice for optimizing meta-parameters of large scale deep RL algorithms. In these algorithms, the policy learning may take tens of thousands of update steps to converge. Unfortunately, due to memory constraints, constructing a computation graph containing the data and parameters quickly becomes infeasible as the update horizon grows. To reduce the memory requirement it is common to truncate the update horizon, which introduces bias. This approach has been employed successfully in many practical meta-learning algorithms, e.g., (Xu et al., 2018; Zahavy et al., 2020; Oh et al., 2020). We know that an unbiased meta-gradient estimator can be computed using the sampling correction in the untruncated setting. To the best of our knowledge none of the algorithms using truncated backpropagation employ any form of the sampling correction, and it is not known how the sampling correction works in the truncated setting.

The truncated meta-gradient estimator is commonly implemented by approximating the estimator in equation 10 by sampling a window shorter than K update steps and limiting backpropagation to that window. Truncating the backpropagation changes the derivatives of the meta-parameters in equation 9, making this estimator biased. Since the sampling correction terms are a product of the derivative of the meta-parameters in equation 9 and the gradient of the log probability of a batch, they also become biased when the estimate of the derivative of the meta-parameters is biased. Because of the complex dynamics of the optimization process, we have no reason to believe that the bias from truncating the backpropagation decreases monotonically with the truncation length. Therefore, it is possible that using the sampling correction on a truncated meta-gradient estimator increases rather than decreases the bias.

In addition to changing the derivatives, approximation of the sum of all of the terms in equation 10 with a sum of the terms in a shorter window causes bias by modifying the surrogate loss even before backpropagation. If the windows are sampled i.i.d. during the agent's lifetime, the bias from this effect changes monotonically with the sampling correction coefficient λ, although it may increase or decrease depending on the problem setting if the derivatives are also truncated. If the i.i.d. assumption is violated, the bias could change non-monotonically with λ. We investigate the effect of sampling correction on the truncated estimators empirically in Section 7.

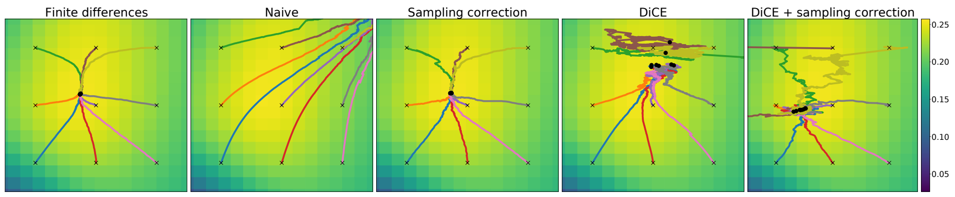

Figure 1: Comparing meta-gradient estimators in the bandit setting. In all figures, on the axes are the meta-parameters corresponding to the learning rate at different stages of the bandit lifetime, the background shows the average return with a meta-parameter value sampled at the center of the cell, the crosses show the nine initial meta-parameter values, and the lines terminating in circles show the trajectories the parameters take during meta-training.

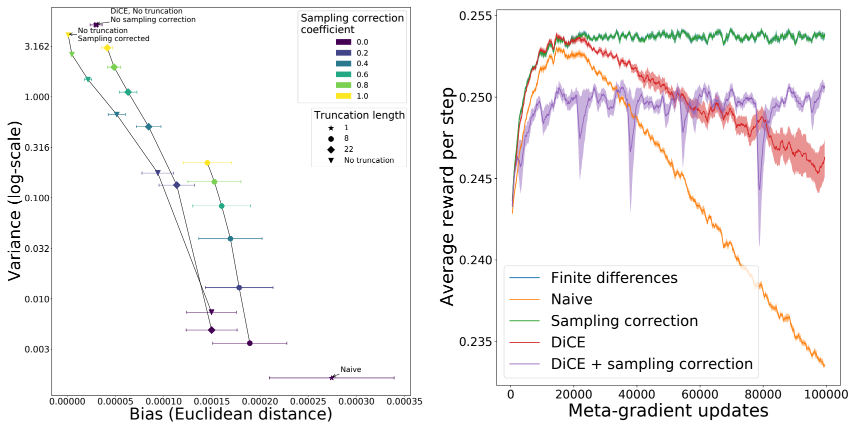

Figure 2: The panel on the left shows the bias and variance estimates for different meta-gradient estimators in the bandit setting averaged over multiple meta-parameter initializations sampled near the local optimum. The error bars show the standard deviation of the bias when computed across bootstrap samples of the initializations. The panel on the right shows the learning curves for the meta-gradient estimators in the bandit setting. The shading of the learning curves shows the standard error across random seeds.

The sampling correction coefficient λ was introduced to trade off bias and variance, but it is only one possible scheme. We consider an alternative weighting using exponential discounting motivated by analogy to the usual discounting procedure in RL. The exponentially discounted meta-gradient estimator is expressed as follows

$$\sum_{k=0}^{K} \frac{1}{|\mathcal{D}|} \left(\sum_{j=0}^{k-1} \alpha^{k-j} \nabla_\eta \theta^j \nabla_{\theta^j} \log p(\mathcal{D}^j|\theta^j) \sum_{\tau \in \mathcal{D}^k} R(\tau) + \sum_{\tau \in \mathcal{D}^k} \nabla_\eta \theta^k \nabla_{\theta^k} \log p(\tau|\theta^k) R(\tau) \right), \quad (16)$$

where α is a meta-discount factor. Analogously to discounting in RL, the weights on the sampling correction terms become exponentially smaller the further back along the update trajectory they are from the return at k. Unlike in standard RL, the discount on the sampling correction term uniformly weights all of the terms in a batch, because all the data in the batch arrives to the update $\Psi(\eta, \theta, \mathcal{D})$ simultaneously. We will compare this variance reduction scheme with the uniform weighting in Section 7.

7 Experiments

We conduct experiments in a bandit setting to investigate the bias and variance of the different estimators and how they affect meta-learning performance. In the experiments, agents are trained with REINFORCE on bandits with randomly sampled arm rewards. Each agent collects 30 batches of experience during its lifetime and is updated on each batch except the last one, which is used only to compute the meta-gradient. The meta-learning problem is to set a learning rate parameter for the first eight updates and another for the remaining updates such that the total reward achieved by the agent during its lifetime is maximized. The meta-parameters are optimized over multiple parallel lifetimes and updated after the lifetimes have concluded. After the lifetimes have concluded, the agent is reset and new lifetimes are sampled. Intuitively, the problem is to meta-learn an optimal trade-off of exploration and exploitation for the given distribution of bandit tasks. Implementation details and hyperparameters can be found in the appendix B.

7.1 Sampling correction and DiCE in practice

In this experiment, we investigate how the different meta-gradient estimators perform in an untruncated setting. In figure 1, we compare five different meta-gradient estimators in the bandit setting by training the learning rate parameters starting from nine initializations. Meta-learning curves in the same setting are shown in figure 2. In the meta-learning curves, all algorithms start from the same initialization. The finite differences gradient estimator converges to the local optimum. The naive estimator does not use the sampling correction, which leads to bias, causing the meta-parameters move away from the local optimum. The sampling corrected gradient estimator is unbiased, and converges to the same local optimum as the finite differences estimator. The gradients of the sampling corrected estimator are similar enough to the finite differences gradient in this setting that their respective learning curves are almost identical. We use DiCE (Foerster et al., 2018) to represent the category of meta-gradient estimators which use an estimate of the expected Hessian in the meta-gradient. Since the bandit problem does not have a time dimension, some of the methods developed for reducing variance of the expected Hessian estimator, including the low variance curvature estimator (Rothfuss et al., 2018) and Loaded DiCE (Farquhar et al., 2019), simply reduce to DiCE. Using DiCE does not stop the meta-optimization from escaping the local optimum, demonstrating that the DiCE meta-gradient is not unbiased by itself. Using DiCE with the sampling corrected meta-gradient estimator converges to a worse local optimum than without DiCE, demonstrating that using the expected Hessian estimator still leads to bias. Additionally, gradient estimation using DiCE suffers from high variance even in the absence of a time dimension.

7.2 Bias-variance tradeoff of the truncated estimators

We investigate the bias-variance tradeoffs possible for truncated sampling corrected estimators. In this experiment, the truncated meta-gradient estimators are computed by randomly sampling a window of one or a few updates during the lifetime and restricting the backpropagation along the update trajectory to that window. The objective of the agent is the same as before, i.e., the average lifetime return, which is approximated with the average return within the truncation window. There are two caveats to this experiment setting as an approximation of the truncated optimization in a long-horizon setting. One is that, due to the strategy of sampling a fixed window length during the lifetime, the number of times each update index appears in the window is not uniform. These edge effects result in bias in the truncated estimators that would be negligible in a longer-horizon setting. The other is that in a larger-scale long-horizon setting, it may be impractical to sample the truncation windows i.i.d. without updating η, which could change how the bias behaves.

In figure 2, we show the bias and variance of estimators with multiple truncation window lengths and sampling correction coefficients. The bias is estimated by computing the average Euclidean distance to the true meta-gradient (computed to high precision with finite differences) at 25 points around the local optimum of the problem shown in figure 1. The error bars show the standard deviation of the bias estimated with 10k bootstrap samples from the 25 meta-parameter points, to reflect how much the bias varies across the meta-optimisation landscape. We know the variance increases with both the sampling correction coefficient and truncation length, so we omit the error bars for variance for clarity. Because the sampling correction coefficient and truncation window length change the meta-gradient magnitude, while its direction may be more important for meta-optimisation in practice, we also present the same data in figure 5 using cosine similarity as the bias measurement.

Effect of truncated backpropagation. Figure 2 shows that the general trend is for the bias to decrease with increased truncation length. However, for the zero sampling correction coefficient the standard deviation of the bias overlaps the mean between multiple truncation lengths suggesting that increasing the truncation length may sometimes increase bias.

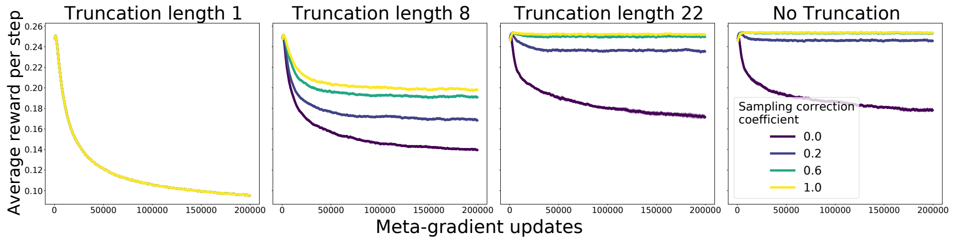

Figure 3: Meta-learning curves for the meta-gradient estimators in the bandit setting with truncated meta-optimization horizons. The shading of the curves shows the standard error across seeds.

Effect of sampling correction with the truncated estimator. The monotonic change in the bias due to the sampling correction coefficient discussed in Section 6 is confirmed by the ordering of the mean bias for each truncation length, i.e., even if the bias increases for a subset of the sampling points and decreases for another subset, the mean should always be either increasing or decreasing with the coefficient for a monotonic bias. We find that for some of the points in the set of 25 meta-parameter values, the bias grows with the sampling correction coefficient. The bias getting larger or smaller depending on the sampling point is reflected in the large standard deviation of the bias of the truncated estimators.

Variance of the estimators. The variance of the meta-gradient estimators grow rapidly with the sampling correction coefficient. Especially the variance of the unbiased estimator is multiple orders of magnitude higher than that of the naive estimator. Part of this is explained by the naive estimator having an order of magnitude smaller norm than the true gradient. In figure 5, we show that the exponential discounting of the sampling correction terms results in a similar bias-variance tradeoff to the simpler sampling correction coefficient. Therefore, at least in this bandit setting, exponential discounting does not seem to have an advantage over using the coefficient.

7.3 Meta-learning using sampling corrected truncated estimators

In the previous experiment, we found that the bias of the truncated meta-gradient estimator may increase with the sampling correction coefficient under some conditions. To investigate how using the sampling correction with the truncated estimator works in practice, we train the meta-parameters of the bandits with the different estimators. We train the meta-parameters for 200k steps using the Adam optimizer (Kingma & Ba, 2014). The meta-learning curves shown in figure 3 suggest that the higher sampling correction values asymptote to better local optima. Therefore, even though the sampling correction may sometimes increase bias of the truncated estimators it can still lead to better meta-learning performance in practice. We confirm that using Adam does not change the ordering of the results by repeating the experiment with SGD, with results shown in figure 4.

8 Conclusion

In this paper, we investigated the estimation of meta-gradients in meta-RL. We showed that, unlike claims in prior work, using an estimator of the expected Hessian in the meta-gradient estimator always adds bias, and is likely to also increase variance. Further, we discussed the sampling correction needed for unbiased meta-gradient estimates, and described how it may be employed in the truncated optimization setting. In doing so, we explored the space of bias-variance tradeoffs that can be made in meta-gradient estimation. We found that applying the sampling correction to truncated meta-gradients leads to convergence to better local optima in our setting, although it increases variance and does not decrease the bias of the truncated gradient estimator in all cases. While these findings can guide researchers to optimize the bias-variance tradeoff for their setting, work remains to be done in extending the investigation to the full RL setting with a time dimension in the inner loop, and developing better meta-gradient estimators that reduce both the bias and variance of the naive estimator.

Acknowledgments and disclosure of funding

Risto Vuorio is supported by EPSRC Doctoral Training Partnership Scholarship and Department of Computer Science Scholarship. Jakob Beck is supported by the Oxford-Google DeepMind Doctoral Scholarship. This work was supported by a generous equipment grant from NVIDIA.

References

Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*, 2017.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pp. 3981–3989, 2016.

Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Yutian Chen, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Timothy P Lillicrap, and Nando de Freitas. Learning to learn for global optimization of black box functions. *arXiv preprint arXiv:1611.03824*, 2016.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pp. 1407–1416. PMLR, 2018.

Alireza Fallah, Kristian Georgiev, Aryan Mokhtari, and Asuman Ozdaglar. Provably convergent policy gradient methods for model-agnostic meta-reinforcement learning. *arXiv preprint arXiv:2002.05135*, 2020.

Gregory Farquhar, Shimon Whiteson, and Jakob Foerster. Loaded dice: Trading off bias and variance in any-order score function gradient estimators for reinforcement learning. 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

Sebastian Flennerhag, Yannick Schroecker, Tom Zahavy, Hado van Hasselt, David Silver, and Satinder Singh. Bootstrapped meta-learning. *arXiv preprint arXiv:2109.04504*, 2021.

Jakob Foerster, Gregory Farquhar, Maruan Al-Shedivat, Tim Rocktäschel, Eric Xing, and Shimon Whiteson. Dice: The infinitely differentiable monte carlo estimator. In *International Conference on Machine Learning*, pp. 1529–1538. PMLR, 2018.

Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning*, pp. 1165–1173. PMLR, 2017.

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

Louis Kirsch, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Improving generalization in meta reinforcement learning using learned objectives. *arXiv preprint arXiv:1910.04098*, 2019.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Hao Liu, Richard Socher, and Caiming Xiong. Taming maml: Efficient unbiased meta-reinforcement learning. In *International Conference on Machine Learning*, pp. 4061–4071. PMLR, 2019.

Jingkai Mao, Jakob Foerster, Tim Rocktäschel, Maruan Al-Shedivat, Gregory Farquhar, and Shimon Whiteson. A baseline for any order gradient estimation in stochastic computation graphs. In *International Conference on Machine Learning*, pp. 4343–4351. PMLR, 2019.

Luke Metz, Niru Maheswaranathan, Jeremy Nixon, Daniel Freeman, and Jascha Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. In *International Conference on Machine Learning*, pp. 4556–4565. PMLR, 2019.

Junhyuk Oh, Matteo Hessel, Wojciech M Czarnecki, Zhongwen Xu, Hado van Hasselt, Satinder Singh, and David Silver. Discovering reinforcement learning algorithms. *arXiv preprint arXiv:2007.08794*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018.

Bradly C Stadie, Ge Yang, Rein Houthooft, Xi Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

Yunhao Tang, Tadashi Kozuno, Mark Rowland, Rémi Munos, and Michal Valko. Unifying gradient estimators for meta-reinforcement learning via off-policy evaluation. *arXiv preprint arXiv:2106.13125*, 2021.

Vivek Veeriah, Tom Zahavy, Matteo Hessel, Zhongwen Xu, Junhyuk Oh, Iurii Kemaev, Hado van Hasselt, David Silver, and Satinder Singh. Discovery of options via meta-learned subgoals. *arXiv preprint arXiv:2102.06741*, 2021.

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

Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization. *arXiv preprint arXiv:1803.02021*, 2018.

Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-gradient reinforcement learning. *arXiv preprint arXiv:1805.09801*, 2018.

Zhongwen Xu, Hado van Hasselt, Matteo Hessel, Junhyuk Oh, Satinder Singh, and David Silver. Meta-gradient reinforcement learning with an objective discovered online. *arXiv preprint arXiv:2007.08433*, 2020.

Tom Zahavy, Zhongwen Xu, Vivek Veeriah, Matteo Hessel, Junhyuk Oh, Hado van Hasselt, David Silver, and Satinder Singh. A self-tuning actor-critic algorithm. *arXiv preprint arXiv:2002.12928*, 2020.

Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods. *arXiv preprint arXiv:1804.06459*, 2018.

Zeyu Zheng, Junhyuk Oh, Matteo Hessel, Zhongwen Xu, Manuel Kroiss, Hado Van Hasselt, David Silver, and Satinder Singh. What can learned intrinsic rewards capture? In *International Conference on Machine Learning*, pp. 11436–11446. PMLR, 2020.

A Derivation of the unbiased meta-gradient

More detailed derivation of the unbiased meta-gradient estimator is given by

$$\begin{aligned}
\nabla_\eta J_K(\eta) &= \nabla_\eta \sum_{k=0}^{K} \mathbb{E}_{\{\mathcal{D}^i \sim p(\mathcal{D}^i|\theta^i)\}_{i=0}^{k-1}} \left[\mathbb{E}_{\tau \sim p(\tau|\theta^k)} \left[\mathcal{R}(\tau) \right] \right] \\
&= \nabla_\eta \sum_{k=0}^{K} \int R(\tau) p(\tau|\theta^k) \prod_{i=0}^{k-1} p(\mathcal{D}^i|\theta^i) d\mathcal{D}^i d\tau \qquad (17) \\
&= \sum_{k=0}^{K} \int R(\tau) \Bigg(\nabla_\eta \theta^k \nabla_{\theta^k} \log p(\tau|\theta^k) \\
&\qquad + \sum_{j=0}^{k-1} \nabla_\eta \theta^j \nabla_{\theta^j} \log p(\mathcal{D}^j|\theta^j) \Bigg) p(\tau|\theta^k) \prod_{i=0}^{k-1} p(\mathcal{D}^i|\theta^i) d\mathcal{D}^j d\tau \qquad (18) \\
&= \sum_{k=0}^{K} \mathbb{E}_{\substack{\{\mathcal{D}^i\}_{i=0}^{k-1} \\ \tau \sim p(\tau|\theta^k)}} \left[\Bigg(\underbrace{\sum_{j=0}^{k-1} \nabla_\eta \theta^j \nabla_{\theta^j} \log p(\mathcal{D}^j|\theta^j)}_{\texttt{sampling correction}} + \underbrace{\nabla_\eta \theta^k \nabla_{\theta^k} \log p(\tau|\theta^k)}_{\texttt{direct meta-gradient}} \Bigg) R(\tau) \right]. \qquad (19)
\end{aligned}$$

B Bandit settings

In this section we describe the details of the bandit experiments. The experiments are conducted in a multi-armed bandit setting. The bandits have 30 arms with the arm means sampled from $exp(Uniform(-100, 1))$. The reward for each pull is computed by adding noise sampled from Gaussian with standard deviation 2 to the arm mean. The policy is parametrized as a softmax over a randomly initialized vector of logits. The inner learning problem is learning the policy for the bandit and the outer problem is to learn a learning rate schedule for the inner learner. The inner loop uses a simple policy gradient algorithm (Williams, 1992). The inner loop updates the policy for 29 update steps sampling 30 batches of experience in total. Each update in the inner loop is computed on ten samples from the bandit. The outer loop updates the learning rate schedule with the meta-gradient computed by sampling from an expression similar to 8. The outer loop objective used in practice considers the return in each batch sampled from the bandit instead of the last one alone. When truncation is applied, the inner loop is run as before but the outer loop only considers returns within the truncation window and only backpropagates gradients within the window. The meta-parameter is a two-dimensional array, from which the learning rate is computed by choosing one of the two values according to the index of the update on the inner loop lifetime and applying the softplus function. The first 8 updates correspond to the first dimension of the meta-parameter and the rest to the second. The meta-gradient is computed across multiple inner learning problems in parallel. For computing one meta-gradient update with the untruncated inner loop, inner batch size $\times$ lifetime length $\times$ parallel runs samples from the bandit are used. All the updates are computed with stochastic gradient descent. The hyperparameters of the experiments in each figure are summarized in 1.

C Additional experimental results

The meta-gradient estimators are compared in terms of cosine similarity in the left panel of figure 5. The two weighting schemes for the sampling correction are compared in the right panel of figure 5. Learning curves for the truncated meta-gradient estimators using SGD as the optimizer are shown in figure 4. The large error in the curves for the higher truncation lengths are due to some of the seeds

	Hyperparameter	Value
Shared hyperparameters	inner batch size	10
	lifetime length	30
	optimizer	SGD
Figure 1	parallel runs	1000
	outer loop updates	100000
Figure 2 right panel	parallel runs	1000
	outer loop updates	100000
	outer learning rate	0.01
	number of random seeds	10
Figure 3	parallel runs	1000
	outer loop updates	200000
	outer learning rate	0.001
	number of random seeds	5
	optimizer	ADAM
Figure 4, truncation lengths 1 and 8	parallel runs	1000
	outer loop updates	500000
	outer learning rate	0.05
	number of random seeds	5
Figure 4, truncation lengths 22 and 29	parallel runs	1000
	outer loop updates	500000
	outer learning rate	0.005
	number of random seeds	5
Figure 2 left panel, and figure 5	parallel runs	2000
	outer loop samples	1000

Table 1: Bandit hyperparameters for each experiment

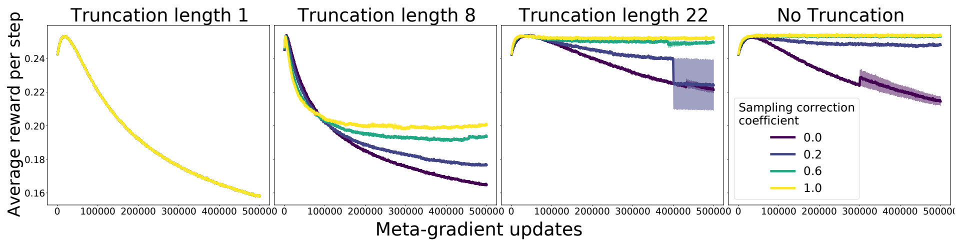

Figure 4: Learning curves for the meta-gradient estimators in the bandit setting with truncated meta-optimization horizons using SGD as the optimizer. The shading of the curves shows the standard error across random seeds.

becoming unstable during training and deviating far from the optimal region in a single gradient step. This effect can be reduced in practice by using adaptive optimizers such as Adam and gradient clipping.

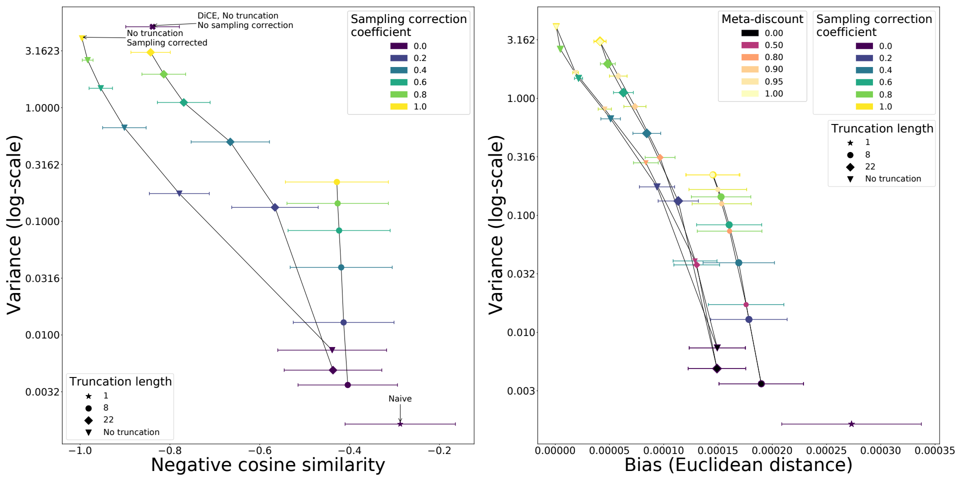

Figure 5: On the left, the bias and variance of meta-gradient estimators in the bandit setting with negative cosine similarity estimate of the bias. Negative cosine similarity is shown instead of cosine similarity to keep the bias axis consistent with figure 2, that is, bias grows toward the right edge of the figure. On the right, the bias and variance of two different weighting schemes for the sampling correction terms are explored.