A. Relationship between PICCOLO and Existing Algorithms

We discuss how the framework of PICCOLO unifies existing online learning algorithms and provides their natural adaptive generalization. To make the presentation clear, we summarize the effective update rule of PICCOLO when the base algorithm is mirror descent

$$\begin{aligned} \pi_n &= \arg\min_{\pi\in\Pi} \langle w_n \hat{g}_n, \pi\rangle + B_{R_{n-1}}(\pi||\hat{\pi}_n) \\ \hat{\pi}_{n+1} &= \arg\min_{\pi\in\Pi} \langle w_n e_n, \pi\rangle + B_{R_n}(\pi||\pi_n) \end{aligned} \tag{9}$$

and that when the base algorithm is FTRL,

$$\begin{aligned} \pi_n &= \arg\min_{\pi\in\Pi} \langle w_n \hat{g}_n, \pi\rangle + \sum_{m=1}^{n-1} \langle w_m g_m, \pi\rangle + B_{r_m}(\pi||\pi_m) \\ \hat{\pi}_{n+1} &= \arg\min_{\pi\in\Pi} \langle w_n e_n, \pi\rangle + B_{r_n}(\pi||\pi_n) + \langle w_n \hat{g}_n, \pi\rangle + \sum_{m=1}^{n-1} \langle w_m g_m, \pi\rangle + B_{r_m}(\pi||\pi_m) \end{aligned} \tag{10}$$

Because $e_n = g_n - \hat{g}_n$, PICCOLO with FTRL exactly matches the update rule (MOBIL) proposed by Cheng et al. (2018)

$$\begin{aligned} \pi_n &= \arg\min_{\pi\in\Pi} \langle w_n \hat{g}_n, \pi\rangle + \sum_{m=1}^{n-1} \langle w_m g_m, \pi\rangle + B_{r_m}(\pi||\pi_m) \\ \hat{\pi}_{n+1} &= \arg\min_{\pi\in\Pi} \sum_{m=1}^{n} \langle w_m g_m, \pi\rangle + B_{r_m}(\pi||\pi_m) \end{aligned} \tag{11}$$

As comparisons, we consider existing two-step update rules, which in our notation can be written as follows:

- Extragradient descent (Korpelevich, 1976), mirror-prox (Nemirovski, 2004; Juditsky et al., 2011) or optimistic mirror descent (Chiang et al., 2012; Rakhlin & Sridharan, 2013a)

$$\begin{aligned} \pi_n &= \arg\min_{\pi\in\Pi} \langle \hat{g}_n, \pi\rangle + B_R(\pi||\hat{\pi}_n) \\ \hat{\pi}_{n+1} &= \arg\min_{\pi\in\Pi} \langle g_n, \pi\rangle + B_R(\pi||\hat{\pi}_n) \end{aligned} \tag{12}$$

- FTRL-with-Prediction/optimistic FTRL (Rakhlin & Sridharan, 2013a)

$$\pi_n = \arg\min_{\pi\in\Pi} R(\pi) + \langle \hat{g}_n, \pi\rangle + \sum_{m=1}^{n-1} \langle w_m g_m, \pi\rangle \tag{13}$$

Let us first review the previous update rules. Originally extragradient descent (Korpelevich, 1976) and mirror prox (Nemirovski, 2004; Juditsky et al., 2011) were proposed to solve VIs (the latter is an extension to consider general Bregman divergences). As pointed out by Cheng et al. (2019), when applied to an online learning problem, these algorithms effectively assign $\hat{g}_n$ to be the online gradient as if the learner plays a decision at $\hat{\pi}_n$. On the other hand, in the online learning literature, optimistic mirror descent (Chiang et al., 2012) was proposed to use $\hat{g}_n = g_{n-1}$. Later (Rakhlin & Sridharan, 2013a) generalized it to use some arbitrary sequence $\hat{g}_n$, and provided a FTRL version update rule in (13). However, it is unclear in (Rakhlin & Sridharan, 2013a) where the prediction $\hat{g}_n$ comes from in general, though they provide an example in the form of learning from experts.

Recently Cheng et al. (2018) generalized the FTRL version of these ideas to design MOBIL, which introduces extra features 1) use of weights 2) non-stationary Bregman divergences (i.e. step size) and 3) the concept of predictive models ($\Phi_n \approx \nabla l_n$). The former two features are important to speed up the convergence rate of IL. With predictive models, they propose a conceptual idea (inspired by Be-the-Leader) which solves for π_n by the VI of finding π_n such that

$$\left\langle w_n \Phi_n(\pi_n) + \sum_{m=1}^{n} w_m g_m, \pi' - \pi_n \right\rangle \geq 0 \qquad \forall \pi' \in \Pi \tag{14}$$

and a more practical version (11) which sets $\hat{g}_n = \Phi_n(\pi_n)$. Under proper assumptions, they prove that the practical version achieves the same rate of non-asymptotic convergence as the conceptual one, up to constant factors.

PICCOLO unifies and generalizes the above update rules. We first notice that when the weight is constant, the set Π is unconstrained, and the Bregman divergence is constant, PICCOLO with mirror descent in (9) is the same as (12), i.e.,

$$\begin{aligned}
\hat{\pi}_{n+1} &= \arg\min_{\pi\in\Pi} \langle e_n, \pi\rangle + B_R(\pi||\pi_n) \\
&= \arg\min_{\pi\in\Pi} \langle e_n, \pi\rangle + R(\pi) - \langle \nabla R(\pi_n), \pi\rangle \\
&= \arg\min_{\pi\in\Pi} \langle g_n - \hat{g}_n, \pi\rangle + R(\pi) - \langle \nabla R(\hat{\pi}_n) - \hat{g}_n, \pi\rangle \\
&= \arg\min_{\pi\in\Pi} \langle g_n, \pi\rangle + R(\pi) - \langle \nabla R(\hat{\pi}_n), \pi\rangle \\
&= \arg\min_{\pi\in\Pi} \langle g_n, \pi\rangle + B_R(\pi||\hat{\pi}_n)
\end{aligned}$$

Therefore, PICCOLO with mirror descent includes previous two-step algorithms with proper choices of $\hat{g}_n$. On the other hand, we showed above that PICCOLO with FTRL (10) recovers exactly (11).

PICCOLO further generalizes these updates in two important aspects. First, it provides a systematic way to make these mirror descent and FTRL algorithms *adaptive*, by the reduction that allows reusing existing adaptive algorithm designed for adversarial settings. By contrast, all the previous update schemes discussed above (even MOBIL) are based on constant or pre-scheduled Bregman divergences, which requires the knowledge of several constants of problem properties that are usually unknown in practice. The use of adaptive schemes more amenable to hyperparameter tuning in practice.

Second, PICCOLO generalize the use of predictive models from the VI formulation in (14) to the *fixed-point* formulation in (8). One can show that when the base algorithm is FTRL and we remove the Bregman divergence[11], (8) is the same as (14). In other words, (14) essentially can be viewed as a mechanism to find $\hat{g}_n$ for (11). But importantly, the fixed-point formulation is method agnostic and therefore applies to also the mirror descent case. In particular, in Section 4.2.3, we point out that when Φ_n is a gradient map, the fixed-point problem reduces to finding a stationary point[12] of a non-convex optimization problem. This observation makes implementation of the fixed-point idea much easier and more stable in practice (as we only require the function associated with Φ_n to be lower bounded to yield a stable problem).

B. Proof of Lemma 3

Without loss of generality we suppose $w_1 = 1$ and $J(\pi) \geq 0$ for all π. And we assume the weighting sequence $\{w_n\}$ satisfies, for all $n \geq m \geq 1$ and $k \geq 0$, $\frac{w_{n+k}}{w_n} \leq \frac{w_{m+k}}{w_m}$. This means $\{w_n\}$ is an non-decreasing sequence and it does not grow faster than exponential (for which $\frac{w_{n+k}}{w_n} = \frac{w_{m+k}}{w_m}$). For example, if $w_n = n^p$ with $p \geq 0$, it easy to see that

$$\frac{(n+k)^p}{n^p} \leq \frac{(m+k)^p}{m^p} \impliedby \frac{n+k}{n} \leq \frac{m+k}{m} \impliedby \frac{k}{n} \leq \frac{k}{m}$$

For simplicity, let us first consider the case where l_n is deterministic. Given this assumption, we bound the performance in terms of the weighted regret below. For l_n defined in (5), we can write

$$\begin{aligned}
&\sum_{n=1}^{N} w_n J(\pi_n) \\
&= \sum_{n=1}^{N} w_n J(\pi_{n-1}) + w_n \mathbb{E}_{d_{\pi_n}} \mathbb{E}_{\pi_n}[A_{\pi_{n-1}}] \\
&= \sum_{n=1}^{N} w_n J(\pi_{n-1}) + w_n l_n(\pi_n)
\end{aligned}$$

[11] Originally the conceptual MOBIL algorithm is based on the assumption that l_n is strongly convex and therefore does not require extra Bregman divergence. Here PICCOLO with FTRL provides a natural generalization to online convex problems.

[12] Any stationary point will suffice.

$$
\begin{aligned}
&= w_1 J(\pi_0) + \sum_{n=1}^{N-1} w_{n+1} J(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \\
&= w_1 J(\pi_0) + \sum_{n=1}^{N-1} w_{n+1} J(\pi_{n-1}) + \sum_{n=1}^{N-1} w_{n+1} l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \\
&= (w_1 + w_2) J(\pi_0) + \sum_{n=1}^{N-2} w_{n+2} J(\pi_n) + \sum_{n=1}^{N-1} w_{n+1} l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \\
&= w_{1:N} J(\pi_0) + \left(w_N l_1(\pi_1) + \sum_{n=1}^{2} w_{n+N-2} l_n(\pi_n) + \cdots + \sum_{n=1}^{N-1} w_{n+1} l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \right) \\
&= w_{1:N} J(\pi_0) + \left(w_N l_1(\pi_1) + \sum_{n=1}^{2} \frac{w_{n+N-2}}{w_n} w_n l_n(\pi_n) + \cdots + \sum_{n=1}^{N-1} \frac{w_{n+1}}{w_n} w_n l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \right) \\
&\leq w_{1:N} J(\pi_0) + \left(w_N l_1(\pi_1) + \frac{w_{N-1}}{w_1} \sum_{n=1}^{2} w_n l_n(\pi_n) + \cdots + \frac{w_2}{w_1} \sum_{n=1}^{N-1} w_n l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \right) \\
&= w_{1:N} J(\pi_0) + \left(w_N l_1(\pi_1) + w_{N-1} \sum_{n=1}^{2} w_n l_n(\pi_n) + \cdots + w_2 \sum_{n=1}^{N-1} w_n l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \right)
\end{aligned}
$$

where the inequality is due to the assumption on the weighting sequence.

We can further rearrange the second term in the final expression as

$$
\begin{aligned}
& w_N l_1(\pi_1) + w_{N-1} \sum_{n=1}^{2} w_n l_n(\pi_n) + \cdots + w_2 \sum_{n=1}^{N-1} w_n l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \\
=& w_N \left(l_1(\pi_1) - \min_{\pi \in \Pi} l_1(\pi) + \min_{\pi \in \Pi} l_1(\pi) \right) \\
& + w_{N-1} \left(\sum_{n=1}^{2} w_n l_n(\pi_n) - \min_{\pi \in \Pi} \sum_{n=1}^{2} w_n l_n(\pi) + \min_{\pi \in \Pi} \sum_{n=1}^{2} w_n l_n(\pi) \right) \\
& + \cdots + \sum_{n=1}^{N} w_n l_n(\pi_n) - \min_{\pi \in \Pi} \sum_{n=1}^{N} w_n l_n(\pi) + \min_{\pi \in \Pi} \sum_{n=1}^{N} w_n l_n(\pi) \\
=& \sum_{n=1}^{N} w_{N-n+1} \left(\text{Regret}_n(f) + w_{1:n} \epsilon_n(f) \right)
\end{aligned}
$$

where the last equality is due to the definition of *static* regret and ϵ_n.

Likewise, we can also write the above expression in terms of *dynamic regret*

$$
\begin{aligned}
& w_N l_1(\pi_1) + w_{N-1} \sum_{n=1}^{2} w_n l_n(\pi_n) + \cdots + w_2 \sum_{n=1}^{N-1} w_n l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \\
=& w_N \left(l_1(\pi_1) - \min_{\pi \in \Pi} l_1(\pi) + \min_{\pi \in \Pi} l_1(\pi) \right) \\
& + w_{N-1} \left(\sum_{n=1}^{2} w_n l_n(\pi_n) - \sum_{n=1}^{2} w_n \min_{\pi \in \Pi} l_n(\pi) + \sum_{n=1}^{2} \min_{\pi \in \Pi} w_n l_n(\pi) \right) \\
& + \cdots + \sum_{n=1}^{N} w_n l_n(\pi_n) - \sum_{n=1}^{N} \min_{\pi \in \Pi} w_n l_n(\pi) + \sum_{n=1}^{N} \min_{\pi \in \Pi} w_n l_n(\pi) \\
=& \sum_{n=1}^{N} w_{N-n+1} \left(\text{Regret}_n^d(l) + w_{1:n} \epsilon_n^d(l) \right)
\end{aligned}
$$

in which we define the weighted dynamic regret as

$$\text{Regret}_n^d(l) = \sum_{m=1}^{n} w_m l_m(\pi_m) - \sum_{m=1}^{n} w_m \min_{\pi\in\Pi} l_m(\pi)$$

and an expressive measure based on dynamic regret

$$\epsilon_n^d = \frac{1}{w_{1:n}} \sum_{m=1}^{n} w_m \min_{\pi\in\Pi} l_m(\pi) \le 0$$

For stochastic problems, because π_n does not depends on $\tilde{l}_n$, the above bound applies to the performance in expectation. Specifically, let h_{n-1} denote all the random variables observed before making decision π_n and seeing $\tilde{l}_n$. As π_n is made independent of $\tilde{l}_n$, we have, for example,

$$\begin{aligned}\mathbb{E}[l_n(\pi_n)|h_{n-1}] &= \mathbb{E}[l_n(\pi_n)|h_{n-1}] - \mathbb{E}[l_n(\pi_n^*)|h_{n-1}] + \mathbb{E}[l_n(\pi_n^*)|h_{n-1}] \\ &= \mathbb{E}[\tilde{l}_n(\pi_n)|h_{n-1}] - \mathbb{E}[\tilde{l}_n(\pi_n^*)|h_{n-1}] + \mathbb{E}[l_n(\pi_n^*)|h_{n-1}] \\ &\le \mathbb{E}[\tilde{l}_n(\pi_n) - \min_{\pi\in\Pi} \tilde{l}_n(\pi)|h_{n-1}] + \mathbb{E}[l_n(\pi_n^*)|h_{n-1}]\end{aligned}$$

where $\pi_n^* = \arg\min_{\pi\in\Pi} l_n(\pi)$. By applying a similar derivation as above recursively, we can extend the previous deterministic bounds to bounds in expectation (for both the static or the dynamic regret case), proving the desired statement.

C. The Basic Operations of Base Algorithms

We provide details of the abstract basic operations shared by different base algorithms. In general, the update rule of any base mirror-descent or FTRL algorithm can be represented in terms of the three basic operations

$$h \leftarrow \texttt{update}(h, H, g, w), \qquad H \leftarrow \texttt{adapt}(h, H, g, w), \qquad \pi \leftarrow \texttt{project}(h, H) \tag{15}$$

where `update` and `project` can be identified standardly, for mirror descent as,

$$\texttt{update}(h, H, g, w) = \arg\min_{\pi'\in\Pi} \langle wg, \pi'\rangle + B_H(\pi||h), \qquad \texttt{project}(h, H) = h \tag{16}$$

and for FTRL as,

$$\texttt{update}(h, H, g, w) = h + wg, \qquad \texttt{project}(h, H) = \arg\min_{\pi'\in\Pi} \langle h, \pi'\rangle + H(\pi') \tag{17}$$

We note that in the main text of this paper the operation `project` is omitted for simplicity, as it is equal to the identify map for mirror descent. In general, it represents the decoding from the abstract representation of the decision h to π. The main difference between and h and π is that h represents the sufficient information that defines the state of the base algorithm.

While `update` and `project` are defined standardly, the exact definition of `adapt` depends on the specific base algorithm. Particularly, `adapt` may depend also on whether the problem is weighted, as different base algorithms may handle weighted problems differently. Based on the way weighted problems are handled, we roughly categorize the algorithms (in both mirror descent and FTRL families) into two classes: the *stationary* regularization class and the *non-stationary* regularization class. Here we provide more details into the algorithm-dependent `adapt` operation, through some commonly used base algorithms as examples.

Please see also Appendix A for connection between PICCOLO and existing two-step algorithms, like optimistic mirror descent (Rakhlin & Sridharan, 2013b).

C.1. Stationary Regularization Class

The `adapt` operation of these base algorithms features two major functions: 1) a moving-average adaptation and 2) a step-size adaption. The moving-average adaptation is designed to estimate some statistics G such that $\|g\|_* = O(G)$ (which is an important factor in regret bounds), whereas the step-size adaptation updates a scalar multiplier η according to the weight w to ensure convergence.

This family of algorithms includes basic mirror descent (Beck & Teboulle, 2003) and FTRL (McMahan & Streeter, 2010; McMahan, 2017) with a scheduled step size, and adaptive algorithms based on moving average e.g. RMSPROP (Tieleman & Hinton, 2012) ADADELTA (Zeiler, 2012), ADAM (Kingma & Ba, 2015), AMSGRAD (Reddi et al., 2018), and the adaptive NATGRAD we used in the experiments. Below we showcase how `adapt` is defined using some examples.

C.1.1. BASIC MIRROR DESCENT (BECK & TEBOULLE, 2003)

We define G to be some constant such that $G \geq \sup \|g_n\|_*$ and define

$$\eta_n = \frac{\eta}{1 + cw_{1:n}/\sqrt{n}}, \tag{18}$$

as a function of the iteration counter n, where $\eta > 0$ is a step size multiplier and $c > 0$ determines the decaying rate of the step size. The choice of hyperparameters η, c pertains to how far the optimal solution is from the initial condition, which is related to the size of Π. In implementation, `adapt` updates the iteration counter n and updates the multiplier η_n using w_n in (18).

Together (n, G, η_n) defines $H_n = R_n$ in the mirror descent update rule (6) through setting $R_n = \frac{G}{\eta_n} R$, where R is a strongly convex function. That is, we can write (6) equivalently as

$$\begin{aligned}
\pi_{n+1} &= \arg\min_{\pi \in \Pi} \langle w_n g_n, \pi \rangle + \frac{G}{\eta_n} B_R(\pi || \pi_n) \\
&= \arg\min_{\pi \in \Pi} \langle w_n g_n, \pi \rangle + B_{H_n}(\pi || \pi_n) \\
&= \texttt{update}(h_n, H_n, g_n, w_n)
\end{aligned}$$

When the weight is constant (i.e. $w_n = 1$), we can easily see this update rule is equivalent to the classical mirror descent with a step size $\frac{\eta/G}{1+c\sqrt{n}}$, which is the optimal step size (McMahan, 2017). For general $w_n = \Theta(n^p)$ with some $p > -1$, it can viewed as having an effective step size $\frac{w_n \eta_n}{G} = O(\frac{1}{G\sqrt{n}})$, which is optimal in the weighted setting. The inclusion of the constant G makes the algorithm invariant to the scaling of loss functions. But as the same G is used across all the iterations, the basic mirror descent is conservative.

C.1.2. BASIC FTRL (MCMAHAN, 2017)

We provide details of general FTRL

$$\pi_{n+1} = \arg\min_{\pi \in \Pi} \sum_{m=1}^{n} \langle g_m, \pi \rangle + B_{r_m}(\pi || \pi_m) \tag{19}$$

where $B_{r_m}(\cdot||\pi_m)$ is a Bregman divergence centered at π_m.

We define, in the nth iteration, h_n, H_n, and `project` of FTRL in (17) as

$$h_n = \sum_{m=1}^{n} w_m g_m, \qquad H_n(\pi) = \sum_{m=1}^{n} B_{r_m}(\pi || \pi_n), \qquad \texttt{project}(h, H) = \arg\min_{\pi' \in \Pi} \langle h, \pi' \rangle + H(\pi')$$

Therefore, we can see that $\pi_{n+1} = \texttt{project}(h_n, H_n)$ indeed gives the update (19):

$$\begin{aligned}
\pi_{n+1} &= \texttt{project}(h_n, H_n) \\
&= \texttt{project}(\sum_{m=1}^{n} w_m g_m, \sum_{m=1}^{n} B_{r_m}(\pi || \pi_n)) \\
&= \arg\min_{\pi \in \Pi} \sum_{m=1}^{n} \langle w_m g_m, \pi \rangle + B_{r_m}(\pi || \pi_m)
\end{aligned}$$

For the basic FTRL, the `adapt` operator is similar to the basic mirror descent, which uses a constant G and updates the memory (n, η_n) using (18). The main differences are how (G, η_n) is mapped to H_n and that the basic FTRL updates H_n

also using h_n (i.e. π_n). Specifically, it performs $H_n \leftarrow \texttt{adapt}(h_n, H_{n-1}, g_n, w_n)$ through the following:

$$H_n(\cdot) = H_{n-1}(\cdot) + B_{r_n}(\cdot||\pi_n)$$

where following (McMahan, 2017) we set

$$B_{r_n}(\pi||\pi_n) = G(\frac{1}{\eta_n} - \frac{1}{\eta_{n-1}})B_R(\pi||\pi_n)$$

and η_n is updated using some scheduled rule.

One can also show that the choice of η_n scheduling in (18) leads to an optimal regret. When the problem is uniformly weighted (i.e. $w_n = 1$), this gives exactly the update rule in (McMahan, 2017). For general $w_n = \Theta(n^p)$ with $p > -1$, a proof of optimality can be found, for example, in the appendix of (Cheng et al., 2019).

C.1.3. ADAM (KINGMA & BA, 2015) AND AMSGRAD (REDDI ET AL., 2018)

As a representing mirror descent algorithm that uses moving-average estimates, ADAM keeps in the memory of the statistics of the first-order information that is provided in `update` and `adapt`. Here we first review the standard description of ADAM and then show how it is summarized in

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

using properly constructed `update`, `adapt`, and `project` operations.

The update rule of ADAM proposed by Kingma & Ba (2015) is originally written as, for $n \geq 1$,[13]

$$\begin{aligned} m_n &= \beta_1 m_{n-1} + (1-\beta_1) g_n \\ v_n &= \beta_2 v_{n-1} + (1-\beta_2) g_n \odot g_n \\ \hat{m}_n &= m_n / (1-\beta_1^n) \\ \hat{v}_n &= v_n / (1-\beta_2^n) \\ \pi_{n+1} &= \pi_n - \eta_n \hat{m}_n \oslash (\sqrt{\hat{v}_n} + \epsilon) \end{aligned} \tag{20}$$

where $\eta_n > 0$ is the step size, $\beta_1, \beta_2 \in [0, 1)$ (default $\beta_1 = 0.9$ and $\beta_2 = 0.999$) are the mixing rate, and $0 < \epsilon \ll 1$ is some constant for stability (default $\epsilon = 10^{-8}$), and $m_0 = v_0 = 0$. The symbols $\odot$ and $\oslash$ denote element-wise multiplication and division, respectively. The third and the forth steps are designed to remove the 0-bias due to running moving averages starting from 0.

The above update rule can be written in terms of the three basic operations. First, we define the memories $h_n = (m_n, \pi_n)$ for policy and (v_n, η_n, n) for regularization that is defined as

$$H_n(\pi) = \frac{1}{2\eta_n} \pi^\top (\text{diag}(\sqrt{\hat{v}_n}) + \epsilon I)\pi \tag{21}$$

where $\hat{v}_n$ is defined in the original ADAM equation in (20).

The `adapt` operation updates the memory to (v_n, η_n, n) in the step

$$H_n \leftarrow \texttt{adapt}(h_n, H_{n-1}, g_n, w_n)$$

It updates the iteration counter n and η_n in the same way in the basic mirror descent using (18), and update v_n (which along with n defines $\hat{v}_n$ used in (21)) using the original ADAM equation in (20).

For `update`, we slightly modify the definition of `update` in (16) (replacing g_n with $\hat{m}_n$) to incorporate the moving average and write

$$\texttt{update}(h_n, H_n, g_n, w_n) = \arg\min_{\pi' \in \Pi} \langle w_n \hat{m}_n, \pi' \rangle + B_{H_n}(\pi'||\pi) \tag{22}$$

[13] We shift the iteration index so it conforms with our notation in online learning, in which π_1 is the initial policy before any update.

where m_n and $\hat{m}_n$ are defined the same as in the original ADAM equations in (20). One can verify that, with these definitions, the update rule in (7) is equivalent to the update rule (20), when the weight is uniform (i.e. $w_n = 1$).

Here the $\sqrt{\hat{v}_n}$ plays the role of G as in the basic mirror descent, which can be viewed as an estimate of the upper bound of $\|g_n\|_*$. ADAM achieves a better performance because a coordinate-wise online estimate is used. With this equivalence in mind, we can easily deduct that using the same scheduling of η_n as in the basic mirror descent would achieve an optimal regret (cf. (Kingma & Ba, 2015; Reddi et al., 2018)). We note that ADAM may fail to converge in some particular problems due to the moving average (Reddi et al., 2018). AMSGRAD (Reddi et al., 2018) modifies the moving average and uses strictly increasing estimates. However in practice AMSGRAD behaves more conservatively.

For weighted problems, we note one important nuance in our definition above: it separates the weight w_n from the moving average and considers w_n as part of the η_n update, because the growth of w_n in general can be much faster than the rate the moving average converges. In other words, the moving average can only be used to estimate a stationary property, not a time-varying one like w_n. Hence, we call this class of algorithms, the *stationary* regularization class.

C.1.4. ADAPTIVE NATGRAD

Given first-order information g_n and weight w_n, we consider an update rule based on Fisher information matrix:

$$\pi_{n+1} = \arg\min_{\pi\in\Pi} \langle w_n g_n, \pi\rangle + \frac{\sqrt{\hat{G}_n}}{2\eta_n}(\pi - \pi_n)^\top F_n(\pi - \pi_n) \tag{23}$$

where F_n is the Fisher information matrix of policy π_n (Amari, 1998) and $\hat{G}_n$ is an adaptive multiplier for the step size which we will describe. When $\hat{G}_n = 1$, the update in (23) gives the standard natural gradient descent update with step size η_n (Kakade, 2002) .

The role of $\hat{G}_n$ is to adaptively and *slowly* changes the step size to minimize $\sum_{n=1}^N \frac{\eta_n}{\sqrt{G_n}}\|g_n\|^2_{F_n,*}$, which plays an important part in the regret bound (see Section 5, Appendix F, and e.g. (McMahan, 2017) for details). Following the idea in ADAM, we update $\hat{G}_n$ by setting (with $G_0 = 0$)

$$\begin{aligned} G_n &= \beta_2 G_n + (1-\beta_2)\frac{1}{2} g_n^\top F_n^{-1} g_n \\ \hat{G}_n &= G_n/(1-\beta_2^n) \end{aligned} \tag{24}$$

similar to the concept of updating v_n and $\hat{v}_n$ in ADAM in (20), and update η_n in the same way as in the basic mirror descent using (18). Consequently, this would also lead to a regret like ADAM but in terms of a different local norm.

The `update` operation of adaptive NATGRAD is defined standardly in (6) (as used in the experiments). The `adapt` operation updates n and η_n like in ADAM and updates G_n through (24).

C.2. Non-Stationary Regularization Class

The algorithms in the non-stationary regularization class maintains a regularization that is increasing over the number of iterations. Notable examples of this class include ADAGRAD (Duchi et al., 2011) and ONLINE NEWTON STEP (Hazan et al., 2007), and its regularization function is updated by applying BTL in a secondary online learning problem whose loss is an upper bound of the original regret (see (Gupta et al., 2017) for details). Therefore, compared with the previous stationary regularization class, the adaption property of η_n and G_n exchanges: η_n here becomes constant and G_n becomes time-varying. This will be shown more clearly in the ADAGRAD example below. We note while these algorithms are designed to be optimal in the convex, they are often too conservative (e.g. decaying the step size too fast) for non-convex problems.

C.2.1. ADAGRAD

The update rule of the diagonal version of ADAGRAD in (Duchi et al., 2011) is given as

$$\begin{aligned} G_n &= G_{n-1} + \operatorname{diag}(g_n \odot g_n) \\ \pi_{n+1} &= \arg\min_{\pi\in\Pi} \langle g_n, \pi\rangle + \frac{1}{2\eta}(\pi - \pi_n)^\top (\epsilon I + G_n)^{1/2}(\pi - \pi_n) \end{aligned} \tag{25}$$

where $G_0 = 0$ and $\eta > 0$ is a constant. ADAGRAD is designed to be optimal for online linear optimization problems. Above we provide the update equations of its mirror descent formulation in (25); a similar FTRL is also available (again the difference only happens when Π is constrained).

In terms of our notation, its `update` and `project` are defined standardly as in (16), i.e.

$$\texttt{update}(h_n, H_n, g_n, w_n) = \arg\min_{\pi' \in \Pi} \langle w_n g_n, \pi' \rangle + B_{H_n}(\pi' || \pi_n) \tag{26}$$

and its `adapt` essentially only updates G_n:

$$\texttt{adapt}(h_n, H_{n-1}, g_n, w_n) : G_n = G_{n-1} + \operatorname{diag}(w_n g_n \odot w_n g_n)$$

where the regularization is defined the updated G_n and the constant η as

$$H_n(\pi) = \frac{1}{2\eta} \pi^\top (\epsilon I + G_n)^{1/2} \pi.$$

One can simply verify the above definitions of `update` and `adapt` agrees with (25).

D. A Practical Variation of PICCOLO

In Section 4.2.2, we show that, given a base algorithm in mirror descent/FTRL, PICCOLO generates a new first-order update rule by recomposing the three basic operations into

$$h_n = \texttt{update}(\hat{h}_n, H_{n-1}, \hat{g}_n, w_n) \qquad \text{[Prediction]} \tag{27}$$

$$\begin{aligned} H_n &= \texttt{adapt}(h_n, H_{n-1}, e_n, w_n) \\ \hat{h}_{n+1} &= \texttt{update}(h_n, H_n, e_n, w_n) \end{aligned} \qquad \text{[Correction]} \tag{28}$$

where $e_n = g_n - \hat{g}_n$ and $\hat{g}_n$ is an estimate of g_n given by a predictive model Φ_n.

Here we propose a slight variation which introduces another operation `shift` inside the Prediction Step. This leads to the new set of update rules:

$$\begin{aligned} \hat{H}_n &= \texttt{shift}(\hat{h}_n, H_{n-1}) \\ h_n &= \texttt{update}(\hat{h}_n, \hat{H}_n, \hat{g}_n, w_n) \end{aligned} \qquad \text{[Prediction]} \tag{29}$$

$$\begin{aligned} H_n &= \texttt{adapt}(h_n, \hat{H}_n, e_n, w_n) \\ \hat{h}_{n+1} &= \texttt{update}(h_n, H_n, e_n, w_n) \end{aligned} \qquad \text{[Correction]} \tag{30}$$

The new `shift` operator additionally changes the regularization based on $\hat{h}_n$ the current representation of the policy in the Prediction Step, *independent* of the predicted gradient $\hat{g}_n$ and weight w_n. The main purpose of including this additional step is to deal with numerical difficulties, such as singularity of H_n. For example, in natural gradient descent, the Fisher information of some policy can be close to being singular along the direction of the gradients that are evaluated at different policies. As a result, in the original Prediction Step of PICCOLO, H_{n-1} which is evaluated at π_{n-1} might be singular in the direction of $\hat{g}_n$ which is evaluated $\hat{\pi}_n$.

The new operator `shift` brings in an extra degree of freedom to account for such issue. Although from a theoretical point of view (cf. Appendix F) the use of `shift` would only increase regrets and should be avoided if possible, in practice, its merits in handling numerical difficulties can out weight the drawback. Because `shift` does not depend on the size of $\hat{g}_n$ and w_n, the extra regrets would only be proportional to $O(\sum_{n=1}^N \|\pi_n - \hat{\pi}_n\|_n)$, which can be smaller than other terms in the regret bound (see Appendix F).

E. Example: PICCOLOing Natural Gradient Descent

We give an alternative example to illustrate how one can use the above procedure to "PICCOLO" a base algorithm into a new algorithm. Here we consider the adaptive natural gradient descent rule in Appendix C as the base algorithm, which

(given first-order information g_n and weight w_n) updates the policy through

$$\pi_{n+1} = \arg\min_{\pi\in\Pi} \langle w_n g_n, \pi\rangle + \frac{\sqrt{\hat{G}_n}}{2\eta_n}(\pi-\pi_n)^\top F_n(\pi-\pi_n) \tag{31}$$

where F_n is the Fisher information matrix of policy π_n (Amari, 1998), η_n a scheduled learning rate, and $\hat{G}_n$ is an adaptive multiplier for the step size which we will shortly describe. When $\hat{G}_n = 1$, the update in (31) gives the standard natural gradient descent update with step size η_n (Kakade, 2002) .

The role of $\hat{G}_n$ is to adaptively and *slowly* changes the step size to minimize $\sum_{n=1}^N \frac{\eta_n}{\sqrt{G_n}}\|g_n\|^2_{F_n,*}$, which plays an important part in the regret bound (see Section 5, Appendix F, and e.g. (McMahan, 2017) for details). To this end, we update $\hat{G}_n$ by setting (with $G_0 = 0$)

$$G_n = \beta_2 G_{n-1} + (1-\beta_2)\tfrac{1}{2} g_n^\top F_n^{-1} g_n, \qquad \hat{G}_n = G_n/(1-\beta_2^n) \tag{32}$$

similar to the moving average update rule in ADAM, and update η_n in the same way as in the basic mirror descent algorithm (e.g. $\eta_n = O(1/\sqrt{n})$). As a result, this leads to a similar regret like ADAM with $\beta_1 = 0$, but in terms of a local norm specified by the Fisher information matrix.

Now, let's see how to PICCOLO the adaptive natural gradient descent rule above. First, it is easy to see that the adaptive natural gradient descent rule is an instance of mirror descent (with $h_n = \pi_n$ and $H_n(g) = \frac{\sqrt{\hat{G}_n}}{2\eta_n} g^\top F_n g$), so the `update` and `project` operations are defined in the standard way, as in Section 4.2.2. The `adapt` operation updates the iteration counter n, the learning rate η_n, and updates $\hat{G}_n$ through (32).

To be more specific, let us explicitly write out the Prediction Step and the Correction Step of the PICCOLOed adaptive natural gradient descent rule in closed form as below: e.g. if $\eta_n = \frac{1}{\sqrt{n}}$, then we can write them as

[Prediction]
$$\pi_n = \arg\min_{\pi\in\Pi} \langle w_n \hat{g}_n, \pi\rangle + \frac{\sqrt{\hat{G}_{n-1}}}{2\eta_{n-1}}(\pi-\hat{\pi}_n)^\top F_{n-1}(\pi-\hat{\pi}_n)$$

[Correction]
$$\begin{aligned}
\eta_n &= 1/\sqrt{n}\\
G_n &= \beta_2 G_{n-1} + (1-\beta_2)\tfrac{1}{2} g_n^\top F_n^{-1} g_n\\
\hat{G}_n &= G_n/(1-\beta_2^n)\\
\hat{\pi}_{n+1} &= \arg\min_{\pi\in\Pi} \langle w_n e_n, \pi\rangle + \frac{\sqrt{\hat{G}_n}}{2\eta_n}(\pi-\pi_n)^\top F_n(\pi-\pi_n)
\end{aligned}$$

F. Regret Analysis of PICCOLO

The main idea of PICCOLO is to achieve optimal performance in predictable online learning problems by *reusing* existing adaptive, optimal first-order algorithms that are designed for adversarial online learning problems. This is realized by the reduction techniques presented in this section.

Here we prove the performance of PICCOLO in general predictable online learning problems, independent of the context of policy optimization. In Appendix F.1, we first show an elegant reduction from predictable problems to adversarial problems. Then we prove Theorem 1 in Appendix F.2, showing how the optimal regret bound for predictable linear problems can be achieved by PICCOLOing mirror descent and FTRL algorithms. Note that we will abuse the notation l_n to denote the per-round losses in this general setting.

F.1. Reduction from Predictable Online Learning to Adversarial Online Learning

Consider a predictable online learning problem with per-round losses $\{l_n\}$. Suppose in round n, before playing π_n and revealing l_n, we have access to some prediction of l_n, called $\hat{l}_n$. In particular, we consider the case where $\hat{l}_n(\pi) = \langle \hat{g}_n, \pi\rangle$ for some vector $\hat{g}_n$. Running an (adaptive) online learning algorithm designed for the general adversarial setting is not optimal here, as its regret would be in $O(\sum_{n=1}^N \|\nabla l_n\|^2_{n,*})$, where $\|\cdot\|_n$ is some local norm chosen by the algorithm and $\|\cdot\|_{n,*}$ is its dual norm. Ideally, we would only want to pay for the information that is unpredictable. Specifically, we wish to achieve an optimal regret in $O(\sum_{n=1}^N \|\nabla l_n - \nabla \hat{l}_n\|^2_{n,*})$ instead (Rakhlin & Sridharan, 2013a).

To achieve the optimal regret bound yet without referring to specialized, nested two-step algorithms (e.g. mirror-prox (Juditsky et al., 2011), optimistic mirror descent (Rakhlin & Sridharan, 2013b), FTRL-prediction (Rakhlin & Sridharan, 2013a)), we consider decomposing a *predictable* problem with N rounds into an *adversarial* problem with $2N$ rounds:

$$\sum_{n=1}^{N} l_n(\pi_n) = \sum_{n=1}^{N} \hat{l}_n(\pi_n) + \delta_n(\pi_n) \tag{33}$$

where $\delta_n = l_n - \hat{l}_n$. Therefore, we can treat the predictable problem as a new adversarial online learning problem with a loss sequence $\hat{l}_1, \delta_1, \hat{l}_2, \delta_2, \ldots, \hat{l}_N, \delta_N$ and consider solving this new problem with some standard online learning algorithm designed for the adversarial setting.

Before analysis, we first introduce a new decision variable $\hat{\pi}_n$ and denote the decision sequence in this new problem as $\hat{\pi}_1, \pi_1, \hat{\pi}_2, \pi_2, \ldots, \hat{\pi}_N, \pi_N$, so the definition of the variables are consistent with that in the problem before. Because this new problem is unpredictable, the optimal regret of this new decision sequence is

$$\sum_{n=1}^{N} \hat{l}_n(\hat{\pi}_n) + \delta_n(\pi_n) - \min_{\pi\in\Pi} \sum_{n=1}^{N} \hat{l}_n(\pi) + \delta_n(\pi) = O(\sum_{n=1}^{N} \|\nabla \hat{l}_n\|_{n,*}^2 + \|\nabla \delta_n\|_{n+1/2,*}^2) \tag{34}$$

where the subscript $n + 1/2$ denotes the extra round due to the reduction.

At first glance, our reduction does not meet the expectation of achieving regret in $O(\sum_{n=1}^{N} \|\nabla l_n - \nabla \hat{l}_n\|_{n,*}^2) = O(\sum_{n=1}^{N} \|\nabla \delta_n\|_{n,*}^2)$. However, we note that the regret for the new problem is too loose for the regret of the original problem, which is

$$\sum_{n=1}^{N} \hat{l}_n(\pi_n) + \delta_n(\pi_n) - \min_{\pi\in\Pi} \sum_{n=1}^{N} \hat{l}_n(\pi) + \delta_n(\pi)$$

where the main difference is that originally we care about $\hat{l}_n(\pi_n)$ rather than $\hat{l}_n(\hat{\pi}_n)$. Specifically, we can write

$$\begin{aligned}\sum_{n=1}^{N} l_n(\pi_n) &= \sum_{n=1}^{N} \hat{l}_n(\pi_n) + \delta_n(\pi_n) \\ &= \left(\sum_{n=1}^{N} \hat{l}_n(\hat{\pi}_n) + \delta_n(\pi_n)\right) + \left(\sum_{n=1}^{N} \hat{l}_n(\pi_n) - \hat{l}_n(\hat{\pi}_n)\right)\end{aligned}$$

Therefore, if the update rule for generating the decision sequence $\hat{\pi}_1, \pi_1, \hat{\pi}_2, \pi_2, \ldots, \hat{\pi}_N, \pi_N$ contributes sufficient negativity in the term $\hat{l}_n(\pi_n) - \hat{l}_n(\hat{\pi}_n)$ compared with the regret of the new adversarial problem, then the regret of the original problem can be smaller than (34). This is potentially possible, as π_n is made after $\hat{l}_n$ is revealed. Especially, in the fixed-point formulation of PICCOLO, π_n and $\hat{l}_n$ can be decided simultaneously.

In the next section, we show that when the base algorithm, which is adopted to solve the new adversarial problem given by the reduction, is in the family of mirror descent and FTRL. Then the regret bound of PICCOLO with respect to the original predictable problem is optimal.

F.2. Optimal Regret Bounds for Predictable Problems

We show that if the base algorithm of PICCOLO belongs to the family of optimal mirror descent and FTRL designed for adversarial problems, then PICCOLO can achieve the optimal regret of predictable problems. In this subsection, we assume the loss sequence is linear, i.e. $l_n(\pi) = \langle g_n, \pi \rangle$ for some g_n, and the results are summarized as Theorem 1 in the main paper (in a slightly different notation).

F.2.1. MIRROR DESCENT

First, we consider mirror descent as the base algorithm. In this case, we can write the PICCOLO update rule as

$$\pi_n = \arg\min_{\pi\in\Pi} \left\langle \nabla \hat{l}_n(\hat{\pi}_n), x \right\rangle + B_{H_{n-1}}(\pi||\hat{\pi}_n) \qquad \text{[Prediction]}$$

$$\hat{\pi}_{n+1} = \arg\min_{\pi \in \Pi} \langle \nabla \delta_n(\pi_n), \pi \rangle + B_{H_n}(\pi || \pi_n) \qquad \text{[Correction]}$$

where H_n can be updated based on $e_n := \nabla \delta_n(\pi_n) = \nabla l_n(\pi_n) - \nabla \hat{l}_n(\hat{\pi}_n)$ (recall by definition $\nabla l_n(\pi_n) = g_n$ and $\nabla \hat{l}_n(\hat{\pi}_n) = \nabla \hat{l}_n(\pi_n) = \hat{g}_n$). Notice that in the Prediction Step, PICCOLO uses the regularization from the previous Correction Step.

To analyze the performance, we use a lemma of the mirror descent's properties. The proof is a straightforward application of the optimality condition of the proximal map (Nesterov, 2013). We provide a proof here for completeness.

Lemma 4. *Let $\mathcal{K}$ be a convex set. Suppose R is 1-strongly convex with respect to norm $\|\cdot\|$. Let g be a vector in some Euclidean space and let*

$$y = \arg\min_{z \in \mathcal{K}} \langle g, z \rangle + \frac{1}{\eta} B_R(z||x)$$

Then for all $z \in \mathcal{K}$

$$\eta \langle g, y - z \rangle \leq B_R(z||x) - B_R(z||y) - B_R(y||x) \tag{35}$$

which implies

$$\eta \langle g, x - z \rangle \leq B_R(z||x) - B_R(z||y) + \frac{\eta^2}{2} \|g\|_*^2 \tag{36}$$

Proof. Recall the definition $B_R(z||x) = R(z) - R(x) - \langle \nabla R(x), z - x \rangle$. The optimality of the proximal map can be written as

$$\langle \eta g + \nabla R(y) - \nabla R(x), y - z \rangle \leq 0, \qquad \forall z \in \mathcal{K}$$

By rearranging the terms, we can rewrite the above inequality in terms Bregman divergences as follows and derive the first inequality (35):

$$\begin{aligned}
\langle \eta g, y - z \rangle &\leq \langle \nabla R(x) - \nabla R(y), y - z \rangle \\
&= B_R(z||x) - B_R(z||y) + \langle \nabla R(x) - \nabla R(y), y \rangle - \langle \nabla R(x), x \rangle + \langle \nabla R(y), y \rangle + R(x) - R(y) \\
&= B_R(z||x) - B_R(z||y) + \langle \nabla R(x), y - x \rangle + R(x) - R(y) \\
&= B_R(z||x) - B_R(z||y) - B_R(y||x)
\end{aligned}$$

The second inequality is the consequence of (35). First, we rewrite (35) as

$$\langle \eta g, x - z \rangle = B_R(z||x) - B_R(z||y) - B_R(y||x) + \langle \eta g, x - y \rangle$$

Then we use the fact that B_R is 1-strongly convex with respect to $\|\cdot\|$, which implies

$$-B_R(y||x) + \langle \eta g, x - y \rangle \leq -\frac{1}{2}\|x - y\|^2 + \langle \eta g, x - y \rangle \leq \frac{\eta^2}{2} \|g\|_*^2$$

Combining the two inequalities yields (36). ■

Lemma 4 is usually stated with (36), which concerns the decision made before seeing the per-round loss (as in the standard adversarial online learning setting). Here, we additionally concern $\hat{l}_n(\pi_n)$, which is the decision made after seeing $\hat{l}_n$, so we need a tighter bound (35).

Now we show that the regret bound of PICCOLO in the predictable linear problems when the base algorithm is mirror descent.

Proposition 1. *Assume the base algorithm of* PICCOLO *is mirror descent satisfying the Assumption 1. Let $g_n = \nabla l_n(\pi_n)$ and $e_n = g_n - \hat{g}_n$. Then it holds that, for any $\pi \in \Pi$,*

$$\sum_{n=1}^{N} w_n \langle g_n, \pi_n - \pi \rangle \leq M_N + \sum_{n=1}^{N} \frac{w_n^2}{2} \|e_n\|_{*,n}^2 - \frac{1}{2} \|\pi_n - \hat{\pi}_n\|_{n-1}^2$$

Proof. Suppose R_n, which is defined by H_n, is 1-strongly convex with respect to $\|\cdot\|_n$. Then by Lemma 4, we can write, for all $\pi \in \Pi$,

$$\begin{aligned} w_n \langle g_n, \pi_n - \pi \rangle &= w_n \langle \hat{g}_n, \pi_n - \pi \rangle + w_n \langle e_n, \pi_n - \pi \rangle \\ &\le B_{R_{n-1}}(\pi||\hat{\pi}_n) - B_{R_{n-1}}(\pi||\pi_n) - B_{R_{n-1}}(\pi_n||\hat{\pi}_n) \\ &\quad + B_{R_n}(\pi||\pi_n) - B_{R_n}(\pi||\hat{\pi}_{n+1}) + \frac{w_n^2}{2}\|e_n\|_{*,n}^2 \end{aligned} \tag{37}$$

where we use (35) for $\hat{g}_n$ and (36) for the loss e_n.

To show the regret bound of the original (predictable) problem, we first notice that

$$\begin{aligned} &\sum_{n=1}^{N} B_{R_{n-1}}(\pi||\hat{\pi}_n) - B_{R_{n-1}}(\pi||\pi_n) + B_{R_n}(\pi||\pi_n) - B_{R_n}(\pi||\hat{\pi}_{n+1}) \\ &= B_{R_0}(\pi||\hat{\pi}_1) - B_{R_N}(\pi||\hat{\pi}_{N+1}) + \sum_{n=1}^{N} B_{R_{n-1}}(\pi||\hat{\pi}_n) - B_{R_{n-1}}(\pi||\pi_n) + B_{R_n}(\pi||\pi_n) - B_{R_{n-1}}(\pi||\hat{\pi}_n) \\ &= B_{R_0}(\pi||\hat{\pi}_1) - B_{R_N}(\pi||\hat{\pi}_{N+1}) + \sum_{n=1}^{N} B_{R_n}(\pi||\pi_n) - B_{R_{n-1}}(\pi||\pi_n) \le M_N \end{aligned}$$

where the last inequality follows from the assumption on the base algorithm. Therefore, by telescoping the inequality in (37) and using the strong convexity of R_n, we get

$$\begin{aligned} \sum_{n=1}^{N} w_n \langle g_n, \pi_n - \pi \rangle &\le M_N + \sum_{n=1}^{N} \frac{w_n^2}{2}\|e_n\|_{*,n}^2 - B_{R_{n-1}}(\pi_n||\hat{\pi}_n) \\ &\le M_N + \sum_{n=1}^{N} \frac{w_n^2}{2}\|e_n\|_{*,n}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2 \end{aligned}$$

■

F.2.2. Follow-the-Regularized-Leader

We consider another type of base algorithm, FTRL, which is mainly different from mirror descent in the way that constrained decision sets are handled (McMahan, 2017). In this case, the exact update rule of PicCoLO can be written as

$$\pi_n = \arg\min_{\pi \in \Pi} \langle w_n \hat{g}_n, \pi \rangle + \sum_{m=1}^{n-1} \langle w_m g_m, \pi \rangle + B_{r_m}(\pi||\pi_m) \qquad \text{[Prediction]}$$

$$\hat{\pi}_{n+1} = \arg\min_{\pi \in \Pi} \sum_{m=1}^{n} \langle w_m g_m, \pi \rangle + B_{r_m}(\pi||\pi_m) \qquad \text{[Correction]}$$

From the above equations, we verify that MoBIL (Cheng et al., 2019) is indeed a special case of PicCoLO, when the base algorithm is FTRL.

We show PicCoLO with FTRL has the following guarantee.

Proposition 2. *Assume the base algorithm of* PicCoLO *is FTRL satisfying the Assumption 1. Then it holds that, for any* $\pi \in \Pi$*,*

$$\sum_{n=1}^{N} w_n \langle g_n, \pi_n - \pi \rangle \le M_N + \sum_{n=1}^{N} \frac{w_n^2}{2}\|e_n\|_{*,n}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2$$

We show the above results of PicCoLO using a different technique from (Cheng et al., 2019). Instead of developing a specialized proof like they do, we simply use the properties of FTRL on the $2N$-step new adversarial problem!

To do so, we recall some facts of the base algorithm FTRL. First, FTRL in (19) is equivalent to Follow-the-Leader (FTL) on a surrogate problem with the per-round loss is $\langle g_n, \pi \rangle + B_{r_n}(\pi||\pi_n)$. Therefore, the regret of FTRL can be bounded

by the regret of FTL in the surrogate problem plus the size of the additional regularization $B_{r_n}(\pi||\pi_n)$. Second, we recall a standard techniques in proving FTL, called Strong FTL Lemma (see e.g. (McMahan, 2017)), which is proposed for *adversarial* online learning.

Lemma 5 (Strong FTL Lemma (McMahan, 2017)). *For any sequence $\{\pi_n \in \Pi\}$ and $\{l_n\}$,*

$$\textit{Regret}_N(l) \coloneqq \sum_{n=1}^{N} l_n(\pi_n) - \min_{\pi\in\Pi} \sum_{n=1}^{N} l_n(\pi) \le \sum_{n=1}^{N} l_{1:n}(\pi_n) - l_{1:n}(\pi_n^\star)$$

where $\pi_n^\star \in \arg\min_{\pi\in\Pi} l_{1:n}(\pi)$.

Using the decomposition idea above, we show the performance of PICCOLO following sketch below: first, we show a bound on the regret in the surrogate predictable problem with per-round loss $\langle g_n, \pi\rangle + B_{r_n}(\pi||\pi_n)$; second, we derive the bound for the original predictable problem with per-round loss $\langle g_n, \pi\rangle$ by considering the effects of $B_{r_n}(\pi||\pi_n)$. We will prove the first step by applying FTL on the transformed $2N$-step adversarial problem of the original N-step predictable surrogate problem and then showing that PICCOLO achieves the optimal regret in the original N-step predictable surrogate problem. Interestingly, we recover the bound in the stronger FTL Lemma (Lemma 6) by Cheng et al. (2019), which they suggest is necessary for proving the improved regret bound of their FTRL-prediction algorithm (MOBIL).

Lemma 6 (Stronger FTL Lemma (Cheng et al., 2019)). *For any sequence $\{\pi_n\}$ and $\{l_n\}$,*

$$\textit{Regret}_N(l) = \sum_{n=1}^{N} l_{1:n}(\pi_n) - l_{1:n}(\pi_n^\star) - \Delta_n$$

where $\Delta_{n+1} := l_{1:n}(\pi_{n+1}) - l_{1:n}(\pi_n^\star) \ge 0$ and $\pi_n^\star \in \arg\min_{\pi\in\Pi} l_{1:n}(\pi)$.

Our new reduction-based regret bound is presented below.

Proposition 3. *Let $\{l_n\}$ be a predictable loss sequence with predictable information $\{\hat{l}_n\}$. Suppose the decision sequence $\hat{\pi}_1, \pi_1, \hat{\pi}_2, \ldots, \hat{\pi}_N, \pi_N$ is generated by running FTL on the transformed adversarial loss sequence $\hat{l}_1, \delta_1, \hat{l}_2, \ldots, \hat{l}_N, \delta_N$, then the bound in the Stonger FTL Lemma holds. That is, $\textit{Regret}_N(l) \le \sum_{n=1}^{N} l_{1:n}(\pi_n) - l_{1:n}(\pi_n^\star) - \Delta_n$, where $\Delta_{n+1} := l_{1:n}(\pi_{n+1}) - l_{1:n}(\pi_n^\star) \ge 0$ and $\pi_n^\star \in \arg\min_{\pi\in\Pi} l_{1:n}(\pi)$.*

Proof. First, we transform the loss sequence and write

$$\sum_{n=1}^{N} l_n(\pi_n) = \sum_{n=1}^{N} \hat{l}_n(\pi_n) + \delta_n(\pi_n) = \left(\sum_{n=1}^{N} \hat{l}_n(\hat{\pi}_n) + \delta_n(\pi_n)\right) + \left(\sum_{n=1}^{N} \hat{l}_n(\pi_n) - \hat{l}_n(\hat{\pi}_n)\right)$$

Then we apply standard Strong FTL Lemma on the new adversarial problem in the left term.

$$\begin{aligned}
&\sum_{n=1}^{N} \hat{l}_n(\hat{\pi}_n) + \delta_n(\pi_n) \\
&\le \sum_{n=1}^{N} (\hat{l}+\delta)_{1:n}(\pi_n) - \min_{\pi\in\Pi} (\hat{l}+\delta)_{1:n}(\pi) + \sum_{n=1}^{N} ((\hat{l}+\delta)_{1:n-1} + \hat{l}_n)(\hat{\pi}_n) - \min_{\pi\in\Pi} ((\hat{l}+\delta)_{1:n-1} + \hat{l}_n)(\pi) \\
&= \sum_{n=1}^{N} l_{1:n}(\pi_n) - \min_{\pi\in\Pi} l_{1:n}(\pi) + \sum_{n=1}^{N} (l_{1:n-1} + \hat{l}_n)(\hat{\pi}_n) - (l_{1:n-1} + \hat{l}_n)(\pi_n)
\end{aligned}$$

where the first inequality is due to Strong FTL Lemma and the second equality is because FTL update assumption.

Now we observe that if we add the second term above and $\sum_{n=1}^{N} \hat{l}_n(\pi_n) - \hat{l}_n(\hat{\pi}_n)$ together, we have

$$\begin{aligned}
&\sum_{n=1}^{N} (l_{1:n-1} + \hat{l}_n)(\hat{\pi}_n) - (l_{1:n-1} + \hat{l}_n)(\pi_n) + (\hat{l}_n(\pi_n) - \hat{l}_n(\hat{\pi}_n)) \\
&= \sum_{n=1}^{N} (l_{1:n-1})(\hat{\pi}_n) - l_{1:n-1}(\pi_n) = \Delta_n
\end{aligned}$$

Thus, combing previous two inequalities, we have the bound in the Stronger FTL Lemma:

$$\sum_{n=1}^{N} l_n(\pi_n) \leq \sum_{n=1}^{N} l_{1:n}(\pi_n) - \min_{\pi\in\Pi} l_{1:n}(\pi) - \Delta_n \qquad \blacksquare$$

Using Proposition 3, we can now bound the regret of PICCOLO in Proposition 2 easily.

Proof of Proposition 2. Suppose $\sum_{m=1}^{n} B_{r_m}(\cdot||\pi_m)$ is 1-strongly convex with respect to some norm $\|\cdot\|_n$. Let $f_n = \langle w_n g_n, \pi_n\rangle + B_{r_n}(\pi||\pi_m)$. Then by a simple convexity analysis (see e.g. see (McMahan, 2017)) and Proposition 3, we can derive

$$\begin{aligned}\text{Regret}_N(f) &\leq \sum_{n=1}^{N}(f_{1:n}(\pi_n) - \min_{\pi\in\Pi} f_{1:n}(\pi)) - (f_{1:n-1}(\pi_n) - f_{1:n-1}(\hat{\pi}_n)) \\ &\leq \sum_{n=1}^{N} \frac{w_n^2}{2}\|e_n\|_{n,*}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2\end{aligned}$$

Finally, because r_n is proximal (i.e. $B_{r_n}(\pi_n||\pi_n) = 0$), we can bound the original regret: for any $\pi \in \Pi$, it satisfies that

$$\begin{aligned}\sum_{n=1}^{N} w_n \langle g_n, \pi_n - \pi\rangle &\leq \sum_{n=1}^{N} f_n(\pi_n) - f_n(\pi) + B_{r_n}(\pi||\pi_n) \\ &\leq M_N + \sum_{n=1}^{N} \frac{w_n^2}{2}\|e_n\|_{*,n}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2\end{aligned}$$

where we use Assumption 1 and the bound of $\text{Regret}_N(f)$ in the second inequality. $\blacksquare$

G. Policy Optimization Analysis of PICCOLO

In this section, we discuss how to interpret the bound given in Theorem 1

$$\sum_{n=1}^{N} w_n \langle g_n, \pi_n - \pi\rangle \leq M_N + \sum_{n=1}^{N} \frac{w_n^2}{2}\|e_n\|_{*,n}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2$$

in the context of policy optimization and show exactly how the optimal bound

$$\mathbb{E}\left[\sum_{n=1}^{N} \langle w_n g_n, \pi_n - \pi\rangle\right] \leq O(1) + C_{\Pi,\Phi}\frac{w_{1:N}}{\sqrt{N}} \tag{38}$$

is derived. We will discuss how model learning can further help minimize the regret bound later in Appendix G.4.

G.1. Assumptions

We introduce some assumptions to characterize the sampled gradient g_n. Recall $g_n = \nabla\tilde{l}_n(\pi_n)$.

Assumption 2. $\|\mathbb{E}[g_n]\|_*^2 \leq G_g^2$ and $\|g_n - \mathbb{E}[g_n]\|_*^2 \leq \sigma_g^2$ for some finite constants G_g and σ_g.

Similarly, we consider properties of the predictive model Φ_n that is used to estimate the gradient of the next per-round loss. Let $\mathcal{P}$ denote the class of these models (i.e. $\Phi_n \in \mathcal{P}$), which can potentially be *stochastic*. We make assumptions on the size of $\hat{g}_n$ and its variance.

Assumption 3. $\|\mathbb{E}[\hat{g}_n]\|_*^2 \leq G_{\hat{g}}^2$ and $\mathbb{E}[\|\hat{g}_n - \mathbb{E}[\hat{g}_n]\|_*^2] \leq \sigma_{\hat{g}}^2$ for some finite constants $G_{\hat{g}}$ and $\sigma_{\hat{g}}$.

Additionally, we assume these models are Lipschitz continuous.

Assumption 4. There is a constant $L \in [0, \infty)$ such that, for any instantaneous cost ψ and any $\Phi \in \mathcal{P}$, it satisfies $\|\mathbb{E}[\Phi(\pi)] - \mathbb{E}[\Phi(\pi')]\|_* \le L\|\pi - \pi'\|$.

Lastly, as PICCOLO is agnostic to the base algorithm, we assume the local norm $\|\cdot\|_n$ chosen by the base algorithm at round n satisfies $\|\cdot\|_n^2 \ge \alpha_n \|\cdot\|^2$ for some $\alpha_n > 0$. This condition implies that $\|\cdot\|_{n,*}^2 \le \frac{1}{\alpha_n}\|\cdot\|_*^2$. In addition, we assume α_n is non-decreasing so that $M_N = O(\alpha_N)$ in Assumption 1, where the leading constant in the bound $O(\alpha_N)$ is proportional to $|\Pi|$, as commonly chosen in online convex optimization.

G.2. A Useful Lemma

We study the bound in Theorem 1 under the assumptions made in the previous section. We first derive a basic inequality, following the idea in (Cheng et al., 2019, Lemma 4.3).

Lemma 7. *Under Assumptions 2, 3, and 4, it holds*

$$\mathbb{E}[\|e_n\|_{*,n}^2] = \mathbb{E}[\|g_n - \hat{g}_n\|_{*,n}^2] \le \frac{4}{\alpha_n}\left(\sigma_g^2 + \sigma_{\hat{g}}^2 + L_n^2\|\pi_n - \hat{\pi}_n\|_n^2 + E_n(\Phi_n)\right)$$

where $E_n(\Phi_n) = \|\mathbb{E}[g_n] - \mathbb{E}[\Phi_n(\pi_n, \psi_n)]\|_*^2$ is the prediction error of model Φ_n.

Proof. Recall $\hat{g}_n = \Phi_n(\hat{\pi}_n, \psi_n)$. Using the triangular inequality, we can simply derive

$$\begin{aligned}
&\mathbb{E}[\|g_n - \hat{g}_n\|_{*,n}^2] \\
&\le 4\left(\mathbb{E}[\|g_n - \mathbb{E}[g_n]\|_{*,n}^2] + \|\mathbb{E}[g_n] - \mathbb{E}[\Phi_n(\pi_n, \psi_n)]\|_{*,n}^2 + \|\mathbb{E}[\Phi_n(\pi_n, \psi_n)] - \mathbb{E}[\hat{g}_n]\|_{*,n}^2 + \mathbb{E}[\|\mathbb{E}[\hat{g}_n] - \hat{g}_n\|_{*,n}^2]\right) \\
&= 4\left(\mathbb{E}[\|g_n - \mathbb{E}[g_n]\|_{*,n}^2] + \|\mathbb{E}[g_n] - \mathbb{E}[\Phi_n(\pi_n, \psi_n)]\|_{*,n}^2 + \|\mathbb{E}[\Phi_n(\pi_n, \psi_n)] - \mathbb{E}[\Phi_n(\hat{\pi}_n, \psi_n)]\|_{*,n}^2 + \mathbb{E}[\|\mathbb{E}[\hat{g}_n] - \hat{g}_n\|_{*,n}^2]\right) \\
&\le 4\left(\frac{1}{\alpha_n}\sigma_g^2 + \frac{1}{\alpha_n}E_n(\Phi_n) + \|\mathbb{E}[\Phi_n(\pi_n, \psi_n)] - \mathbb{E}[\Phi_n(\hat{\pi}_n, \psi_n)]\|_{*,n}^2 + \frac{1}{\alpha_n}\sigma_{\hat{g}}^2\right) \\
&\le \frac{4}{\alpha_n}\left(\sigma_g^2 + \sigma_{\hat{g}}^2 + L^2\|\pi_n - \hat{\pi}_n\|_n^2 + E_n(\Phi_n)\right)
\end{aligned}$$

where the last inequality is due to Assumption 4. ∎

G.3. Optimal Regret Bounds

We now analyze the regret bound in Theorem 1

$$\sum_{n=1}^N w_n \langle g_n, \pi_n - \pi\rangle \le M_N + \sum_{n=1}^N \frac{w_n^2}{2}\|e_n\|_{*,n}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2 \tag{39}$$

We first gain some intuition about the size of

$$M_N + \mathbb{E}\left[\sum_{n=1}^N \frac{w_n^2}{2}\|e_n\|_{*,n}^2\right]. \tag{40}$$

Because when $\texttt{adapt}(h_n, H_{n-1}, e_n, w_n)$ is called in the Correction Step in (28) with the error gradient e_n as input, an optimal base algorithm (e.g. all the base algorithms listed in Appendix C) would choose a local norm sequence $\|\cdot\|_n$ such that (40) is optimal. For example, suppose $\|e_n\|_*^2 = O(1)$ and $w_n = n^p$ for some $p > -1$. If the base algorithm is basic mirror descent (cf. Appendix C), then $\alpha_n = O(\frac{w_{1:n}}{\sqrt{n}})$. By our assumption that $M_N = O(\alpha_N)$, it implies (40) can be upper bounded by

$$\begin{aligned}
M_N + \mathbb{E}\left[\sum_{n=1}^N \frac{w_n^2}{2}\|e_n\|_{*,n}^2\right] &\le O\left(\frac{w_{1:N}}{\sqrt{N}}\right) + \left[\sum_{n=1}^N \frac{w_n^2\sqrt{n}}{2w_{1:n}}\|e_n\|_*^2\right] \\
&\le O\left(\frac{w_{1:N}}{\sqrt{N}} + \sum_{n=1}^N \frac{w_n^2\sqrt{n}}{w_{1:n}}\right) = O\left(N^{p+1/2}\right)
\end{aligned}$$

which will lead to an optimal weighted average regret in $O(\frac{1}{\sqrt{N}})$.

PICCOLO actually has a better regret than the simplified case discussed above, because of the negative term $-\frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2$ in (39). To see its effects, we combine Lemma 7 with (39) to reveal some insights:

$$\mathbb{E}\left[\sum_{n=1}^{N} w_n \langle g_n, \pi_n - \pi\rangle\right] \leq O(\alpha_N) + \mathbb{E}\left[\sum_{n=1}^{N} \frac{w_n^2}{2}\|e_n\|_{*,n}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2\right] \tag{41}$$

$$\leq O(\alpha_N) + \mathbb{E}\left[\sum_{n=1}^{N} \frac{2w_n^2}{\alpha_n}\left(\sigma_g^2 + \sigma_{\hat{g}}^2 + L^2\|\pi_n - \hat{\pi}_n\|_n^2 + E_n(\Phi_n)\right) - \frac{\alpha_{n-1}}{2}\|\pi_n - \hat{\pi}_n\|^2\right]$$

$$= \left(O(\alpha_N) + \mathbb{E}\left[\sum_{n=1}^{N} \frac{2w_n^2}{\alpha_n}\left(\sigma_g^2 + \sigma_{\hat{g}}^2 + E_n(\Phi_n)\right)\right]\right) + \left(\mathbb{E}\left[\sum_{n=1}^{N}(\frac{2w_n^2}{\alpha_n}L^2 - \frac{\alpha_{n-1}}{2})\|\pi_n - \hat{\pi}_n\|^2\right]\right) \tag{42}$$

The first term in (42) plays the same role as (40); when the base algorithm has an optimal `adapt` operation and $w_n = n^p$ for some $p > -1$, it would be in $O\left(N^{p+1/2}\right)$. Here we see that the constant factor in this bound is proportional to $\sigma_g^2 + \sigma_{\hat{g}}^2 + E_n(\Phi_n)$. Therefore, if the variances σ_g^2, $\sigma_{\hat{g}}^2$ of the gradients are small, the regret would mainly depend on the prediction error $E_n(\Phi_n)$ of Φ_n. In the next section (Appendix G.4), we will show that when Φ_n is learned online (as the authors in (Cheng et al., 2019) suggest), on average the regret is close to the regret of using the best model in the hindsight. The second term in (42) contributes to $O(1)$ in the regret, when the base algorithm adapts properly to w_n. For example, when $\alpha_n = \Theta(\frac{w_{1:n}}{\sqrt{n}})$ and $w_n = n^p$ for some $p > -1$, then

$$\sum_{n=1}^{N} \frac{2w_n^2}{\alpha_n}L^2 - \frac{\alpha_{n-1}}{2} = \sum_{n=1}^{N} O(n^{p-1/2} - n^{p+1/2}) = O(1)$$

In addition, because $\|\pi_n - \hat{\pi}_n\|$ would converge to zero, the effects of the second term in (42) becomes even minor.

In summary, for a reasonable base algorithm and $w_n = n^p$ with $p > -1$, running PICCOLO has the regret bound

$$\mathbb{E}\left[\sum_{n=1}^{N} w_n \langle g_n, \pi_n - \pi\rangle\right] = O(\alpha_N) + O\left(\frac{w_{1:N}}{\sqrt{N}}(\sigma_g^2 + \sigma_{\hat{g}}^2)\right) + O(1) + \mathbb{E}\left[\sum_{n=1}^{N} \frac{2w_n^2}{\alpha_n}E_n(\Phi_n)\right] \tag{43}$$

Suppose $\alpha_n = \Theta(|\Pi|\frac{w_{1:n}}{\sqrt{n}})$ and $w_n = n^p$ for some $p > -1$, This implies the inequality

$$\mathbb{E}\left[\sum_{n=1}^{N} \langle w_n g_n, \pi_n - \pi\rangle\right] \leq O(1) + C_{\Pi,\Phi}\frac{w_{1:N}}{\sqrt{N}} \tag{38}$$

where $C_{\Pi,\Phi} = O(|\Pi| + \sigma_g^2 + \sigma_{\hat{g}}^2 + \sup_n E_n(\Phi_n))$. The use of non-uniform weights can lead to a faster on average decay of the standing $O(1)$ term in the final weighted average regret bound, i.e.

$$\frac{1}{w_{1:N}}\mathbb{E}\left[\sum_{n=1}^{N} \langle w_n g_n, \pi_n - \pi\rangle\right] \leq O\left(\frac{1}{w_{1:N}}\right) + \frac{C_{\Pi,\Phi}}{\sqrt{N}}$$

In general, the authors in (Cheng et al., 2018; 2019) recommend using $p \ll N$ (e.g. in the range of $[0, 5]$) to remove the undesirable constant factor, yet without introducing large multiplicative constant factor.

G.4. Model Learning

The regret bound in (43) reveals an important factor that is due to the prediction error $\mathbb{E}\left[\sum_{n=1}^{N} \frac{2w_n^2}{\alpha_n}E_n(\Phi_n)\right]$, where we recall $E_n(\Phi_n) = \|\mathbb{E}[g_n] - \mathbb{E}[\Phi_n(\pi_n)]\|_*^2$. Cheng et al. (2019) show that, to minimize this error sum through model learning, a secondary online learning problem with per-round loss $E_n(\cdot)$ can be considered. Note that this is a standard weighted adversarial online learning problem (weighted by $\frac{2w_n^2}{\alpha_n}$), because $E_n(\cdot)$ is revealed after one commits to using model Φ_n.

While in implementation the exact function $E_n(\cdot)$ is unavailable (as it requires infinite data), we can adopt an unbiased upper bound. For example, Cheng et al. (2019) show that $E_n(\cdot)$ can be upper bounded by the single- or multi-step prediction error of a transition dynamics model. More generally, we can learn a neural network to minimize the gradient prediction error directly. As long as this secondary online learning problem is solved by a no-regret algorithm, the error due to online model learning would contribute a term in $O(w_{1:N}\epsilon_{\mathcal{P},N}/\sqrt{N}) + o(w_{1:N}/\sqrt{N})$ in (43), where $\epsilon_{\mathcal{P},N}$ is the minimal error achieved by the best model in the model class $\mathcal{P}$ (see (Cheng et al., 2019) for details).

H. Experimental Details

H.1. Algorithms

Base Algorithms In the experiments, we consider three commonly used first-order online learning algorithms: ADAM, NATGRAD, and TRPO, all of which adapt the regularization online to alleviate the burden of learning rate tuning. We provide the decomposition of ADAM into the basic three operations in Appendix C, and that of NATGRAD in Appendix E. In particular, the adaptivity of NATGRAD is achieved by adjusting the step size based on a moving average of the dual norm of the gradient. TRPO adjusts the step size to minimize a given cost function (here it is a linear function defined by the first-order oracle) within a pre-specified KL divergence centered at the current decision. While greedily changing the step size in every iteration makes TRPO an inappropriate candidate for adversarial online learning. Nonetheless, it can still be written in the form of mirror descent and allows a decomposition using the three basic operators; its `adapt` operator can be defined as the process of finding the maximal scalar step along the natural gradient direction such that the updated decision stays within the trust region. For all the algorithms, a decaying step size multiplier in the form $\eta/(1+\alpha\sqrt{n})$ is also used; for TRPO, it is used to specify the size of trust regions. The values chosen for the hyperparameters η and α can be found in Table 2. To the best of our knowledge, the conversion of these approaches into unbiased model-based algorithms is novel.

Reinforcement Learning Per-round Loss In iteration n, in order to compute the online gradient (5), GAE (Schulman et al., 2016) is used to estimate the advantage function $A_{\pi_{n-1}}$. More concretely, this advantage estimate utilizes an estimate of value function $V_{\pi_{n-1}}$ (which we denote $\hat{V}_{\pi_{n-1}}$) and on-policy samples. We chosen $\lambda = 0.98$ in GAE to reduce influence of the error in $V_{\pi_{n-1}}$, which can be catastrophic. Importance sampling can be used to estimate $A_{\pi_{n-1}}$ in order to leverage data that are collected on-policy by running π_n. However, since we select a large λ, importance sampling can lead to vanishing importance weights, making the gradient extremely noisy. Therefore, in the experiments, importance sampling is not applied.

Gradient Computation and Control Variate The gradients are computed using likelihood-ratio trick and the associated advantage function estimates described above. A scalar control variate is further used to reduce the variance of the sampled gradient, which is set to the mean of the advantage estimates evaluated on newly collected data.

Policies and Value Networks Simple feed-forward neural networks are used to construct all of the function approximators (policy and value function) in the tasks. They have 1 hidden layer with 32 $\tanh$ units for all policy networks, and have 2 hidden layers with 64 $\tanh$ units for value function networks. Gaussian stochastic policies are considered, i.e., for any state $s \in \mathbb{S}$, π_s is Gaussian, and the mean of π_s is modeled by the policy network, whereas the diagonal covariance matrix is state independent (which is also learned). Initial value of $\log\sigma$ of the Gaussian policies -1.0, the standard deviation for initializing the output layer is 0.01, and the standard deviation for initialization hidden layer is 1.0. After the policy update, a new value function estimate $\hat{V}_{\pi_n}$ is computed by minimizing the mean of squared difference between $\hat{V}_{\pi_n}$ and $\hat{V}_{\pi_{n-1}} + \hat{A}_{\pi_n}$, where $\hat{A}_{\pi_n}$ is the GAE estimate using $\hat{V}_{\pi_{n-1}}$ and $\lambda = 0.98$, through ADAM with batch size 128, number of batches 2048, and learning rate 0.001. Value function is pretrained using examples collected by executing the randomly initialized policy.

Computing Model Gradients We compute $\hat{g}_n$ in two ways. The first approach is to use the simple heuristic that sets $\hat{g}_n = \Phi_n(\hat{\pi}_n)$, where Φ_n is some predictive models depending on the exact experimental setup. The second approach is to use the fixed-point formulation (8). This is realized by solving the equivalent optimization problem mentioned in the paper. In implementation, we only solves this problem approximately using some finite number of gradient steps; though this is insufficient to yield a stationary point as desired in the theory, we experimentally find that it is sufficient to yield improvement over the heuristic $\hat{g}_n = \Phi_n(\hat{\pi}_n)$.

Approximate Solution to Fixed-Point Problems of PICCOLO PICCOLO relies on the predicted gradient $\hat{g}_n$ in the Prediction Step. Recall ideally we wish to solve the fixed-point problem that finds h_n^* such that

$$h_n^* = \texttt{update}(\hat{h}_n, H_{n-1}, \Phi_n(\pi_n(h_n^*)), w_n) \tag{44}$$

and then apply $\hat{g}_n = \Phi_n(\pi_n(h_n^*))$ in the Prediction Step to get h_n, i.e.,

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

Because h_n^* is the solution to the fixed-point problem, we have $h_n = h_n^*$. Such choice of $\hat{g}_n$ will fully leverage the information provided by Φ_n, as it does not induce additional linearization due to evaluating Φ_n at points different from h_n.

Exactly solving the fixed-point problem is difficult. In the experiments, we adopt a heuristic which computes an approximation to h_n^* as follows. We suppose $\Phi_n = \nabla f_n$ for some function f_n, which is the case e.g. when Φ_n is the simulated gradient based on some (biased) dynamics model. This restriction makes the fixed-point problem as finding a stationary point of the optimization problem $\min_{\pi \in \Pi} f_n(\pi) + B_{R_{n-1}}(\pi || \hat{\pi}_n)$. In implementation, we initialize the iterate in this subproblem as $\texttt{update}(\hat{h}_n, H_{n-1}, \Phi_n(\hat{\pi}_n), w_n)$, which is the output of the Prediction Step if we were to use $\hat{g}_n = \Phi_n(\hat{\pi}_n)$. We made this choice in initializing the subproblem, as we know that using $\hat{g}_n = \Phi_n(\hat{\pi}_n)$ in PICCOLO already works well (see the experiments) and it can be viewed as the solution to the fixed-point problem with respect to the linearized version of Φ_n at $\hat{\pi}_n$. Given the this initialization point, we proceed to compute the approximate solution to the fixed-point by applying the given base algorithm for 5 iterations and then return the last iterate as the approximate solution. For example, if the base algorithm is natural gradient descent, we fixed the Bregman divergence (i.e. its the Fisher information matrix as $\hat{\pi}_n$) and only updated the scalar stepsize adaptively along with the policy in solving this *regularized model-based RL problem* (i.e. $\min_{\pi \in \Pi} f_n(\pi) + B_{R_{n-1}}(\pi || \hat{\pi}_n)$). While such simple implementation is not ideal, we found it works in practice, though we acknowledge that a better implementation of the subproblem solver would improve the results.

H.2. Tasks

The robotic control tasks that are considered in the experiments are CartPole, Hopper, Snake, and Walker3D from OpenAI Gym (Brockman et al., 2016) with the DART physics engine (Lee et al., 2018)[14]. CartPole is a classic control problem, and its goal is to keep a pole balanced in a upright posture, by only applying force to the cart. Hopper, Snake, and Walker3D are locomotion tasks, of which the goal is to control an agent to move forward as quickly as possible without falling down (for Hopper and Walker3D) or deviating too much from moving forward (for Snake). Hopper is monopedal and Walker3D is bipedal, and both of them are subjected to significant contact discontinuities that are hard or even impossible to predict.

H.3. Full Experimental Results

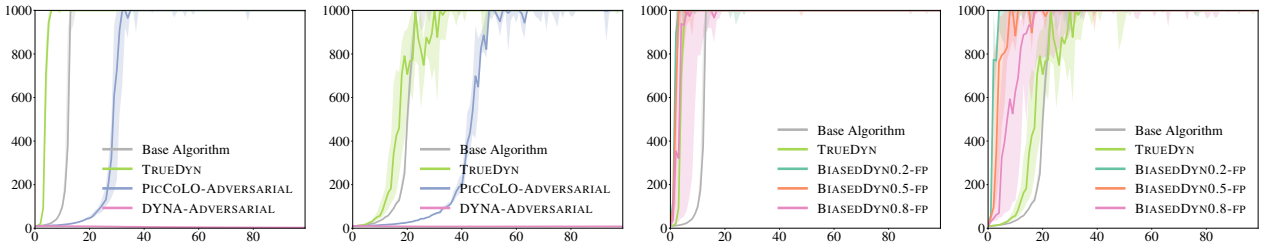
In Figure 3, we empirically study the properties of PICCOLO that are predicted by theory on CartPole environment. In Figure 4, we "PICCOLO " three base algorithms: ADAM, NATGRAD, TRPO, and apply them on four simulated environments: Cartpole, Hopper, Snake, and Walker3D.

H.4. Experiment Hyperparameters

The hyperparameters used in the experiments and the basic attributes of the environments are detailed in Table 2.

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

[15] α and η appear in the decaying step size multiplier for all the algorithms in the form $\eta/(1+\alpha\sqrt{n})$. α influences how fast the step size decays. We chose α in the experiments based on the number of iterations.

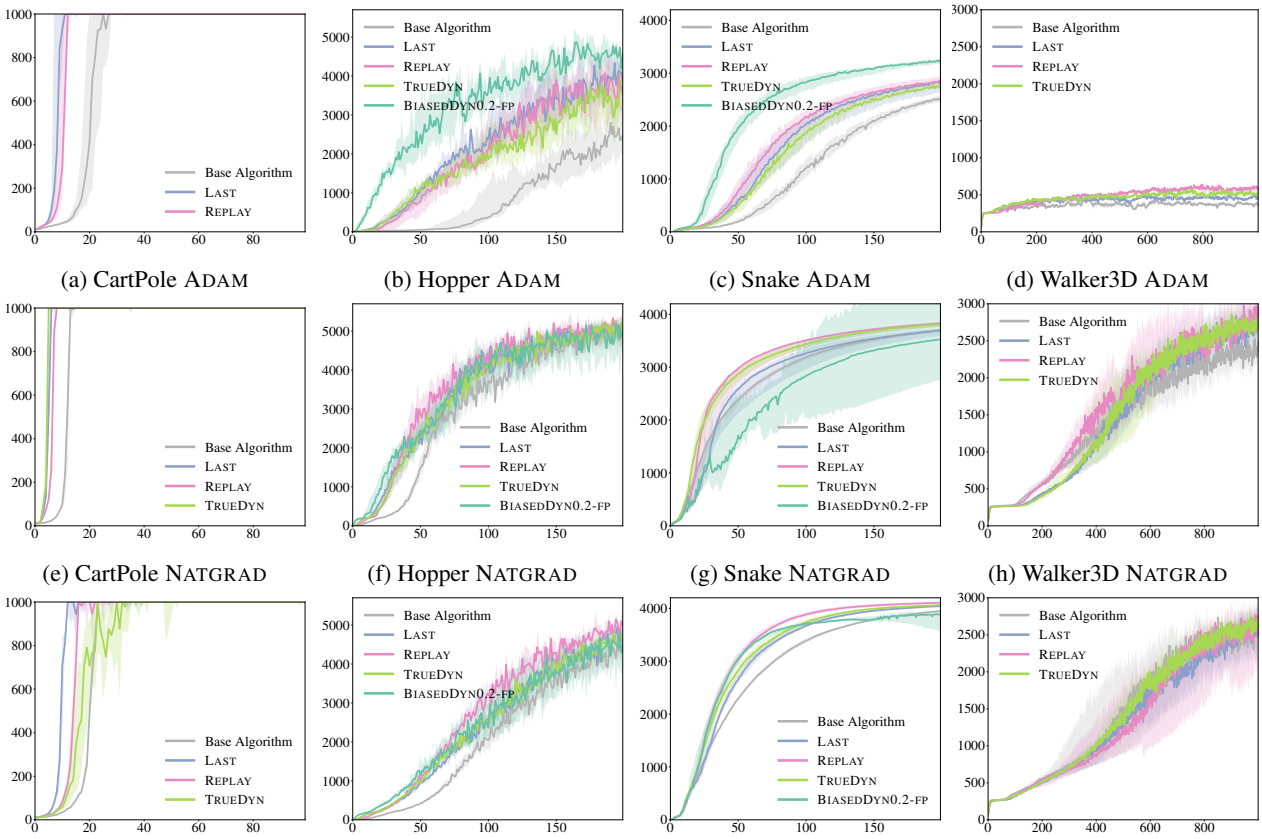


(a) Adv. model, NATGRAD (b) Adv. model, TRPO (c) Diff. fidelity, NATGRAD (d) Diff. fidelity, TRPO

Figure 3: Performance of PICCOLO with different predictive models on CartPole. 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. The update rule, by default, is PICCOLO. For example TRUEDYN in (a) refers to PICCOLO with TRUEDYN predictive model. (a), (b): Comparison of PICCOLO and DYNA with adversarial model using NATGRAD and TRPO as base algorithms. (c), (d): 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 probablity 0.5 respectively.

	CartPole	Hopper	Snake	Walker3D
Observation space dimension	4	11	17	41
Action space dimension	1	3	6	15
State space dimension	4	12	18	42
Number of samples from env. per iteration	4k	16k	16k	32k
Number of samples from model dyn. per iteration	4k	16k	16k	32k
Length of horizon	1,000	1,000	1,000	1,000
Number of iterations	100	200	200	1,000
Number of iterations of samples for REPLAY buffer	5	4	3	2 (3 for ADAM)
α [15]	0.1	0.1	0.1	0.01
η in ADAM	0.005	0.005	0.002	0.01
η in NATGRAD	0.05	0.05	0.2	0.2
η in TRPO	0.002	0.002	0.01	0.04

Table 2: Tasks specifics and hyperparameters.



(a) CartPole ADAM (b) Hopper ADAM (c) Snake ADAM (d) Walker3D ADAM

(e) CartPole NATGRAD (f) Hopper NATGRAD (g) Snake NATGRAD (h) Walker3D NATGRAD

(i) CartPole TRPO (j) Hopper TRPO (k) Snake TRPO (l) Walker3D TRPO

Figure 4: The performance of PICCOLO with different predictive models on various tasks, compared to base algorithms. The rows use ADAM, NATGRAD, and TRPO as the base algorithms, respectively. 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.