Predictor-Corrector Policy Optimization

Ching-An Cheng [1 2] Xinyan Yan [1] Nathan Ratliff [2] Byron Boots [1 2]

Abstract

We present a predictor-corrector framework, called PicCoLO, that can transform a first-order model-free reinforcement or imitation learning algorithm into a new hybrid method that leverages predictive models to accelerate policy learning. The new "PicCoLOed" algorithm optimizes a policy by recursively repeating two steps: In the Prediction Step, the learner uses a model to predict the unseen future gradient and then applies the predicted estimate to update the policy; in the Correction Step, the learner runs the updated policy in the environment, receives the true gradient, and then corrects the policy using the gradient error. Unlike previous algorithms, PicCoLO corrects for the mistakes of using imperfect predicted gradients and hence does not suffer from model bias. The development of PicCoLO is made possible by a novel reduction from predictable online learning to adversarial online learning, which provides a systematic way to modify existing first-order algorithms to achieve the optimal regret with respect to predictable information. We show, in both theory and simulation, that the convergence rate of several first-order model-free algorithms can be improved by PicCoLO.

1. Introduction

Reinforcement learning (RL) has recently solved a number of challenging problems (Mnih et al., 2013; Duan et al., 2016; Silver et al., 2018). However, many of these successes are confined to games and simulated environments, where a large number of agent-environment interactions can be cheaply performed. Therefore, they are often unrealistic in real-word applications (like robotics) where data collection is an expensive and time-consuming process. Improving sample efficiency still remains a critical challenge for RL.

[1]Georgia Tech [2]NVIDIA. Correspondence to: Ching-An Cheng <cacheng@gatech.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

Model-based RL methods improve sample efficiency by leveraging an accurate model that can cheaply simulate interactions to compute policy updates in lieu of real-world interactions (Tan et al., 2018). A classical example of pure model-based methods is optimal control (Jacobson & Mayne, 1970; Todorov & Li, 2005; Deisenroth & Rasmussen, 2011; Pan & Theodorou, 2014), which has recently been extended to model abstract latent dynamics with neural networks (Silver et al., 2017; Oh et al., 2017). These methods use a (local) model of the dynamics and cost functions to predict cost-to-go functions, policy gradients, or promising improvement direction when updating policies (Levine & Koltun, 2013; Sun et al., 2018; Anthony et al., 2017). Another way to use model information is the hybrid Dyna framework (Sutton, 1991; Sutton et al., 2012), which interleaves model-based and model-free updates, ideally cutting learning time in half. However, all of these approaches, while potentially accelerating policy learning, suffer from a common drawback: when the model is inaccurate, the performance of the policy can become *biased* away from the best achievable in the policy class.

Several strategies have been proposed to remove this performance bias. Learning-to-plan attempts to train the planning process end-to-end (Pascanu et al., 2017; Srinivas et al., 2018; Amos et al., 2018), so the performance of a given planning structure is directly optimized. However, these algorithms are still optimized through standard model-free RL techniques; it is unclear as to whether they are more sample efficient. In parallel, another class of bias-free algorithms is control variate methods (Chebotar et al., 2017; Grathwohl et al., 2018; Papini et al., 2018), which use models to reduce the variance of sampled gradients to improve convergence.

In this paper, we provide a novel learning framework that can leverage models to improve sample efficiency while avoiding performance bias due to modeling errors. Our approach is built on techniques from online learning (Gordon, 1999; Zinkevich, 2003). The use of online learning to analyze policy optimization was pioneered by Ross et al. (2011), who proposed to reduce imitation learning (IL) to adversarial online learning problems. This reduction provides a framework for performance analysis, leading to algorithms such as DAgger (Ross et al., 2011) and AggreVaTe (Ross & Bagnell, 2014). However, it was recently shown that the naïve reduction to adversarial online learning loses information (Cheng & Boots, 2018): in practice, IL is *predictable* (Cheng et al., 2019) and can be thought of as a predictable online learning problem (Rakhlin & Sridharan, 2013a). Based on this insight, Cheng et al. (2019) recently proposed a two-step algorithm, MoBIL. The authors prove that, by leveraging predictive models to estimate future gradients, MoBIL can speed up the convergence of IL, without incurring performance bias due to imperfect models.

Given these theoretical advances in IL, it is natural to ask if similar ideas can be extended to RL. In this paper, we show that RL can also be formulated as a predictable online learning problem, and we propose a novel first-order learning framework, PicCoLO (PredICtor-COrrector poLicy Optimization), for general predictable online learning problems. PicCoLO is a *meta-algorithm*: it takes a standard online learning algorithm designed for adversarial problems (e.g. ADAGRAD (Duchi et al., 2011)) as input and returns a new hybrid algorithm that can use model information to accelerate convergence. This new "PicCoLOed" algorithm optimizes the policy by alternating between Prediction and Correction steps. In the Prediction Step, the learner uses a predictive model to estimate the gradient of the next loss function and then uses it to update the policy; in the Correction Step, the learner executes the updated policy in the environment, receives the true gradient , and then corrects the policy using the gradient *error*. We note that PicCoLO is orthogonal to control variate methods; it can still improve learning even in the noise-free setting (see Section 5.2).

Theoretically, we prove that PicCoLO can improve the convergence rate of *any* base algorithm that can be written as mirror descent (Beck & Teboulle, 2003) or Follow-the-Regularized-Leader (FTRL) (McMahan & Streeter, 2010). This family of algorithms is rich and covers most first-order algorithms used in RL and IL (Cheng et al., 2018). And, importantly, we show that PicCoLO does not suffer from performance bias due to model error, unlike previous model-based approaches. To validate the theory, we "PicCoLO" multiple algorithms in simulation. The experimental results show that the PicCoLOed versions consistently surpass the base algorithm and are robust to model errors.

The design of PicCoLO is made possible by a novel reduction that converts a given predictable online learning problem into a new adversarial problem, so that standard online learning algorithms can be applied optimally without referring to specialized algorithms. We show that PicCoLO includes and generalizes many existing algorithms, e.g., MoBIL, mirror-prox (Juditsky et al., 2011), and optimistic mirror descent (Rakhlin & Sridharan, 2013a) (Appendix A). Thus, we can treat PicCoLO as an automatic process for designing new algorithms that safely leverages imperfect predictive models (such as off-policy gradients or gradients simulated through dynamics models) to speed up learning.

2. Problem Definition

We consider solving policy optimization problems: given state and action spaces $\mathbb{S}$ and $\mathbb{A}$, and a parametric policy class Π, we desire a stationary policy $\pi \in \Pi$ that solves

$$\min_{\pi \in \Pi} J(\pi), \quad J(\pi) \coloneqq \mathbb{E}_{(s,t)\sim d_\pi} \mathbb{E}_{a\sim\pi_s} \left[c_t(s,a)\right] \tag{1}$$

where $c_t(s,a)$ is the instantaneous cost at time t of state $s \in \mathbb{S}$ and $a \in \mathbb{A}$, π_s is the distribution of a at state s under policy π, and d_π is a generalized stationary distribution of states generated by running policy π in a Markov decision process (MDP); the notation $\mathbb{E}_{a\sim\pi_s}$ denotes evaluation when π is deterministic. The use of d_π in (1) abstracts different discrete-time RL/IL problems into a common setup. For example, an infinite-horizon γ-discounted problem with time-invariant cost c can be modeled by setting $c_t = c$ and $d_\pi(s,t) = (1-\gamma)\gamma^t d_{\pi,t}(s)$, where $d_{\pi,t}$ is the state distribution visited by policy π at time t starting from some *fixed* but unknown initial state distribution.

For convenience, we will usually omit the random variable in expectation notation (e.g. we will write (1) as $\mathbb{E}_{d_\pi}\mathbb{E}_\pi\left[c\right]$). For a policy π, we overload the notation π to also denote its parameter, and write $Q_{\pi,t}$ and $V_{\pi,t} \coloneqq \mathbb{E}_\pi[Q_{\pi,t}]$ as its Q-function and value function at time t, respectively.

3. IL and RL as Predictable Online Learning

We study policy optimization through the lens of online learning (Hazan et al., 2016), by treating a policy optimization algorithm as the learner in online learning and *each intermediate policy* that it produces as an online decision. This identification recasts the iterative process of policy optimization into a standard online learning setup: in round n, the learner plays a decision $\pi_n \in \Pi$, a *per-round loss* l_n is then selected, and finally some information of l_n is revealed to the leaner for making the next decision. We note that the "rounds" considered here are the number of episodes that an algorithm interacts with the (unknown) MDP environment to obtain new information, not the time steps in the MDP. And we will suppose the learner receives an unbiased stochastic approximation $\tilde{l}_n$ of l_n as feedback.

We show that, when the per-round losses $\{l_n\}$ are properly selected, the policy performance $\{J(\pi_n)\}$ in IL and RL can be upper bounded in terms the N-round weighted regret

$$\text{Regret}_N(l) \coloneqq \sum_{n=1}^{N} w_n l_n(\pi_n) - \min_{\pi\in\Pi} \sum_{n=1}^{N} w_n l_n(\pi) \tag{2}$$

and an expressiveness measure of the policy class Π

$$\epsilon_{\Pi,N}(l) \coloneqq \tfrac{1}{w_{1:N}} \min_{\pi\in\Pi} \textstyle\sum_{n=1}^N w_n l_n(\pi) \tag{3}$$

where $w_n > 0$ and $w_{1:n} \coloneqq \sum_{m=1}^n w_m$. Moreover, we show that these online learning problems are *predictable*:

that is, the per-round losses are not completely adversarial but can be estimated from past information. We will use these ideas to design PICCOLO in the next section.

3.1. IL as Online Learning

We start by reviewing the classical online learning approach to IL (online IL for short) (Ross et al., 2011) to highlight some key ideas. IL leverages domain knowledge about a policy optimization problem through expert demonstrations. Online IL, in particular, optimizes policies by letting the learner π query the expert $\pi^\star$ for desired actions, so that a policy can be quickly trained to perform as well as the expert. At its heart, online IL is based on the following lemma, which relates the performance between π and $\pi^\star$.

Lemma 1. (Kakade & Langford, 2002) *Let π and π' be two policies and $A_{\pi',t}(s,a) := Q_{\pi',t}(s,a) - V_{\pi',t}(s)$. Then $J(\pi) = J(\pi') + \mathbb{E}_{d_\pi}\mathbb{E}_\pi[A_{\pi'}]$.*

Given the equality in Lemma 1, the performance difference between π and $\pi^\star$ can then be upper-bounded as

$$J(\pi) - J(\pi^\star) = \mathbb{E}_{d_\pi}\mathbb{E}_\pi[A_{\pi^\star}] \leq C_{\pi^\star}\mathbb{E}_{(s,t)\sim d_\pi}[D_t(\pi_s^\star||\pi_s)]$$

for some positive constant $C_{\pi^\star}$ and function D_t, which is often derived from statistical distances such as KL divergence (Cheng et al., 2018). When $A_{\pi^*,t}$ is available, we can also set $D_t(\pi_s^\star||\pi_s) = \mathbb{E}_{a\sim\pi_s}[A_{\pi^\star,t}(s,a)]$, as in value aggregation (AGGREVATE) (Ross & Bagnell, 2014).

Without loss of generality, let us suppose $D_t(\pi_s^\star||\pi_s) = \mathbb{E}_{a\sim\pi_s}[\bar{c}_t(s,a)]$ for some $\bar{c}_t$. Online IL converts policy optimization into online learning with per-round loss

$$l_n(\pi) := \mathbb{E}_{d_{\pi_n}}\mathbb{E}_\pi[\bar{c}]. \tag{4}$$

By the inequality above, it holds that $J(\pi_n) - J(\pi^\star) \leq C_{\pi^\star} l_n(\pi_n)$ for every n, establishing the reduction below.

Lemma 2. (Cheng et al., 2019) *For l_n defined in (4), $\mathbb{E}\left[\sum_{n=1}^N \frac{w_n J(\pi_n)}{w_{1:N}}\right] \leq J(\pi^\star) + C_{\pi^\star}\mathbb{E}\left[\epsilon_{\Pi,N}(l) + \frac{\mathit{Regret}_N(\tilde{l})}{w_{1:N}}\right]$, where the expectation is due to sampling $\tilde{l}_n$.*

That is, when a no-regret algorithm is used, the performance concentrates toward $J(\pi^\star) + C_{\pi^\star}\mathbb{E}[\epsilon_{\Pi,N}(l)]$.

3.2. RL as Online Learning

Can we also formulate RL as online learning? Here we propose a new perspective on RL using Lemma 1. Given a policy π_n in round n, we define a per-round loss

$$l_n(\pi) := \mathbb{E}_{d_{\pi_n}}\mathbb{E}_\pi[A_{\pi_{n-1}}]. \tag{5}$$

which describes how well a policy π performs relative to the previous policy π_{n-1} under the state distribution of π_n. By Lemma 1, for l_n defined in (5), $l_n(\pi_n) = J(\pi_n) - J(\pi_{n-1})$ for every n, similar to the pointwise inequality of l_n that Lemma 2 is based on. With this observation, we derive the reduction below (proved in Appendix B).

Lemma 3. *Suppose $\frac{w_{n+k}}{w_n} \leq \frac{w_{m+k}}{w_m}$, for all $n \geq m \geq 1$ and $k \geq 0$. For (5) and any π_0, $\mathbb{E}[\sum_{n=1}^N \frac{w_n J(\pi_n)}{w_{1:N}}] \leq J(\pi_0) + \sum_{n=1}^N \frac{w_{N-n+1}}{w_{1:N}}\mathbb{E}[\mathit{Regret}_n(\tilde{l}) + w_{1:n}\epsilon_{\Pi,n}(l)]$, where the expectation is due to sampling $\tilde{l}_n$.*

3.2.1. INTERPRETATIONS

Lemma 3 is a policy improvement lemma, which shows that when the learning algorithm is no-regret, the policy sequence improves on-average from the initial reference policy π_0 that defines l_1. This is attributed to an important property of the definition in (5) that $\min_{\pi\in\Pi} l_n(\pi) \leq 0$. To see this, suppose $\mathbb{E}[\epsilon_{\Pi,n}(l)] \leq -\Omega(1)$ (i.e. there is a policy that is better than all previous n policies); this is true for small n or when the policy sequence is concentrated. Under this assumption, if $w_n = 1$ and $\text{Regret}_n(\tilde{l}) \leq O(\sqrt{n})$, then the average performance improves roughly $N\mathbb{E}[\epsilon_{\Pi,N}(l)]$ away from $J(\pi_0)$.

While it is unrealistic to expect $\mathbb{E}[\epsilon_{\Pi,n}(l)] \leq 0$ for large n, we can still use Lemma 3 to comprehend *global* properties of policy improvement, for two reasons. First, the inequality in Lemma 3 holds for any interval of the policy sequence. Second, as we show in Appendix B, the Lemma 3 also applies to dynamic regret (Zinkevich, 2003), with respect to which $\mathbb{E}[\epsilon_{\Pi,n}(l)]$ is always negative. Therefore, if an algorithm is strongly-adaptive (Daniely et al., 2015) (i.e. it is no-regret for any interval) or has sublinear dynamic regret (Jadbabaie et al., 2015), then its generated policy sequence will strictly, non-asymptotically improve. In other words, for algorithms with a stronger notion of convergence, Lemma 3 describes the global improvement rate.

3.2.2. CONNECTIONS

The choice of per-round loss in (5) has an interesting relationship to both actor-critic in RL (Konda & Tsitsiklis, 2000) and AGGREVATE in IL (Ross & Bagnell, 2014).

Relationship to Actor-Critic Although actor-critic methods, theoretically, use $\mathbb{E}_{d_{\pi_n}}(\nabla\mathbb{E}_\pi)[A_{\pi_n}]|_{\pi=\pi_n}$ to update policy π_n, in practice, they use $\mathbb{E}_{d_{\pi_n}}(\nabla\mathbb{E}_\pi)[A_{\pi_{n-1}}]|_{\pi=\pi_n}$, because the advantage/value function estimate in round n is updated after the policy update in order to prevent bias due to over-fitting on finite samples (Sutton & Barto, 1998). This practical gradient is *exactly* $\nabla\tilde{l}_n(\pi_n)$, the sampled gradient of (5). Therefore, Lemma 3 explains the properties of these practical modifications.

Relationship to Value Aggregation AGGREVATE (Ross & Bagnell, 2014) can be viewed as taking a policy improvement step from some reference policy: e.g., with the per-round loss $\mathbb{E}_{d_{\pi_n}}\mathbb{E}_{\pi}[A_{\pi^\star}]$, it improves one step from π^*. Realizing this one step improvement in AGGREVATE, however, requires solving multiple rounds of online learning, as it effectively solves an equilibrium point problem (Cheng & Boots, 2018). Therefore, while ideally one can solve multiple AGGREVATE problems (one for each policy improvement step) to optimize policies, computationally this can be very challenging. Minimizing the loss in (5) can be viewed as an approximate policy improvement step in the AGGREVATE style. Rather than waiting until convergence in each AGGREVATE policy improvement step, it performs only a *single* policy update and then switches to the next AGGREVATE problem with a new reference policy (i.e. the latest policy π_{n-1}). This connection is particularly tightened if we choose $\pi_0 = \pi^\star$ and the bound in Lemma 3 becomes relative to $J(\pi^\star)$.

3.3. Predictability

An important property of the above online learning problems is that they are not completely adversarial, as pointed out by Cheng & Boots (2018) for IL. This can be seen from the definitions of l_n in (4) and (5), respectively. For example, suppose the cost c_t in the original RL problem (1) is known; then the information unknown before playing the decision π_n in the environment is only the state distribution d_{π_n}. Therefore, the per-round loss cannot be truly adversarial, as the same dynamics and cost functions are used across different rounds. That is, in an idealized case where the true dynamics and cost functions are exactly known, using the policy returned from a model-based RL algorithm would incur zero regret, since only the interactions with the real MDP environment, not the model, counts as rounds. We will exploit this property to design PICCOLO.

4. Predictor-Corrector Learning

We showed that the performance of RL and IL can be bounded by the regret of properly constructed predictable online learning problems. These results provide a foundation for designing policy optimization algorithms: efficient learning algorithms for policy optimization can be constructed from powerful online learning algorithms that achieve small regret. This perspective explains why common methods (e.g. mirror descent) based on gradients of (4) and (5) work well in IL and RL. However, the predictable nature of policy optimization problems suggests that directly applying these standard online learning algorithms designed for adversarial settings is *suboptimal*. The predictable information must be considered to achieve optimal convergence.

One way to include predictable information is to develop specialized two-step algorithms based on, e.g., mirror-prox or FTRL-prediction (Juditsky et al., 2011; Rakhlin & Sridharan, 2013a; Ho-Nguyen & Kılınç-Karzan, 2018). For IL, MOBIL was recently proposed (Cheng et al., 2019), which updates policies by approximate Be-the-Leader (Kalai & Vempala, 2005) and provably achieves faster convergence than previous methods. However, these two-step algorithms often have obscure and non-sequential update rules, and their adaptive and accelerated versions are less accessible (Diakonikolas & Orecchia, 2017). This can make it difficult to implement and tune them in practice.

Here we take an alternative, *reduction-based* approach. We present PICCOLO, a general first-order framework for solving predictable online learning problems. PICCOLO is a meta-algorithm that turns a base algorithm designed for adversarial problems into a new algorithm that can leverage the predictable information to achieve better performance. As a result, we can adopt sophisticated first-order adaptive algorithms to optimally learn policies, without reinventing the wheel. Specifically, given *any* first-order base algorithm belonging to the family of (adaptive) mirror descent and FTRL algorithms, we show how one can "PICCOLO it" to achieve a faster convergence rate without introducing additional performance bias due to prediction errors. Most first-order policy optimization algorithms belong to this family (Cheng et al., 2018), so we can PICCOLO these model-free algorithms into new hybrid algorithms that can robustly use (imperfect) predictive models, such as off-policy gradients and simulated gradients, to improve policy learning.

4.1. The PICCOLO Idea

The design of PICCOLO is based on the observation that an N-round predictable online learning problem can be written as a new adversarial problems with $2N$ rounds. To see this, let $\{l_n\}_{n=1}^N$ be the original predictable loss sequence. Suppose, before observing l_n, we have access to a *model loss* $\hat{l}_n$ that contains the predictable information of l_n. Define $\delta_n = l_n - \hat{l}_n$. We can then write the accumulated loss (which regret concerns) as $\sum_{n=1}^N l_n(\pi_n) = \sum_{n=1}^N \hat{l}_n(\pi_n) + \delta_n(\pi_n)$. That is, we can view the predictable problem with $\{l_n\}_{n=1}^N$ as a new adversarial online learning problem with a loss sequence $\hat{l}_1, \delta_1, \hat{l}_2, \delta_2, \dots, \hat{l}_N, \delta_N$.

The idea of PICCOLO is to apply standard online learning algorithms designed for adversarial settings to this new $2N$-round problem. This would create a new set of decision variables $\{\hat{\pi}_n\}_{n=1}^N$, in which $\hat{\pi}_n$ denotes the decision made before seeing $\hat{l}_n$, and leads to the following sequence $\pi_1, \delta_1, \hat{\pi}_2, \hat{l}_2, \pi_2, \delta_2, \dots$ (in which we define $\delta_1 = l_1$). We show that when the base algorithm is optimal in adversarial settings, this simple strategy results in a decision sequence $\{\pi_n\}_{n=1}^N$ whose regret with respect to $\{l_n\}_{n=1}^N$ is optimal, just as those specialized two-step algorithms (Juditsky et al., 2011; Rakhlin & Sridharan, 2013a; Ho-Nguyen & Kılınç-Karzan, 2018). In Appendix A, we show PICCOLO unifies and generalize these two-step algorithms to be adaptive.

4.2. The Meta Algorithm PicCoLO

We provide details to realize this reduction. We suppose, in round n, the model loss is given as $\hat{l}_n(\pi) = \langle \hat{g}_n, \pi \rangle$ for some vector $\hat{g}_n$, and stochastic first-order feedback $g_n = \nabla \tilde{l}_n(\pi_n)$ from l_n is received. Though this linear form of model loss seems restrictive, later in Section 4.2.3 we will show that it is sufficient to represent predictable information.

4.2.1. Base Algorithms

We first give a single description of different base algorithms for the formal definition of the reduction steps. Here we limit our discussions to mirror descent and postpone the FTRL case to Appendix C. We assume that Π is a convex compact subset in some normed space with norm $\|\cdot\|$, and we use $B_R(\pi||\pi') = R(\pi) - R(\pi') - \langle \nabla R(\pi'), \pi - \pi' \rangle$ to denote a Bregman divergence generated by a strictly convex function R, called the distance generator.

Mirror descent updates decisions based on proximal maps. In round n, given direction g_n and weight w_n, it executes

$$\pi_{n+1} = \arg\min_{\pi \in \Pi} \langle w_n g_n, \pi \rangle + B_{R_n}(\pi||\pi_n) \qquad (6)$$

where R_n is a strongly convex function; (6) reduces to gradient descent with step size η_n when $R_n(\cdot) = \frac{1}{2\eta_n}\|\cdot\|^2$. More precisely, (6) is composed of two steps: 1) the update of the distance generator to R_n, and 2) the update of the decision to π_{n+1}; different mirror descent algorithms differ in how the regularization is selected and adapted.

PicCoLO explicitly treats a base algorithm as the composition of two basic operations (this applies also to FTRL)

$$\begin{aligned} H_n &= \texttt{adapt}(h_n, H_{n-1}, g_n, w_n) \\ h_{n+1} &= \texttt{update}(h_n, H_n, g_n, w_n) \end{aligned} \qquad (7)$$

so that later it can recompose them to generate the new algorithm. For generality, we use h and H to denote the abstract representations of the decision variable and the regularization, respectively. In mirror descent, h is exactly the decision variable, H is the distance generator, and we can write $\texttt{update}(h, H, g, w) = \arg\min_{\pi' \in \Pi} \langle wg, \pi' \rangle + B_H(\pi'||h)$. The operation $\texttt{adapt}$ denotes the algorithm-specific scheme for the regularization update (e.g. changing the step size), which in general updates the size of regularization to grow slowly and inversely proportional to the norm of g_n.

4.2.2. The PicCoLOed Algorithm

PicCoLO generates decisions by applying a given base algorithm in (7) to the new problem with losses $\delta_1, \hat{l}_2, \delta_2, \dots$. This is accomplished by recomposing the basic operations in (7) into the Prediction and the Correction Steps:

$$h_n = \texttt{update}(\hat{h}_n, H_{n-1}, \hat{g}_n, w_n) \qquad \text{[Prediction]}$$

$$\begin{aligned} H_n &= \texttt{adapt}(h_n, H_{n-1}, e_n, w_n) \\ \hat{h}_{n+1} &= \texttt{update}(h_n, H_n, e_n, w_n) \end{aligned} \qquad \text{[Correction]}$$

where $\hat{h}_n$ is the abstract representation of $\hat{\pi}_n$, and $e_n = g_n - \hat{g}_n$ is the error direction. We can see that the Prediction and Correction Steps are exactly the update rules resulting from applying (7) to the new adversarial problem, except that only h_n is updated in the Prediction Step, *not* the regularization (i.e. the step size). This asymmetry design is important for achieving optimal regret, because in the end we care only about the regret of $\{\pi_n\}$ on the original loss sequence $\{l_n\}$.

In round n, the "PicCoLOed" algorithm first performs the Prediction Step using $\hat{g}_n$ to generate the learner's decision (i.e. π_n) and runs this new policy in the environment to get the true gradient g_n. Using this feedback, the algorithm performs the Correction Step to amend the bias of using $\hat{g}_n$. This is done by first adapting the regularization to H_n and then updating π_n to $\hat{\pi}_{n+1}$ along the error $e_n = g_n - \hat{g}_n$.

4.2.3. Model Losses and Predictive Models

The Prediction Step of PicCoLO relies on the vector $\hat{g}_n$ to approximate the future gradient g_n. Here we discuss different ways to specify $\hat{g}_n$ based on the concept of predictive models (Cheng et al., 2019). A *predictive model* Φ_n is a first-order oracle such that $\Phi_n(\cdot)$ approximates $\nabla l_n(\cdot)$. In practice, a predictive model can be a simulator with an (online learned) dynamics model (Tan et al., 2018; Deisenroth & Rasmussen, 2011), or a neural network trained to predict the required gradients (Silver et al., 2017; Oh et al., 2017). An even simpler heuristic is to construct predictive models by *off-policy* gradients $\Phi_n(\cdot) = \sum_{m=n-K}^{n-1} \nabla \tilde{l}_m(\cdot)$ where K is the buffer size.

In general, we wish to set $\hat{g}_n$ to be close to g_n, as we will later show in Section 5 that the convergence rate of PicCoLO depends on their distance. However, even when we have perfect predictive models, this is still a non-trivial task. We face a chicken-or-the-egg problem: g_n depends on π_n, which in turn depends on $\hat{g}_n$ from the Prediction Step.

Cheng et al. (2019) show one effective heuristic is to set $\hat{g}_n = \Phi_n(\hat{\pi}_n)$, because we may treat $\hat{\pi}_n$ as an estimate of π_n. However, due to the mismatch between $\hat{\pi}_n$ and π_n, this simple approach has errors even when the predictive model is perfect. To better leverage a given predictive model, we propose to solve for $\hat{g}_n$ and π_n *simultaneously*. That is, we wish to solve a fixed-point problem, finding h_n such that

$$h_n = \texttt{update}(\hat{h}_n, H_{n-1}, \Phi_n(\pi_n(h_n)), w_n) \qquad (8)$$

The exact formulation of the fixed-point problem depends on the class of base algorithms. For mirror descent, it is a variational inequality: find $\pi_n \in \Pi$ such that $\forall \pi \in \Pi$, $\langle \Phi_n(\pi_n) + \nabla R_{n-1}(\pi_n) - \nabla R_{n-1}(\hat{\pi}_n), \pi - \pi_n \rangle \geq 0$. In a special case when $\Phi_n = \nabla f_n$ for some function f_n, the above variational inequality is equivalent to finding a stationary point of the optimization problem $\min_{\pi \in \Pi} f_n(\pi) + B_{R_{n-1}}(\pi||\hat{\pi}_n)$. In other words, one way to implement the

Algorithm 1 PICCOLO

```
Input: policy $\pi_1$, cost sequence $\{\psi_n\}$, regularization $H_0$, model $\Phi_1$, iteration $N$, exponent $p$
Output: $\bar{\pi}_N$
1: Set $\hat{\pi}_1 = \pi_1$ and weights $w_n = n^p$
2: Sample integer $K \in [1, N]$ with $P(K = n) \propto w_n$
3: for $n = 1 \dots K - 1$† do
4:     $\pi_n, \hat{g}_n = \texttt{PredictionStep}(\hat{\pi}_n, \Phi_n, H_{n-1}, w_n)$
5:     $\mathcal{D}_n, g_n = \texttt{DataCollection}(\pi_n)$
6:     $H_n, \hat{\pi}_{n+1} = \texttt{CorrectionStep}(\pi_n, e_n, H_{n-1}, w_n)$,
       where $e_n = g_n - \hat{g}_n$.
7:     $\Phi_{n+1} = \texttt{ModelUpdate}(\Phi_n, \mathcal{D})$, where $\mathcal{D} = \mathcal{D} \bigcup \mathcal{D}_n$.
8: end for
9: Set $\bar{\pi}_N = \pi_{K-1}$
```

Prediction Step is to solve the above minimization problem for π_n and use $\nabla f_n(\pi_n)$ as the effective prediction $\hat{g}_n$.

4.3. Summary: Why Does PICCOLO Work?

We provide a summary of the full algorithm for policy optimization in Algorithm 1. We see that PICCOLO uses the predicted gradient to take an extra step to accelerate learning, and, meanwhile, to prevent the error accumulation, it adaptively adjusts the step size (i.e. the regularization) based on the prediction error and corrects for the bias on the policy right away. To gain some intuition, let us consider ADAGRAD (Duchi et al., 2011) as a base algorithm[1]:

$$G_n = G_{n-1} + \operatorname{diag}(w_n g_n \odot w_n g_n)$$
$$\pi_{n+1} = \arg\min_{\pi\in\Pi} \langle w_n g_n, \pi\rangle + \frac{1}{2\eta}(\pi - \pi_n)^\top G_n^{1/2}(\pi - \pi_n)$$

where $G_0 = \epsilon I$ and $\eta, \epsilon > 0$, and $\odot$ denotes element-wise multiplication. This update has an `adapt` operation as $\texttt{adapt}(h, H, g, w) = G + \operatorname{diag}(wg \odot wg)$ which updates the Bregman divergence based on the gradient size.

PICCOLO transforms ADAGRAD into a new algorithm. In the Prediction Step, it performs

$$\pi_n = \arg\min_{\pi\in\Pi} \langle w_n \hat{g}_n, \pi\rangle + \frac{1}{2\eta}(\pi - \pi_{n-1})^\top G_{n-1}^{1/2}(\pi - \pi_{n-1})$$

In the Correction Step, it performs

$$G_n = G_{n-1} + \operatorname{diag}(w_n e_n \odot w_n e_n)$$
$$\hat{\pi}_{n+1} = \arg\min_{\pi\in\Pi} \langle w_n e_n, \pi\rangle + \frac{1}{2\eta}(\pi - \hat{\pi}_n)^\top G_n^{1/2}(\pi - \hat{\pi}_n)$$

We see that the PICCOLO-ADAGRAD updates G_n proportional to the prediction error e_n instead of g_n. It takes larger steps when models are accurate, and decreases the step size once the prediction deviates. As a result, PICCOLO is robust to model quality: it accelerates learning when the model is informative, and prevents inaccurate (potentially adversarial) models from hurting the policy. We will further demonstrate this in theory and in the experiments.

[1] We provide another example in Appendix E.

† Here we assume `project` is automatically performed inside `PredictionStep` and `CorrectionStep`.

5. Theoretical Analysis

In this section, we show that PICCOLO has two major benefits over previous approaches: 1) it accelerates policy learning when the models predict the required gradient well on average; and 2) it does not bias the performance of the policy, even when the prediction is incorrect.

To analyze PICCOLO, we introduce an assumption to quantify the `adapt` operator of a base algorithm.

Assumption 1. `adapt` chooses a regularization sequence such that, for some $M_N = o(w_{1:N})$, $\|H_0\|_\mathcal{R} + \sum_{n=1}^N \|H_n - H_{n-1}\|_\mathcal{R} \le M_N$ for some norm $\|\cdot\|_\mathcal{R}$ which measures the size of regularization.

This assumption, which requires the regularization to increase slower than the growth of $w_{1:N}$, is satisfied by most reasonably-designed base algorithms. For example, in a uniformly weighted problem, gradient descent with a decaying step size $O(\frac{1}{\sqrt{n}})$ has $M_N = O(\sqrt{N})$. In general, for stochastic problems, an optimal base algorithm would ensure $M_N = O(\frac{w_{1:N}}{\sqrt{N}})$.

5.1. Convergence Properties

Now we state the main result, which quantifies the regret of PICCOLO with respect to the sequence of linear loss functions that it has access to. The proof is given in Appendix F.

Theorem 1. *Suppose H_n defines a strongly convex function with respect to $\|\cdot\|_n$. Under Assumption 1, running* PICCOLO *ensures* $\sum_{n=1}^N \langle w_n g_n, \pi_n - \pi\rangle \le M_N + \sum_{n=1}^N \frac{w_n^2}{2}\|e_n\|_{*,n}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2$, *for all* $\pi \in \Pi$.

The term $\|e_n\|_{*,n}^2$ in Theorem 1 says that the performance of PICCOLO depends on how well the base algorithm adapts to the error e_n through the `adapt` operation in the Correction Step. Usually `adapt` updates H_n gradually (Assumption 1) while minimizing $\frac{1}{2}\|e_n\|_{*,n}^2$, like we showed in ADAGRAD.

In general, when the base algorithm is adaptive and optimal for adversarial problems, we show in Appendix G that its PICCOLOed version guarantees that $\mathbb{E}[\sum_{n=1}^N \langle w_n g_n, \pi_n - \pi\rangle] \le O(1) + C_{\Pi,\Phi}\frac{w_{1:N}}{\sqrt{N}}$, where $C_{\Pi,\Phi} = O(|\Pi| + E_\Phi + \sigma_g^2 + \sigma_{\hat{g}}^2)$ is some constant related to the diameter of Π (denoted as $|\Pi|$), the model bias E_Φ, and the sampling variance σ_g^2 and $\sigma_{\hat{g}}^2$ of g_n and $\hat{g}_n$, respectively. Through Lemma 2 and 3, this bound directly implies accelerated and bias-free policy performance.

Theorem 2. *Suppose $\tilde{l}_n$ is convex*[3] *and $w_n \ge \Omega(1)$. Then running* PICCOLO *yields* $\mathbb{E}[\mathit{Regret}_n(\tilde{l})/w_{1:N}] = O(\frac{C_{\Pi,\Phi}}{\sqrt{N}})$, *where* $C_{\Pi,\Phi} = O(|\Pi| + E_\Phi + \sigma_g^2 + \sigma_{\hat{g}}^2) = O(1)$.

[3] The convexity assumption is standard, as used in (Duchi et al., 2011; Ross et al., 2011; Kingma & Ba, 2015; Cheng & Boots,

Table 1: Upper bounds of the average regret of different policy optimization algorithms.

ALGORITHMS	UPPER BOUNDS IN BIG-O
PICCOLO	$\frac{1}{\sqrt{N}}\left(\lvert\Pi\rvert + \sigma_g^2 + \sigma_{\hat{g}}^2 + E_\Phi\right)$
MODEL-FREE	$\frac{1}{\sqrt{N}}\left(\lvert\Pi\rvert + G_g^2 + \sigma_g^2\right)$
MODEL-BASED	$\frac{1}{\sqrt{N}}\left(\lvert\Pi\rvert + G_{\hat{g}}^2 + \sigma_{\hat{g}}^2\right) + E_\Phi$
DYNA	$\frac{1}{\sqrt{2N}}\left(\lvert\Pi\rvert + \frac{1}{2}\left(G_g^2 + G_{\hat{g}}^2 + \sigma_g^2 + \sigma_{\hat{g}}^2\right)\right) + E_\Phi$

5.2. Comparison

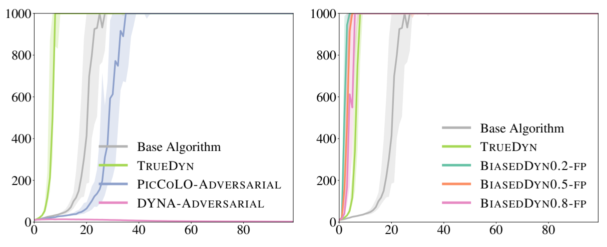
To appreciate the advantages of PICCOLO, we review several policy optimization algorithms and compare their regret. We show that they can be viewed as incomplete versions of PICCOLO, which only either result in accelerated learning *or* are unbiased, but not both (see in Table 1).

We first consider the common model-free approach (Sutton et al., 2000; Kakade, 2002; Peters & Schaal, 2008; Peters et al., 2010; Silver et al., 2014; Sun et al., 2017; Cheng et al., 2018), i.e. applying the base algorithm with g_n. To make the comparison concrete, suppose $\|\mathbb{E}[g_n]\|_*^2 \le G_g^2$ for some constant G_g, where we recall g_n is the sampled true gradient. As the model-free approach is equivalent to setting $\hat{g}_n = 0$ in PICCOLO, by Theorem 1 (with $e_n = g_n$), the constant C_Π in Theorem 2 would become $O(|\Pi| + G_g^2 + \sigma_g^2)$. In other words, PICCOLOing the base algorithm improves the constant factor from G_g^2 to $\sigma_{\hat{g}}^2 + E_\Phi$. Therefore, while the model-free approach is bias-free, its convergence can be further improved by PICCOLO, as long as the models $\{\Phi_n\}$ are reasonably accurate on average.[4]

Next we consider the pure model-based approach with a model that is potentially learned online (Jacobson & Mayne, 1970; Todorov & Li, 2005; Deisenroth & Rasmussen, 2011; Pan & Theodorou, 2014; Levine & Koltun, 2013; Sun et al., 2018). As this approach is equivalent to only performing the Prediction Step[5], its performance suffers from any modeling error. Specifically, suppose $\|\mathbb{E}[\hat{g}_n]\|_*^2 \le G_{\hat{g}}^2$ for some constant $G_{\hat{g}}$. One can show that the bound in Theorem 2 would become $O((|\Pi| + G_{\hat{g}}^2 + \sigma_{\hat{g}}^2)/\sqrt{N} + E_\Phi)$, introducing a constant bias in $O(E_\Phi)$.

A hybrid heuristic to combine the model-based and model-free updates is DYNA (Sutton, 1991; Sutton et al., 2012), which interleaves the two steps during policy optimization. This is equivalent to applying g_n, instead of the error e_n, in the Correction Step of PICCOLO. Following a similar analysis as above, one can show that the convergence rate in Theorem 2 would become $O((|\Pi| + G^2 + \sigma^2)/\sqrt{2N} + E_\Phi)$, where $G^2 = \frac{1}{2}(G_g^2 + G_{\hat{g}}^2)$ and $\sigma^2 = \frac{1}{2}(\sigma_g^2 + \sigma_{\hat{g}}^2)$. Therefore, DYNA is effectively twice as fast as the pure model-free approach when the model is accurate. However, it would eventually suffer from the performance bias due model error, as reflected in the term E_Φ. We will demonstrate this property experimentally in Figure 1.

Finally, we note that the idea of using Φ_n as control variate (Chebotar et al., 2017; Grathwohl et al., 2018; Papini et al., 2018) is orthogonal to the setups considered above, and it can be naturally combined with PICCOLO. For example, we can also use Φ_n to compute a better sampled gradient g_n with smaller variance (line 5 of Algorithm 1). This would improve σ_g^2 in the bounds of PICCOLO to a smaller $\tilde{\sigma}_g^2$, the size of reduced variance.



(a) Adversarial model (b) Different model fidelities

Figure 1: Performance of PICCOLO with different predictive models. x axis is iteration number and y axis is sum of rewards. The curves are the median among 8 runs with different seeds, and the shaded regions account for 25% percentile. ADAM is used as the base algorithm, and the update rule, by default, is PICCOLO; e.g. TRUEDYN in (a) refers to PICCOLO with TRUEDYN predictive model. (a) Comparison of PICCOLO and DYNA with adversarial model. (b) PICCOLO with the fixed-point setting (8) with dynamics model in different fidelities. BIASEDDYN0.8 indicates that the mass of each individual robot link is either increased or decreased by 80% with probability 0.5 respectively.

6. Experiments

We corroborate our theoretical findings with experiments[6] in learning neural network policies to solve robot RL tasks (CartPole, Hopper, Snake, and Walker3D) from OpenAI Gym (Brockman et al., 2016) with the DART physics engine (Lee et al., 2018)[7]. The aim is to see if PICCOLO improves the performance of a base algorithm, even though in these experiments the convexity assumption in the theory does not hold. We choose several popular first-order mir-

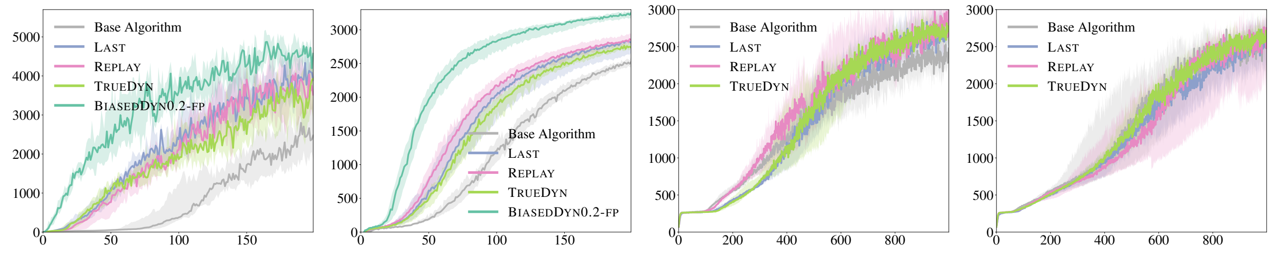
2018), which holds for tabular problems as well as some special cases, like continuous-time problems (cf. (Cheng & Boots, 2018)).

[4]It can be shown that if the model is learned online with a no-regret algorithm, it would perform similarly to the best model in the hindsight (cf. Appendix G.4)

[5]These algorithms can be realized by the fixed-point formulation of the Prediction Step with (arbitrarily small) regularization.

[6]The codes are available at `https://github.com/gtrll/rlfamily`.

[7]The environments are defined in DartEnv, hosted at `https://github.com/DartEnv`.



(a) Hopper ADAM (b) Snake ADAM (c) Walker3D NATGRAD (d) Walker3D TRPO

Figure 2: Performance of PICCOLO in various tasks. x axis is iteration number and y axis is sum of rewards. The curves are the median among 8 runs with different seeds, and the shaded regions account for 25% percentile.

ror descent base algorithms (ADAM (Kingma & Ba, 2015), natural gradient descent NATGRAD (Kakade, 2002), and trust-region optimizer TRPO (Schulman et al., 2015)). We compute g_n by GAE (Schulman et al., 2016). For predictive models, we consider off-policy gradients (with the samples of the last iteration LAST or a replay buffer REPLAY) and gradients computed through simulations with the true or biased dynamics models (TRUEDYN or BIASEDDYN). We will label a model with FP if $\hat{g}_n$ is determined by the fixed-point formulation (8)[8]; otherwise, $\hat{g}_n = \Phi_n(\hat{\pi}_n)$. Please refer to Appendix H for the details.

In Figure 1, we first use CartPole to study Theorem 2, which suggests that PICCOLO is unbiased and improves the performance when the prediction is accurate. Here we additionally consider an extremely bad model, ADVERSARIAL, that predicts the gradients adversarially.[9] Figure 1 (a) illustrates the performance of PICCOLO and DYNA, when ADAM is chosen as the base algorithm. We observe that PICCOLO improves the performance when the model is accurate (i.e. TRUEDYN). Moreover, PICCOLO is robust to modeling errors. It still converges when the model is adversarially attacking the algorithm, whereas DYNA fails completely. In Figure 1 (b), we conduct a finer comparison of the effects of different model accuracies (BIASEDDYN-FP), when $\hat{g}_n$ is computed using (8). To realize inaccurate dynamics models to be used in the Prediction step, we change the mass of links of the robot by a certain factor, e.g. BIASEDDYN0.8 indicates that the mass of each individual link is either increased or decreased by 80% with probability 0.5, respectively. We see that the fixed-point formulation (8), which makes multiple queries of Φ_n for computing $\hat{g}_n$, performs much better than the heuristic of setting $\hat{g}_n = \Phi(\hat{\pi}_n)$, even when the latter is using the true model (TRUEDYN). Overall, we see PICCOLO with BIASEDDYN-FP is able to accelerate learning, though with a degree varying with model accuracies; but even for models with a large bias, it still converges unbiasedly, as we previously observed in Figure 1 (a),

In Figure 2, we study the performance of PICCOLO in a range of environments. In general, we find that PICCOLO indeed improves the performance[10] though the exact degree depends on how $\hat{g}_n$ is computed. In Figure 2 (a) and (b), we show the results of using ADAM as the base algorithm. We observe that, while setting $\hat{g}_n = \Phi_n(\hat{\pi}_n)$ is already an effective heuristic, the performance of PICCOLO can be further and largely improved if we adopt the fixed-point strategy in (8), as the latter allows the learner to take more globally informed update directions. Finally, to demonstrate the flexibility of the proposed framework, we also "PICCOLO" two other base algorithms, NATGRAD and TRPO, in Figure 2 (c) and (d), respectively. The complete set of experimental results can be found in Appendix H.

7. Conclusion

PICCOLO is a general reduction-based framework for solving predictable online learning problems. It can be viewed as an automatic strategy for generating new algorithms that can leverage prediction to accelerate convergence. Furthermore, PICCOLO uses the Correction Step to recover from the mistake made in the Prediction Step, so the presence of modeling errors does not bias convergence, as we show in both the theory and experiments. The design of PICCOLO leaves open the question of how to design good predictive models. While PICCOLO is robust against modeling error, the accuracy of a predictive model can affect its effectiveness. PICCOLO only improves the performance when the model can make non-trivial predictions. In the experiments, we found that off-policy and simulated gradients are often useful, but they are not perfect. It would be interesting to see whether a predictive model that is trained to directly minimize the prediction error can further help policy learning. Finally, we note that, despite the focus of this paper on policy optimization, PICCOLO can naturally be applied to other optimization and learning problems.

[8]In implementation, we solve the corresponding optimization problem with a few number of iterations. For example, BIASEDDYN-FP is aporoximatedly solved with 5 iterations.

[9]We set $\hat{g}_{n+1} = -\left(\max_{m=1,\dots,n} \|g_m\| / \|g_n\|\right) g_n$.

[10]Note that different base algorithms are not directly comparable, as further fine-tuning of step sizes is required.

Acknowledgements

This research is supported in part by NSF NRI 1637758 and NSF CAREER 1750483.

References

Amari, S.-I. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Amos, B., Jimenez, I., Sacks, J., Boots, B., and Kolter, J. Z. Differentiable MPC for end-to-end planning and control. In *Advances in Neural Information Processing Systems*, pp. 8299–8310, 2018.

Anthony, T., Tian, Z., and Barber, D. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems*, pp. 5360–5370, 2017.

Beck, A. and Teboulle, M. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Chebotar, Y., Hausman, K., Zhang, M., Sukhatme, G., Schaal, S., and Levine, S. Combining model-based and model-free updates for trajectory-centric reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 703–711, 2017.

Cheng, C.-A. and Boots, B. Convergence of value aggregation for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, volume 84, pp. 1801–1809, 2018.

Cheng, C.-A., Yan, X., Wagener, N., and Boots, B. Fast policy learning through imitation and reinforcement. In *Proceedings of the 34th Conference on Uncertanty in Artificial Intelligence*, pp. 845–855, 2018.

Cheng, C.-A., Yan, X., Theodorou, E., and Boots, B. Accelerating imitation learning with predictive models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

Chiang, C.-K., Yang, T., Lee, C.-J., Mahdavi, M., Lu, C.-J., Jin, R., and Zhu, S. Online optimization with gradual variations. In *Conference on Learning Theory*, pp. 6–1, 2012.

Daniely, A., Gonen, A., and Shalev-Shwartz, S. Strongly adaptive online learning. In *International Conference on Machine Learning*, pp. 1405–1411, 2015.

Deisenroth, M. and Rasmussen, C. E. Pilco: A model-based and data-efficient approach to policy search. In *International Conference on machine learning*, pp. 465–472, 2011.

Diakonikolas, J. and Orecchia, L. Accelerated extra-gradient descent: A novel accelerated first-order method. *arXiv preprint arXiv:1706.04680*, 2017.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pp. 1329–1338, 2016.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Gordon, G. J. Regret bounds for prediction problems. In *Annual Conference on Computational Learning Theory*, pp. 29–40. ACM, 1999.

Grathwohl, W., Choi, D., Wu, Y., Roeder, G., and Duvenaud, D. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018.

Gupta, V., Koren, T., and Singer, Y. A unified approach to adaptive regularization in online and stochastic optimization. *arXiv preprint arXiv:1706.06569*, 2017.

Hazan, E., Agarwal, A., and Kale, S. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

Hazan, E. et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Ho-Nguyen, N. and Kılınç-Karzan, F. Exploiting problem structure in optimization under uncertainty via online convex optimization. *Mathematical Programming*, pp. 1–35, 2018.

Jacobson, D. H. and Mayne, D. Q. Differential dynamic programming. 1970.

Jadbabaie, A., Rakhlin, A., Shahrampour, S., and Sridharan, K. Online optimization: Competing with dynamic comparators. In *Artificial Intelligence and Statistics*, pp. 398–406, 2015.

Juditsky, A., Nemirovski, A., and Tauvel, C. Solving variational inequalities with stochastic mirror-prox algorithm. *Stochastic Systems*, 1(1):17–58, 2011.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, volume 2, pp. 267–274, 2002.

Kakade, S. M. A natural policy gradient. In *Advances in neural information processing systems*, pp. 1531–1538, 2002.

Kalai, A. and Vempala, S. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms. In *Advances in neural information processing systems*, pp. 1008–1014, 2000.

Korpelevich, G. The extragradient method for finding saddle points and other problems. *Matecon*, 12:747–756, 1976.

Lee, J., Grey, M. X., Ha, S., Kunz, T., Jain, S., Ye, Y., Srinivasa, S. S., Stilman, M., and Liu, C. K. DART: Dynamic animation and robotics toolkit. *The Journal of Open Source Software*, 3(22):500, feb 2018.

Levine, S. and Koltun, V. Guided policy search. In *International Conference on Machine Learning*, pp. 1–9, 2013.

McMahan, H. B. A survey of algorithms and analysis for adaptive online learning. *The Journal of Machine Learning Research*, 18(1):3117–3166, 2017.

McMahan, H. B. and Streeter, M. Adaptive bound optimization for online convex optimization. In *COLT 2010 - The 23rd Conference on Learning Theory*, 2010.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Nemirovski, A. Prox-method with rate of convergence o (1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1): 229–251, 2004.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Oh, J., Singh, S., and Lee, H. Value prediction network. In *Advances in Neural Information Processing Systems*, pp. 6120–6130, 2017.

Pan, Y. and Theodorou, E. Probabilistic differential dynamic programming. In *Advances in Neural Information Processing Systems*, pp. 1907–1915, 2014.

Papini, M., Binaghi, D., Canonaco, G., Pirotta, M., and Restelli, M. Stochastic variance-reduced policy gradient. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 4023–4032, 2018.

Pascanu, R., Li, Y., Vinyals, O., Heess, N., Buesing, L., Racanière, S., Reichert, D., Weber, T., Wierstra, D., and Battaglia, P. Learning model-based planning from scratch. *arXiv preprint arXiv:1707.06170*, 2017.

Peters, J. and Schaal, S. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

Peters, J., Mülling, K., and Altun, Y. Relative entropy policy search. In *AAAI*, pp. 1607–1612. Atlanta, 2010.

Rakhlin, A. and Sridharan, K. Online learning with predictable sequences. In *COLT 2013 - The 26th Annual Conference on Learning Theory*, pp. 993–1019, 2013a.

Rakhlin, S. and Sridharan, K. Optimization, learning, and games with predictable sequences. In *Advances in Neural Information Processing Systems*, pp. 3066–3074, 2013b.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

Ross, S. and Bagnell, J. A. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *International conference on artificial intelligence and statistics*, pp. 627–635, 2011.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *Proceedings of the 31th International Conference on Machine Learning*, pp. 387–395, 2014.

Silver, D., van Hasselt, H., Hessel, M., Schaul, T., Guez, A., Harley, T., Dulac-Arnold, G., Reichert, D., Rabinowitz, N., Barreto, A., et al. The predictron: End-to-end learning and planning. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362 (6419):1140–1144, 2018. ISSN 0036-8075.

Srinivas, A., Jabri, A., Abbeel, P., Levine, S., and Finn, C. Universal planning networks: Learning generalizable representations for visuomotor control. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Sun, W., Venkatraman, A., Gordon, G. J., Boots, B., and Bagnell, J. A. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3309–3318, 2017.

Sun, W., Gordon, G. J., Boots, B., and Bagnell, J. A. Dual policy iteration. In *Advances in Neural Information Processing Systems 31*, pp. 7059–7069, 2018.

Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4): 160–163, 1991.

Sutton, R. S. and Barto, A. G. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pp. 1057–1063, 2000.

Sutton, R. S., Szepesvári, C., Geramifard, A., and Bowling, M. P. Dyna-style planning with linear function approximation and prioritized sweeping. *arXiv preprint arXiv:1206.3285*, 2012.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. Sim-to-real: Learning agile locomotion for quadruped robots. In *Robotics: Science and Systems XIV*, 2018.

Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4 (2):26–31, 2012.

Todorov, E. and Li, W. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear systems. In *American Control Conference*, pp. 300–306. IEEE, 2005.

Zeiler, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning*, pp. 928–936, 2003.