Proceedings of Machine Learning Research vol 107:1–23, 2020

Borrowing From the Future: An Attempt to Address Double Sampling

Yuhua Zhu YUHUAZHU@STANFORD.EDU
Department of Mathematics, Stanford University, Stanford, CA 94305

Lexing Ying LEXING@STANFORD.EDU
Department of Mathematics and ICME, Stanford University, Stanford, CA 94305

Abstract

For model-free reinforcement learning, one of the main challenges of stochastic Bellman residual minimization is the double sampling problem, i.e., while only one single sample for the next state is available in the model-free setting, two independent samples for the next state are required in order to perform unbiased stochastic gradient descent. We propose new algorithms for addressing this problem based on the idea of borrowing extra randomness from the future. When the transition kernel varies slowly with respect to the state, it is shown that the training trajectory of new algorithms is close to the one of unbiased stochastic gradient descent. Numerical results for policy evaluation in both tabular and neural network settings are provided to confirm the theoretical findings.

Keywords: Reinforcement learning; Policy evaluation; Double sampling; Bellman residual minimization; Stochastic gradient descent.

1. Introduction

Reinforcement learning (RL) has received a lot of attention following the recent successes, such as AlphaGo and AlphaZero (Silver et al., 2016, 2017). At the center of RL is the problem of optimizing Markov decision process (MDP), i.e., finding the optimal policy that maximizes the return (Sutton and Barto, 2018). As a type of learning with minimal or no supervision, RL can be more powerful than supervised learning and is often considered to be closer to the natural learning process. On the other hand, as an optimization and control problem, RL is also significantly harder with many practical challenges, such as long trajectories required for convergence, high dimensional continuous state and action spaces, learning with limited and noisy samples, etc.

Background. This paper considers policy evaluation, also known as prediction, which is one of the most basic problems of RL. In model-based RL, especially with small to medium-sized state space, value iteration is commonly used in practice as it guarantees the convergence (Bertsekas and Tsitsiklis, 1996). In model-free RL, the temporal difference (TD) algorithm (Sutton, 1988) converges in the tabular setting as well as with linear approximation. However, the stability and convergence of TD are not guaranteed when the nonlinear approximation is used (Boyan and Moore, 1995; Baird, 1995; Tsitsiklis and Van Roy, 1997). With the recent development and empirical successes of deep neural networks (DNNs), it becomes even more important to understand and stabilize nonlinear approximations.

One direction for stabilizing the nonlinear approximation is to formulate the policy evaluation as a minimization problem rather than a fixed-point iteration; one such example is the Bellman residual minimization (BRM) (Baird, 1995), which is sometimes also called Bellman error minimization in the literature. However, BRM suffers from the so-called double sampling problem, i.e., at a

© 2020 Y. Zhu & L. Ying.

given state, two independent samples for the next state are required to perform unbiased stochastic gradient descent (SGD). Such a requirement is often hard to fulfill in a model-free setting, especially for continuous state space.

Related work. One way to avoid the double sampling problem in BRM is to consider the primal-dual representation. This method turns the task into finding a saddle point of a minimax problem. Such a method includes GTD (Sutton et al., 2008, 2009) and its variants (Bhatnagar et al., 2009; Maei et al., 2010; Dai et al., 2017; Liu et al., 2015). However, the minimax problem can be less stable than the original minimization problem when the maximum is taken over a non-concave function. Therefore, a direct application of the primal-dual method to RL problem with nonlinear function approximations and continuous state space might result in suboptimal performance (see Section 2.2 and Figure 2 for details.)

Another approach for BRM chooses to solve the minimization problem directly. The stochastic compositional gradient method (SCGD), proposed in (Wang et al., 2017, 2016), is a two time-scale algorithm that addresses the double sampling problem in minimizing a function in the form of two nested expectations. When dealing with the continuous state space, the performance of the SCGD method is less clear because the minimizer SCGD found is not necessarily the fixed point of the Bellman operator (see Section 2.2 for details.)

Contributions. In this paper, we revisit the Bellman residual minimization and develop two algorithms to alleviate the double sampling problem. The key idea of the new algorithms is to *borrow extra randomness from the future*. When the transition kernel varies slowly with respect to the state, we show that the training trajectories of the proposed algorithms are statistically close to the one of the unbiased SGD. The proposed algorithms are applied to the prediction problem (i.e., policy evaluation) in both the tabular and neural network settings to confirm the theoretical findings. We also show that, for continuous state space, our method results in better performance than the primal-dual method and SCGD. Though the discussion here focuses on policy evaluation, the same techniques can be extended to Q-Learning (Watkins, 1989) or value iteration.

Organization. The rest of this paper is organized as follows. Section 2 introduces the key idea of the proposed algorithms and a summary of the paper's main results. A discussion of related work is given in Section 2.2. Section 3 gives the details of the proposed algorithms. Section 4 bounds the errors between the new algorithm and the accurate but unrealistic *uncorrelated sampling* algorithm. Numerical results are given in Section 5 to confirm the theoretical findings and demonstrate the efficiency of the new algorithms.

2. Models and key ideas

This paper considers both continuous and discrete state space models.

- In the *continuous state space* setting, we consider a discrete-time Markov decision process (MDP) with continuous state space. The state space $\mathbb{S} \subset \mathbb{R}^{d_s}$ is a compact set. Given a fixed policy, we assume that the one-step transition $P(s, s')$ is prescribed by an unknown drift $\alpha(\cdot)$ and an unknown diffusion $\sigma(\cdot)$,

$$s_{m+1} = s_m + \alpha(s_m)\epsilon + \sigma(s_m)\sqrt{\epsilon}Z_m, \quad Z_m \sim \text{Normal}(0, I_{d_s \times d_s}). \tag{2.1}$$

When the state reaches the boundary, it follows a prescribed boundary condition. Here, we choose to work with a stochastic differential equation (SDE) setting to simplify the presentation of the algorithms and the theorems. The scalings ϵ and $\sqrt{\epsilon}$ of the drift and the noise terms correspond to discretizing the SDE with the time step ϵ. However, both the algorithms and theorems can be extended to more general cases. The real-valued immediate reward is denoted by $r(s, s')$ for $s, s' \in \mathbb{S}$, and the discount factor γ is in $(0, 1)$.

- In the *discrete state space* setting, $\mathbb{S} = \{0, 1, \ldots, n-1\}$. Given a policy, the transition matrix $P(s, s')$ that represents the probability from state s to s'

$$P(s, s') = P(s_{m+1} = s' | s_m = s)$$

is assumed to vary slowly in both s and s'. The function $r(s, s')$ denotes the immediate reward function from state s to s', under the current policy. Finally, the discount rate γ is between 0 and 1.

Since we only consider the prediction problem here, the policy is considered fixed for the rest of the paper. With these notations, the value function $V(s)$ is the expected discounted return if the policy is followed from state s,

$$V(s) = \mathbb{E}\left[\sum_{m=0}^{\infty} \gamma^t r(s_m, s_{m+1}) | s_0 = s\right]. \tag{2.2}$$

Let $R(s) = \mathbb{E}[r(s_m, s_{m+1})|s_m = s]$ be the immediate reward under the fixed policy, and $\mathbb{T}$ be the Bellman operator defined as

$$\mathbb{T}V(s) = R(s) + \gamma \mathbb{E}[V(s_{m+1})|s_m = s]. \tag{2.3}$$

The value function $V(s)$ is the fixed point of the operator $\mathbb{T}$.

2.1. Main ideas and results

Let us consider approximating the value function in a parameterized form $V(s, \theta)$ in model-free RL. Here, the value function approximation can either be linear or nonlinear with respect to the parameter θ. One way for computing the optimal parameter θ^* is to perform gradient descent to the so-called mean-square Bellman residual

$$\min_{\theta} \ \mathbb{E}\left(\mathbb{E}\left[R(s_m) + \gamma V(s_{m+1}; \theta) - V(s_m; \theta) | s_m\right]\right)^2. \tag{2.4}$$

This approach is thus called the Bellman residual minimization (BRM) in the literature. The stochastic gradient descent of BRM is based on an unbiased gradient estimation of the objective function (2.4), which requires two independent transitions from s_m to s_{m+1}. However, for model-free RL, one does not know explicitly how the environment interacts with the agent. That is to say, besides observing the agent's trajectory $\{s_m\}_{m=0}^T$ under the given policy, one cannot generate s_{m+1} from s_m because one does not know the drift and diffusion in (2.1) explicitly. Since the trajectory $\{s_m\}_{m=0}^T$ provides only one simulation from s_m to s_{m+1}, there is no direct means to generate the second copy. This is the so-called double sampling issue.

The main idea of this paper is to alleviate the double sampling issue by *borrowing randomness from the future*: instead of requiring a new copy of s'_{m+1} starting from s_m, we approximate it using the difference between s_{m+2} and s_{m+1}

$$s'_{m+1} \approx s_m + (s_{m+2} - s_{m+1}).$$

When the derivatives of the drift and diffusion terms are under control, the difference between Δs_m and Δs_{m+1} is small, which makes the new s'_{m+1} statistically close to the distribution of the true next state. We refer to the algorithm based on this idea the BFF algorithm, where BFF is short for "borrowing from the future".

Before diving into the algorithmic details, let us first summarize the main properties of the BFF algorithm. When the derivatives of the implicit drift and diffusion terms are small, we are able to show, in the continuous state space setting,

- The difference between the biased objective function in the BFF algorithm and the true objective function is only $O(\epsilon^2)$ (Lemma 1);
- The equilibrium distribution of θ of the BFF algorithm differs from the unbiased SGD within an error of order $O(\frac{\epsilon^2}{\eta})$ (Theorem 3);
- The evolution of θ of the BFF algorithm differs statistically from the unbiased SGD only within an error of $O(1 + \frac{\epsilon^2}{\eta})O(\epsilon^2)$ (Theorem 4).

Here η is the ratio of the learning rate over the batch size. Note that in order to have the error under control, η cannot be too small. Intuitively, this is because: the algorithm minimizes a slightly biased objective function, so if the optimization is done without any randomness, the solution will have little overlap with the exact one.

Although the theoretical results only concern with the first case, we verify numerically in Section 5 that the proposed algorithms perform well also in the discrete state space setting.

2.2. Discussion on related work

There are two other ways commonly used to solve the BRM problem. One is the SCGD method proposed in (Wang et al., 2016), the goal of which is to solve the minimization problem in the form of

$$\min_\theta \mathbb{E}_v[f_v(\mathbb{E}_\omega[g_\omega(\theta)])].$$

When $V(s;\theta) : \mathbb{S} \to \mathbb{R}^{|\mathbb{S}|}$ is in the vector form, one could set $g_{s_{m+1}} = R(s_m) + \gamma V(s_{m+1}) - V(s_m)$ as a vector function in $\mathbb{R}^{|\mathbb{S}|}$, and $f = \sum_i g_i^2$. Then the above minimization problem automatically becomes

$$\min_\theta \mathbb{E}\left|\mathbb{E}[R(s_m) + \gamma V(s_{m+1}) - V(s_m)|s_m]\right|^2,$$

which gives the fixed point of the Bellman operator $\mathbb{T}$. According to Algorithm 1 in (Wang et al., 2017), the basic SCGD updates the parameter θ in the following way,

$$\begin{aligned} y_{k+1} &= y_k + \beta\left(R(s_m) + \gamma V(s_{m+1};\theta_k) - V(s_m;\theta_k) - y_k\right); \\ \theta_{k+1} &= \theta_k - \tau\left(\gamma \nabla_\theta V(s_{m+1};\theta_k) - \nabla_\theta V(s_m;\theta_k)\right) y_{k+1}. \end{aligned} \tag{2.5}$$

where $y \in \mathbb{R}^{|\mathbb{S}|}$ is a vector.

When the state space $\mathbb{S}$ is continuous, $V(s;\theta) \in \mathbb{R}$ is usually represented as a scalar function. Note that simply set $g = R(s_m) + \gamma V(s_{m+1}) - V(s_m)$ and $f = g^2$ does not necessarily give the fixed point of the Bellman operator because there is no conditional expectation of s_{m+1} on s_m in the objective function. So it is less clear how to apply SCGD to the case of the continuous state space.

The second method is based on the primal-dual formulation. The GTD method proposed in (Sutton et al., 2008) is later shown in (Liu et al., 2015) to a primal-dual algorithm for the mean-squared temporal difference minimization problem. The primal-dual representation of the objective function (2.4) is

$$\min_{\theta} \max_{y \in \mathbb{R}^{|\mathbb{S}|}} \mathbb{E}\left[\left(\mathbb{E}\left[R(s_m) + \gamma V(s_{m+1};\theta) - V(s_m;\theta) | s_m \right] \right) y(s_m) - \frac{1}{2} y(s_m)^2 \right].$$

It is easy to check that for a fixed θ, the maximum over y gives the same objective function (2.4). The SGD method for the above minimax problem is (2.5), which is the same as SCGD.

However, when the state space is continuous, $y(\cdot)$ usually is represented by a nonlinear function, then the minimax problem becomes

$$\min_{\theta} \max_{\omega \in \mathbb{R}^{|\mathbb{S}|}} \mathbb{E}\left[\left(\mathbb{E}\left[R(s_m) + \gamma V(s_{m+1};\theta) - V(s_m;\theta) | s_m \right] \right) y(s_m;\omega) - \frac{1}{2} y(s_m;\omega)^2 \right], \tag{2.6}$$

which makes the maximum problem non-concave. Solving the maximum problem over ω by (stochastic) gradient descent will not necessarily give the objective function (2.4), which brings in more instability to the performance of the primal-dual algorithm. (See Figure 2 in Section 5.1.)

To summarize, when compared with the other methods, BFF has an advantage for continuous state space model-free RL problems. BFF also gives comparable results in discrete state space with the above methods mentioned. However, BFF requires the smoothness assumption of the underlying dynamics, while such an assumption is not necessary for applying the primal-dual method or the SCGD.

3. Algorithms

This section describes the BFF algorithms in both the nonlinear approximation setting and the tabular setting.

3.1. Algorithm with nonlinear approximation

Let us write the BRM (2.4) in an abstract form,

$$\theta^* = \min_{\theta \in \Omega} J(\theta), \quad J(\theta) := \mathbb{E}\left[\frac{1}{2} \left(\mathbb{E}\left[f(s_m, s_{m+1};\theta) | s_m \right] \right)^2 \right], \tag{3.1}$$

where $\Omega \subset \mathbb{R}^d$ is a compact domain, and

$$f(s_m, s_{m+1};\theta) = R(s_m) + \gamma V(s_{m+1};\theta) - V(s_m;\theta)$$

is the Bellman residual (sometimes also called Bellman error in the literature). Note that, when V is approximated by a neural network with standard activation functions, the boundedness of θ and s_m implies that V is also bounded. Following (2.1), define

$$\Delta s_m := s_{m+1} - s_m = a(s_m)\epsilon + \sigma(s_m)\sqrt{\epsilon} Z_m. \tag{3.2}$$

Suppose now that functions $\alpha(s)$ and $\sigma(s)$ were known explicitly. SGD with uncorrelated samples at each state updates the parameter θ as follows

Algorithm 1 **[Uncorrelated sampling]** *Given a trajectory $\{s_m\}_{m=0}^T$, at step k, randomly select M elements from $\{0, \cdots, T\}$ to form the index subset B_k, generate a new s'_{m+1} from s_m according to (2.1), and update*

$$\theta_{k+1} = \theta_k - \frac{\tau}{M} \sum_{m \in B_k} f(s_m, s_{m+1}; \theta_k) \nabla_\theta f(s_m, s'_{m+1}; \theta_k),$$

where τ is the learning rate, and M is the batch size.

However, as we pointed out already, generating another s'_{m+1} is impractical as $\alpha(s)$ and $\sigma(s)$ are unknown in the model-free setting. Instead, the following *Sample-cloning* algorithm is sometimes used.

Algorithm 2 **[Sample-cloning]** *Given a trajectory $\{s_m\}_{m=0}^T$, at step k with τ, B_k, M the same as in Algorithm 1, update*

$$\theta_{k+1} = \theta_k - \frac{\tau}{M} \sum_{m \in B_k} f(s_m, s_{m+1}; \theta_k) \nabla_\theta f(s_m, s_{m+1}; \theta_k).$$

Note that the gradient in the above algorithm is not an unbiased gradient of the objective function in (3.1). In fact, it is an unbiased gradient of $\mathbb{E}\left[\frac{1}{2}\mathbb{E}\left[f(s_m, s_{m+1}; \theta)^2 | s_m\right]\right]$. We shall see in Section 5 that this algorithm fails to identify the true solution θ^* even if the underlying drift and diffusion terms are smooth.

Borrow from the future. Below we propose two algorithms that approximate the minimizer efficiently when the underlying drift term $\alpha(s)$ and diffusion term $\sigma(s)$ change smoothly. Instead of minimizing $J(\theta)$, the first algorithm minimizes $\hat{J}(\theta)$,

$$\min_{\theta \in \Omega} \hat{J}(\theta), \quad \hat{J}(\theta) := \frac{1}{2}\mathbb{E}\left[\mathbb{E}\left[f(s_m, s_{m+1}; \theta) | s_m\right] \mathbb{E}\left[f(s_m, s_m + \Delta s_{m+1}; \theta) | s_m\right]\right] \tag{3.3}$$

where Δs_{m+1} is defined as (3.2). The main idea is to approximate $s'_{m+1} = s_m + \Delta s_m$ in Algorithm 1 with $s'_{m+1} \approx s_m + \Delta s_{m+1}$, i.e., creating another simulation of $s_m \to s_{m+1}$ by borrowing randomness from the future step $s_{m+1} \to s_{m+2}$. When ϵ is small, and the change of the drift and the diffusion are small as well, we expect the approximation should be close to the unbiased gradient. Due to the independence between Δs_m and Δs_{m+1}, we have

$$\hat{J}(\theta) = \frac{1}{2}\mathbb{E}\left[f(s_m, s_{m+1}; \theta) f(s_m, s_m + \Delta s_{m+1}; \theta)\right]. \tag{3.4}$$

From (3.4), one can directly apply SGD algorithm to update the parameter θ from the observed trajectory $\{s_m\}_{m=0}^T$.

Algorithm 3 **[BFF-loss]** *Given a trajectory $\{s_m\}_{m=0}^T$, at step k, τ, B_k, M are the same as in Algorithm 1*

$$\theta_{k+1} = \theta_k - \frac{\tau}{M} \sum_{m \in B_k} \frac{1}{2} \nabla_\theta \left[f(s_m, s_{m+1}; \theta_k) f(s_m, s_m + \Delta s_{m+1}; \theta_k)\right],$$

where $\Delta s_{m+1} = s_{m+2} - s_{m+1}$.

An alternative algorithm is applying the same technique directly on the unbiased gradient of the true objective function. We will show in Section 5 that the two new algorithms behave similarly in practice.

Algorithm 4 [BFF-gradient] *Given a trajectory* $\{s_m\}_{m=0}^T$*, at step* k*,* τ, B_k, M *are the same as in Algorithm 1*

$$\theta_{k+1} = \theta_k - \frac{\tau}{M} \sum_{m \in B_k} f(s_m, s_{m+1}; \theta_k) \nabla_\theta f(s_m, s_m + \Delta s_{m+1}; \theta_k),$$

where $\Delta s_{m+1} = s_{m+2} - s_{m+1}$.

Although these new BFF algorithms are biased when compared with Algorithm 1, Section 4 proves that the bias of Algorithm 3 is small. More specifically, we show that the differences between Algorithms 1 and 3 are all under control, in terms of the objective function, the evolution of SGD, and the steady-state distribution.

3.2. Algorithms in the tabular setting

For tabular form, the value function $V^* \in \mathbb{R}^n$ satisfies the following Bellman equation,

$$V^* = \mathbb{T}(V^*) = r + \gamma P V^*.$$

In the discrete setting, the BRM becomes

$$V^* = \operatorname*{argmin}_{v \in \mathbb{R}^n} \frac{1}{2} \left\| r + \gamma P v - v \right\|_\mu^2,$$

where μ is the stationary distribution of the Markov chain. The gradient of the above objective function can be written as

$$\nabla_v J = (\gamma P - I)^\top \operatorname{diag}(\mu)(r + \gamma P v - v). \tag{3.5}$$

Since P appears twice in the above formula, in order to obtain an unbiased approximation of the above gradient, we need two simulations from s_m to s_{m+1}. Given a trajectory $\{s_m\}_{m=1}^T$, choose a state $s_m = i$ and the next state $s_{m+1} = j$. Assuming that the Markov chain reaches equilibrium, such a choice is an unbiased estimate for $\operatorname{diag}(\mu)$. If the transition matrix P were known, we could generate a new state $s'_{m+1} = s_{j'}$ and construct an unbiased estimation of $\nabla_v J$: replacing the first and second copies of P in (3.5) with P_1 and P_2 given as follows:

$$(P_1)_{il} = \begin{cases} 1, & l = j' \\ 0, & \text{otherwise} \end{cases}, \quad (P_2)_{il} = \begin{cases} 1, & l = j \\ 0, & \text{otherwise} \end{cases}.$$

Equivalently, the unbiased estimate of the gradient can be written as

$$(\nabla_v J)_i = -(r_i + \gamma v_j - v_i), \quad (\nabla_v J)_{j'} = \gamma(r_i + \gamma v_j - v_i), \quad (\nabla_v J)_l = 0, \quad \forall l \neq i, j'. \tag{3.6}$$

However, in the setup of model-free RL, without knowing the transition matrix P, one needs to approximate V^* based on the observed trajectory $\{s_m\}_{m=1}^T$ alone. The BFF idea $s'_{m+1} \approx s_m +$

$(s_{m+2} - s_{m+1})$ can also be applied here to give rise to the two BFF algorithms in the tabular form. Below are the pseudocodes for Algorithms 1-4 in the tabular form, where v is updated based on an estimate G of the true gradient $\nabla_v J$,

$$v_{k+1} = v_k - \eta G_{m_k},$$

where m_k is randomly selected from $\{1, \cdots, T\}$ and G_m (dropping the k index for notation convenience) is computed differently as follows in the four algorithms.

- **Uncorrelated sampling:** Assume $s_m = i, s_{m+1} = j$, (same for the other three algorithms). Generate a new s'_{m+1} by (5.6) and let $j' = s'_{m+1}$

$$(G_m)_i = -(r_i + \gamma v_j - v_i), \quad (G_m)_{j'} = \gamma(r_i + \gamma v_j - v_i), \quad (G_m)_l = 0, \quad \forall l \neq i, j. \tag{3.7}$$

  The uncorrelated sampling algorithm gives an unbiased estimation of the loss function (3.5). However, it is impractical for model-free RL.

- **Sample-cloning:**

$$(G_m)_i = -(r_i + \gamma v_j - v_i), \quad (G_m)_j = \gamma(r_i + \gamma v_j - v_i), \quad (G_m)_l = 0, \quad \forall l \neq i, j. \tag{3.8}$$

- **BFF-gradient:** Let $j' = s_m + (s_{m+2} - s_{m+1})$.

$$(G_m)_i = -(r_i + \gamma v_j - v_i), \quad (G_m)_{j'} = \gamma(r_i + \gamma v_j - v_i), \quad (G_m)_l = 0, \quad \forall l \neq i, j. \tag{3.9}$$

- **BFF-loss:** Let $j' = s_m + (s_{m+2} - s_{m+1})$.

$$\begin{aligned} &(G_m)_i = -\frac{1}{2}(r_i + \gamma v_j - v_i) - \frac{1}{2}(r_i + \gamma v_{j'} - v_i), \quad (G_m)_{j'} = \frac{\gamma}{2}(r_i + \gamma v_j - v_i), \\ &(G_m)_j = \frac{\gamma}{2}(r_i + \gamma v_{j'} - v_i), \quad (G_m)_l = 0, \quad \forall l \neq i, j, j'. \end{aligned} \tag{3.10}$$

4. Error estimates

The aim of this section is to show that Algorithm 3 (BFF-loss) for the continuous state space RL with underlying dynamics (2.1) is close to Algorithm 1 (uncorrelated sampling) statistically.

4.1. Difference between the objective functions

Let us introduce

$$\tilde{J}(\theta) := \hat{J}(\theta) - J(\theta). \tag{4.1}$$

Notice that $\tilde{J}(\theta) = \mathbb{E}\tilde{j}(s_m; \theta)$ with

$$\tilde{j}(s_m; \theta) = \mathbb{E}\left[f(s_m, s_{m+1}; \theta)|s_m\right] \mathbb{E}\left[f(s_m, s_m + \Delta s_{m+1}; \theta) - f(s_m, s_m + \Delta s_m; \theta)|s_m\right]. \tag{4.2}$$

The following lemma shows that if the values and derivatives of the drift, diffusion, and nonlinear approximation are bounded, then the difference between the two objective functions $J(\theta)$ and $\hat{J}(\theta)$ is smaller than $C\epsilon^2$, with the constant depending only on the size of α, σ, f and their derivatives until the second order.

Lemma 1 *For $\tilde{J}, \tilde{j}$ defined in (4.1), (4.2), if $\left\|\alpha^{(k)}(\cdot)\right\|_{L^\infty}, 0 \le k \le 3, \left\|\sigma^{(l)}(\cdot)\right\|_{L^\infty}, 0 \le l \le 4$ are bounded, and $\left\|\partial_{s_2}^i f(s_1, s_2; \theta)\right\|_{L^\infty_{s_1,s_2}}, 0 \le i \le 5$ are also uniformly bounded for any θ, then for all θ, one has*

$$\begin{aligned} \left\|\tilde{j}(s,\theta)\right\|_{L^\infty_s} &\le C\epsilon^2 + o(\epsilon^2), \\ \tilde{J}(\theta) &\le C\epsilon^2 + o(\epsilon^2), \end{aligned} \tag{4.3}$$

for some constant C depending on $\left\|\alpha^{(i)}(\cdot)\right\|_{L^\infty}, \left\|\sigma^{(i)}(\cdot)\right\|_{L^\infty}, \left\|\partial_{s_2}^i f(s_1, s_2; \theta)\right\|_{L^\infty_{s_1,s_2,\theta}}, 0 \le i \le 2$.

The boundedness of the residual f is followed by the boundedness of R and V. Since we assume that the state space $\mathbb{S}$ and the parameter space Ω are both compact, for parametric value approximation, such as neural network, it is natural to assume R, V are bounded.

Proof See Appendix A. ■

4.2. Difference between the asymptotic distributions

In this section and Section 4.3, we assume the optimization region of (3.1) is a bounded connected open subset Ω in $\mathbb{R}^d$. This assumption is to guarantee the Poincare inequality. The updates of the parameter θ of $J(\cdot)$ and $\hat{\theta}$ of $\hat{J}(\cdot)$ by SGD according to Algorithms 1 and 3 can be approximated by stochastic differential equations (SDEs) with $\eta = \frac{\tau}{M}$ (Li et al., 2017; Hu et al., 2017)

$$\begin{aligned} d\theta_t &= -\nabla J(\theta_t)dt + \sqrt{\eta}\Sigma^{\frac{1}{2}}(\theta_t)dB_t; \\ d\hat{\theta}_t &= -\nabla\left(J(\hat{\theta}_t) + \tilde{J}(\hat{\theta}_t)\right)dt + \sqrt{\eta}\left(\Sigma(\hat{\theta}_t) + \tilde{\Sigma}(\hat{\theta}_t)\right)^{\frac{1}{2}} dB_t, \end{aligned}$$

where

$$\begin{aligned} \Sigma(\theta_t) &= \mathbb{V}\left[\frac{1}{2}(\mathbb{E}\left[f(s_m, s_{m+1}; \theta)|s_m\right])^2\right]; \\ \tilde{\Sigma}(\hat{\theta}_t) &= \mathbb{V}\left[\frac{1}{2}(\mathbb{E}\left[f(s_m, s_{m+1}; \theta)|s_m\right])^2 + \tilde{j}\right] - \Sigma(\hat{\theta}). \end{aligned}$$

Here $\mathbb{V}$ represents the variance, and $\tilde{j}$ is defined in (4.2).

Therefore, the corresponding probability density functions $p(t,\theta)$, $\hat{p}(t,\theta)$ of θ_t, $\hat{\theta}_t$ can be described by (Pavliotis, 2014)

$$\partial_t p(t,\theta) = \nabla \cdot \left[(\nabla J)\, p + \frac{\eta}{2}\nabla \cdot (\Sigma p)\right]; \tag{4.4}$$

$$\partial_t \hat{p}(t,\theta) = \nabla \cdot \left[\left(\nabla J + \nabla \tilde{J}\right)\hat{p} + \frac{\eta}{2}\nabla \cdot \left((\Sigma + \tilde{\Sigma})\hat{p}\right)\right], \tag{4.5}$$

with the same initial data $p(0,\theta) = \hat{p}(0,\theta)$. We use the reflecting boundary condition on $\partial\Omega$,

$$\begin{aligned} \left(\nabla J p + \frac{\eta}{2}\nabla \cdot (\Sigma p)\right) \cdot \boldsymbol{n}\Big|_{\partial\Omega} &= 0, \\ \left((\nabla J + \nabla \tilde{J})\hat{p} + \frac{\eta}{2}\nabla \cdot (\Sigma + \tilde{\Sigma}\hat{p})\right) \cdot \boldsymbol{n}\Big|_{\partial\Omega} &= 0, \end{aligned} \tag{4.6}$$

which means that θ will be reflected after hitting the boundary.

Besides, from the estimation we obtained in (4.3), we know that $\nabla\tilde{J} \leq O(\epsilon^2)$, and it is easy to see that $\tilde{\Sigma} \leq O(\epsilon^2)$ because

$$\begin{aligned}\tilde{\Sigma}(\hat{\theta}) =& \Sigma(\hat{\theta}) + \mathbb{V}[\tilde{j}] + \mathbb{E}\left[\tilde{j}(\mathbb{E}\left[f(s_m, s_{m+1}; \theta)|s_m\right])^2\right] - \mathbb{E}\left[\tilde{j}\right]\mathbb{E}\left[(\mathbb{E}\left[f(s_m, s_{m+1}; \theta)|s_m\right])^2\right] - \Sigma(\hat{\theta}) \\ \leq& O(\epsilon^4) + C(\hat{\theta})\epsilon^2 \leq C\epsilon^2 + o(\epsilon^2).\end{aligned} \tag{4.7}$$

Assumption 2 *We assume both loss functions $J(\theta)$ and $\hat{J}(\theta)$ satisfy the following:*

- $\int e^{-J(\theta)}d\theta \leq \infty$, *and* $\int e^{-\hat{J}(\theta)}d\theta \leq \infty$.
- *The Frobenius norms of $G = \|H(J)\|_{L^\infty_\theta}$ and $\hat{G} = \left\|H(\hat{J})\right\|_{L^\infty_\theta}$ are bounded by a constant M, where H represents the Hessian and L^∞_θ is taken element-wisely to the matrix.*

The first assumption is to guarantee that the steady-state is well defined. The second assumption is used to prove the boundedness of $\nabla\hat{p}$.

Theorem 3 *Assume $\Sigma \sim O(1), \tilde{\Sigma}$ are both constants, then there exist steady-state distributions $p^\infty, \hat{p}^\infty$ for (4.4), (4.5)*

$$p^\infty(\theta) = \frac{1}{Z}e^{-\beta J(\theta)}, \quad \hat{p}^\infty(\theta) = \frac{1}{\hat{Z}}e^{-\hat{\beta}(J(\theta)+\tilde{J}(\theta))}, \quad \beta = \frac{2}{\eta\Sigma}, \quad \hat{\beta} = \frac{2}{\eta(\Sigma+\tilde{\Sigma})},$$

where $Z = \int e^{\frac{-2J}{\eta\Sigma}}d\theta$, $\hat{Z} = \int e^{\frac{-2(J+\tilde{J})}{\eta(\Sigma+\tilde{\Sigma})}}d\theta$ are normalized constants. In addition,

$$\left\|\frac{\hat{p}^\infty}{p^\infty}\right\|_{L^\infty} \leq 1 + O\left(\frac{\epsilon^2}{\eta}\right).$$

Theorem 3 implies the following:

- If the probability of the unbiased SGD (Algorithm 1) converging to the optimal θ^* is p, then the probability of Algorithm 3 is $\left(1 + O(\frac{\epsilon^2}{\eta})\right)p$.
- In order to make Algorithm 3 behaves similarly to the unbiased SGD, we have to let $\frac{\epsilon^2}{\eta}$ be small, which means we require ϵ to be small, but η to be larger than ϵ^2. This makes sense because we are minimizing a biased objective function, so if we do the biased SGD too carefully, it will end up a worse minimizer of the true objective function.

Proof See Appendix B. ■

4.3. Difference between finite time distributions

Theorem 3 is about the asymptotic behavior of the algorithm. Now we will study the difference between the two algorithms at a finite time. The following Poincare inequality of the probability measure $d\mu = p^\infty d\theta$ or $d\mu = \hat{p}^\infty d\theta$ in a bounded connected domain is valid for any $\int h d\mu = 0$,

$$\int_\Omega |\nabla h|^2 \, d\mu \geq \lambda \int_\Omega h^2 d\mu, \tag{4.8}$$

with a constant λ depending on $d\mu$ and Ω. Based on the above Poincare inequality, we can prove the difference between the two algorithms, as shown in the following theorem. The difference is measured in the following norm,

$$\|h\|_*^2 = \int h^2 \frac{1}{p^\infty} d\theta, \tag{4.9}$$

where p^∞ is defined in Theorem 3.

Theorem 4 *Under Assumption 2 and the assumptions in Theorem 3, one has,*

$$\|p(t,\theta) - \hat{p}(t,\theta)\|_*^2 \leq \left(1 + O\left(\frac{\epsilon^2}{\eta}\right)\right) O(\epsilon^2). \tag{4.10}$$

Theorem 4 tells us that the evolution of θ in Algorithm 3 differs from the unbiased SGD within an error of $\left(1 + O\left(\frac{\epsilon^2}{\eta}\right)\right) O(\epsilon^2)$.

Proof See Appendix C ■

5. Numerical Examples

Several numerical examples are presented here to demonstrate the performance of the proposed algorithms. Recall that the goal of the prediction problem (i.e., policy evaluation) is to approximate $V(s)$ based on the trajectories.

5.1. Continuous state space

Consider a Markov decision process with a continuous state space $\mathbb{S} = \{s \in (0, 2\pi]\}$. Suppose that the transition probability is prescribed implicitly via the following dynamics

$$\begin{aligned} &s_{m+1} = s_m + \alpha(s_m)\epsilon + \sigma(s_m)\sqrt{\epsilon} Z_m, \\ &\alpha(s) = 2\sin(s)\cos(s), \quad \sigma(s) = 1 + \cos(s)^2, \quad \epsilon = 0.1. \end{aligned} \tag{5.1}$$

The immediate reward function is $R(s) = (\cos(2s) + 1)$, and the discount factor γ is 0.9.

A three-layer, fully connected neural network is used to approximate the value function $V(s;\theta)$. The network has two hidden layers with $\cos$ as its activation function. Each hidden layer contains 50 neurons, i.e.,

$$\begin{aligned} &V(s;\theta) = V(x;\{w_i, b_i\}_{i=1}^3) = L_{w_3,b_3} \circ \cos \circ L_{w_2,b_2} \circ \cos \circ L_{w_1,b_1}((\cos s, \sin s)), \\ &L_{w_i,b_i}(x) = w_i x + b_i, \quad w_i \in \mathbb{R}^{n_{i-1}\times n_i}, \quad b_i \in \mathbb{R}^{n_i}, \quad n_0 = 2, n_1 = n_2 = 50, n_3 = 1. \end{aligned} \tag{5.2}$$

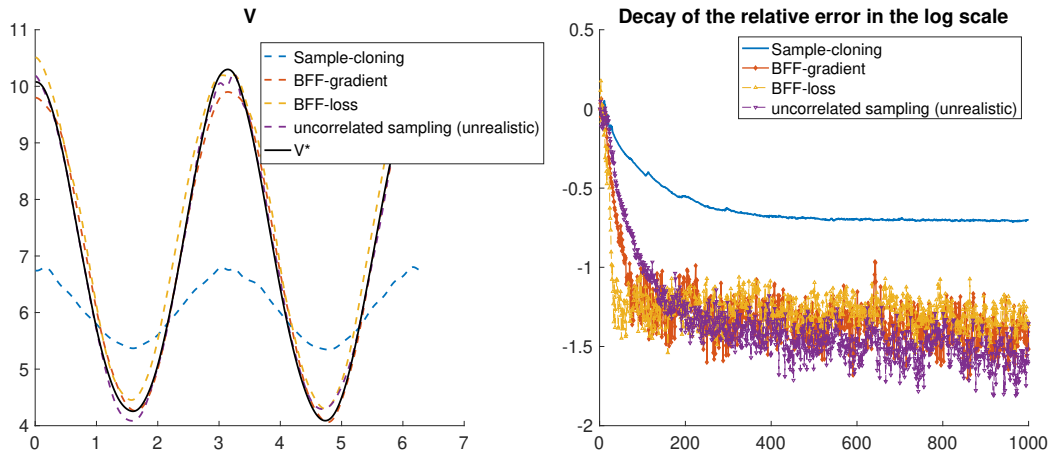

Figure 1: Continuous state space: approximation with a 3-layer neural network with batch size 1000.

The optimal θ^* is computed by Algorithms 1-4 based on a trajectory $\{s_m\}_{m=1}^{10^6}$ generated from (5.1). The function f in Algorithms 1-4 refers to

$$f(s_m, s_{m+1}, \theta) = R(s_m) + \gamma V(s_{m+1}; \theta) - V(s_m; \theta) \tag{5.3}$$

in the value evaluation. The learning rate τ and the batch size M are set to be

$$\tau = 0.1, \quad M = 1000. \tag{5.4}$$

Once the whole trajectory is recorded, we perform a random permutation and use batches of size $M = 1000$ for training. In each experiment, the SGD algorithm runs for a single epoch with the same initialization θ_0 and randomly-permuted trajectory. The error e_k at each step k is defined as the squared L_2 norm $\|V(\cdot, \theta_k) - V^*(\cdot)\|_2$, where the reference solution $V^*(s)$ is computed by running Algorithm 1 for 10 epochs based on a longer trajectory $\{s_m\}_{m=1}^{10^7}$, with hyper-parameters $\tau = 0.01$ and $M = 1000$. The left plot of Figure 1 shows the final $V(s, \theta)$ obtained by four different methods, while the relative errors $\log_{10}(e_k/e_0)$ in the log scale are summarized in the right plot. The plots show that the Sample-cloning algorithm introduces a rather large error while the BFF algorithms are much closer to the (impractical) uncorrelated sampling algorithm.

Next, We compare the BFF algorithm with the primal-dual algorithm, i.e., the mini-batch SGD method for the objective function (2.6). The algorithm reads,

$$\begin{aligned} \omega_{k+1} &= \omega_k + \frac{\beta}{M} \sum_{s_m \in B_k} \left(f(s_m, s_{m+1}, \theta_k) \nabla_\omega y(s_m; \omega_k) - y(s_m; \omega_k) \nabla_\omega y(s_m; \omega_k) \right), \\ \theta_{k+1} &= \theta_k - \frac{\tau}{M} \left(\sum_{s_m \in B_k} \nabla_\theta f(s_m, s_{m+1}, \theta_k) y(s_m; \omega_{k+1}) \right), \end{aligned} \tag{5.5}$$

with f, τ, M as in (5.3), (5.4) and $\beta = 0.5$. Each simulation uses the same initialization for θ_0 and the same random permutation to the trajectory. The results are summarized in Figure 2. The five dash curves in Figure 2 are the relative errors of the primal-dual algorithm with five different

random initializations of ω_0, and among them, only one simulation converges within 1000 steps. This is because the algorithm might be trapped at a suboptimal stationary point in the nonlinear approximation setting.

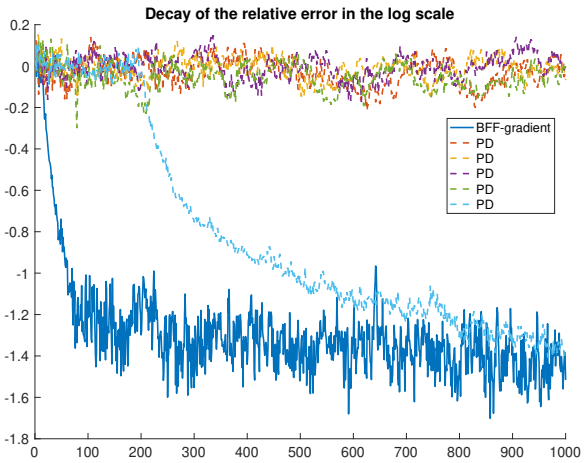

Figure 2: Continuous state space: approximation with a 3-layer neural network with batch size 1000.

5.2. Discrete state space

Consider a Markov decision process with a discrete state space $\mathbb{S} = \{i = 0, \cdots, n-1\}$ for $n = 32$. The transition matrix is the following,

$$\begin{aligned} \mathbb{P}_{i,i+1} &= \frac{1}{2} - \frac{1}{5}\sin\frac{2\pi i}{n}, \quad \text{when } i = n-1, i+1 = 0; \\ \mathbb{P}_{i,i-1} &= \frac{1}{2} + \frac{1}{5}\sin\frac{2\pi i}{n}, \quad \text{when } i = 0, i-1 = n-1. \end{aligned} \tag{5.6}$$

The immediate reward function is $r \in \mathbb{R}^n$ with $r_i = 1 + \cos\frac{2\pi i}{n}$, and the discount rate is $\gamma = 0.9$. The value function $V^* \in \mathbb{R}^n$ satisfies the following Bellman equation,

$$V^* = \mathbb{T}(V^*) = r + \gamma P V^*, \tag{5.7}$$

and can be solved from (5.7). The numerical results are carried out both with the neural network approximation and in the tabular setting.

Neural network approximation. The same neural network architecture is used as in (5.2) with input $s = \frac{2\pi i}{n}$ for approximating the value function. We run Algorithms 1-4 and the primal-dual algorithm (2.5) to approximate θ^* based on a trajectory $\{s_m\}_{m=1}^T, T = 4 \times 10^6$ simulated from (5.6).

Figure 3 summarizes the results obtained from SGD with a single batch at the learning rate $\tau = 0.001$ and $\beta = 0.1$ for the primal-dual algorithm (2.5). The same initialization for θ_0 and random permutation are used for all simulations. The initialization for y_0 in the primal-dual algorithm is the

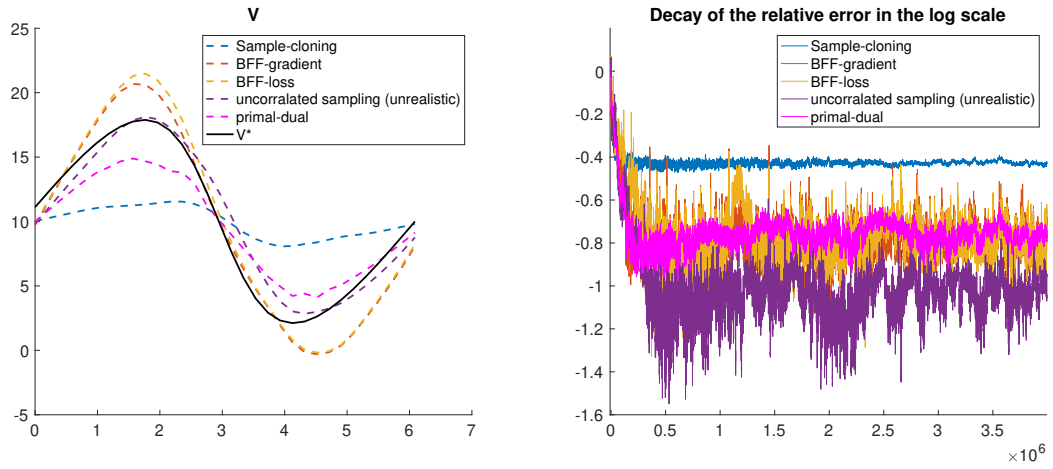

Figure 3: Discrete state space: approximation with a 3-layer neural network with batch size 1.

zero vector. The relative errors e_k/e_0, with e_k defined as

$$e_k = \sqrt{\sum_{i=0}^{n-1} \left(V\left(\frac{2\pi i}{n}, \theta_k\right) - V_i^* \right)^2},$$

are shown in the log scale on the right. The plots of Figure 3 demonstrate that the performance of the two new algorithms (BFF-loss and BFF-gradient) are very similar. Although they are less accurate than the uncorrelated case, the performance is much better than the Sample-cloning algorithm and comparable to the primal-dual algorithm.

Tabular setting. The BFF algorithms proposed in Section 3.2 are used to approximate V^* in the tabular form. In the experiments, $\tau = 0.1$ and the SGD is run for 5 epochs. Again, we compare the BFF with the primal-dual algorithm (2.5) (equivalent to the SCGD algorithm in this case) with $\beta = 0.5$. In the tabular setting, $V(s;\theta) = \Phi(s)^\top \theta$ with $\theta \in \mathbb{R}^N$ and $\Phi(s_i) = e_i \in \mathbb{R}^N$, where $\{e_i\}$ are the standard basis vectors of $\mathbb{R}^N$. The results summarized in Figure 4 shows that the BFF-loss algorithm converges slightly faster than the primal-dual/SCGD algorithm, and both significantly better than the Sample-cloning algorithm. Comparing with Figure 3, it seems that the neural network approximation results in significantly faster error decay at the initial stage of the training.

Summary. The numerical experiments suggest that BFF algorithms significantly outperform the Sample-cloning algorithm, as the theory predicted. For the continuous state space setting, the BFF algorithm performs better than primal-dual algorithms. For the discrete state space setting, BFF is comparable to the primal-dual algorithm and SCGD.

Acknowledgments

The work of L.Y. and Y.Z. is partially supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program. The work of L.Y. is also partially supported by the National Science

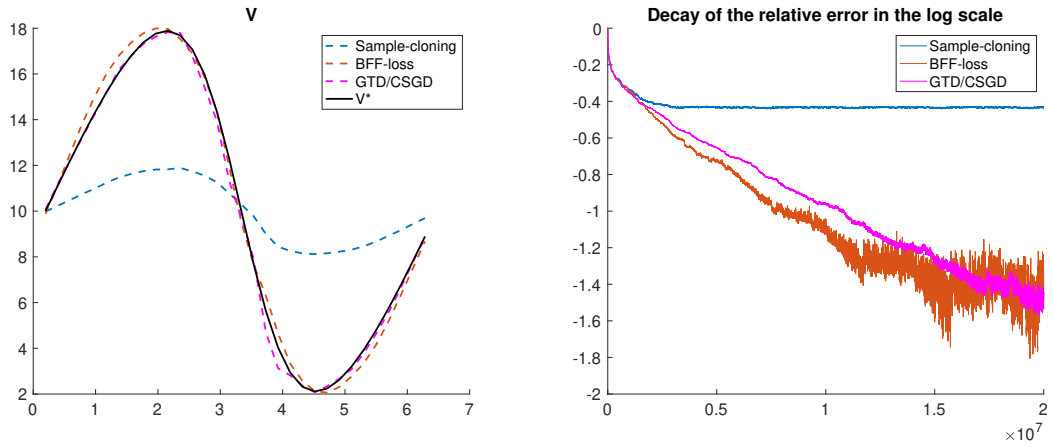

Figure 4: Discrete state space: tabular approximation.

Foundation under award DMS-1818449. L.Y. thanks Mohammad Ghavamzadeh, Yuandong Tian, and Amy Zhang for helpful discussions.

References

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*, volume 5. Athena Scientific Belmont, MA, 1996.

Shalabh Bhatnagar, Doina Precup, David Silver, Richard S Sutton, Hamid R Maei, and Csaba Szepesvári. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems*, pages 1204–1212, 2009.

Justin A Boyan and Andrew W Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in neural information processing systems*, pages 369–376, 1995.

Bo Dai, Albert Shaw, Lihong Li, Lin Xiao, Niao He, Zhen Liu, Jianshu Chen, and Le Song. Sbeed: Convergent reinforcement learning with nonlinear function approximation. *arXiv preprint arXiv:1712.10285*, 2017.

Wenqing Hu, Chris Junchi Li, Lei Li, and Jian-Guo Liu. On the diffusion approximation of nonconvex stochastic gradient descent. *arXiv preprint arXiv:1705.07562*, 2017.

Qianxiao Li, Cheng Tai, et al. Stochastic modified equations and adaptive stochastic gradient algorithms. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2101–2110. JMLR. org, 2017.

Bo Liu, Ji Liu, Mohammad Ghavamzadeh, Sridhar Mahadevan, and Marek Petrik. Finite-sample analysis of proximal gradient TD algorithms. In *UAI*, pages 504–513. Citeseer, 2015.

Hamid R Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S Sutton. Toward off-policy learning control with function approximation. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 719–726, 2010.

Grigorios A Pavliotis. *Stochastic processes and applications: Diffusion processes, the Fokker-Planck and Langevin equations*, volume 60. Springer, 2014.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Stochastic processes and applicationsature*, 529(7587):484, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3 (1):9–44, 1988.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, Csaba Szepesvári, and Hamid Reza Maei. A convergent O(n) algorithm for off-policy temporal-difference learning with linear function approximation. *Advances in neural information processing systems*, 21(21):1609–1616, 2008.

Richard S Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 993–1000. ACM, 2009.

John N Tsitsiklis and Benjamin Van Roy. Analysis of temporal-diffference learning with function approximation. In *Advances in neural information processing systems*, pages 1075–1081, 1997.

Mengdi Wang, Ji Liu, and Ethan Fang. Accelerating stochastic composition optimization. In *Advances in Neural Information Processing Systems*, pages 1714–1722, 2016.

Mengdi Wang, Ethan X Fang, and Han Liu. Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions. *Mathematical Programming*, 161(1-2):419–449, 2017.

Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

Appendix A. Proof of Lemma 1

Proof Let

$$\delta(s_m,\theta)=\mathbb{E}\left[f(s_m,s_m+\Delta s_{m+1};\theta)-f(s_m,s_m+\Delta s_m;\theta)|s_m\right],$$

then

$$\tilde{j}(s_m,\theta)=\mathbb{E}\left[f(\,s_m,s_{m+1};\theta)|\,s_m\right]\delta(s_m,\theta);\quad \tilde{J}(\theta)=\frac{1}{2}\mathbb{E}\,\tilde{j}(s_m,\theta). \tag{A.1}$$

We first estimate the term $\delta(s_m,\theta)$. By Taylor expansion,

$$\begin{aligned}
&f(s_m,s_m+\Delta s_{m+1};\theta)-f(s_m,s_m+\Delta s_m;\theta)\\
=&[f(s_m,s_m+\Delta s_{m+1};\theta)-f(s_m,s_m;\theta)]-[f(s_m,s_m+\Delta s_m-f(s_m,s_m;\theta)]\\
=&\partial_{s_2}f(s_m,s_m;\theta)\left(\Delta s_{m+1}-\Delta s_m\right)+\frac{1}{2}\partial^2_{s_2}f(s_m,s_m;\theta)\left(\Delta s^2_{m+1}-\Delta s^2_m\right)\\
&+\frac{1}{6}\partial^3_{s_2}f(s_m,s_m;\theta)\left(\Delta s^3_{m+1}-\Delta s^3_m\right)+\frac{1}{24}\partial^4_{s_2}f(s_m,s_m;\theta)\left(\Delta s^4_{m+1}-\Delta s^4_m\right)\\
&+\frac{1}{120}\left(\partial^5_{s_2}f(s_m,s_m+s';\theta)\Delta s^5_{m+1}-\partial^5_{s_2}f(s_m,s_m+s'';\theta)\Delta s^5_m\right),
\end{aligned}$$

for some $s'\in(0,\Delta s_{m+1})$, $s''\in(0,\Delta s_m)$. By the definition of Δs_m in (3.2), we have $\Delta s^5_m,\Delta s^5_{m+1}\leq o(\epsilon^2)$, so the last term of the above equation is of order $o(\epsilon^2)$. Therefore, one has,

$$\delta(s_m,\theta)=\sum_{i=1}^{4}\frac{1}{i!}\partial^i_{s_2}f(s_m,s_m;\theta)\mathbb{E}\left[\Delta s^i_m-\Delta s^i_{m+1}|s_m\right]+o(\epsilon^2). \tag{A.2}$$

The Taylor expansion of $\alpha(s_{m+1}), \sigma(s_{m+1})$ can be represented by

$$\begin{aligned}\alpha(s_{m+1}) &= \alpha(s_m) + \alpha'(s_m)\Delta s_m + \frac{1}{2}\alpha''(s_m)\Delta s_m^2 + o(\epsilon),\\ \sigma(s_{m+1}) &= \sigma(s_m) + \sigma'(s_m)\Delta s_m + \frac{1}{2}\sigma''(s_m)\Delta s_m^2 + \frac{1}{6}\sigma'''(s_m)\Delta s_m^3 + o(\epsilon^{3/2}),\end{aligned} \tag{A.3}$$

which gives,

$$\begin{aligned}&\Delta s_{m+1} - \Delta s_m = (\alpha(s_{m+1}) - \alpha(s_m))\,\epsilon + \sigma(s_m)Z_{m+1}\sqrt{\epsilon} - \sigma(s_m)Z_m\sqrt{\epsilon}\\ =&\left(\alpha'(s_m)\Delta s_m + \frac{1}{2}\alpha''(s_m)\Delta s_m^2\right)\epsilon + o(\epsilon^2)\\ &+\left(\sigma(s_m) + \sigma'(s_m)\Delta s_m + \frac{1}{2}\sigma''(s_m)\Delta s_m^2 + \frac{1}{6}\sigma'''(s_m)\Delta s_m^3\right)Z_{m+1}\sqrt{\epsilon} - \sigma(s_m)Z_m\sqrt{\epsilon} + o(\epsilon^2).\end{aligned}$$

Since $\mathbb{E}[h(s_m)Z_m|s_m] = \mathbb{E}[g(s_m)Z_{m+1}|s_m] = 0$ for any functions h, g, the last line of the above equation vanishes after taking conditional expectation on s_m. This implies,

$$\begin{aligned}&\mathbb{E}\left[\Delta s_{m+1} - \Delta s_m|s_m\right] = \mathbb{E}\left[\left(\alpha'(s_m)\Delta s_m + \frac{1}{2}\alpha''(s_m)\Delta s_m^2\right)\epsilon|s_m\right] + o(\epsilon^2)\\ =&\mathbb{E}\left[\left(\alpha'(\alpha\epsilon + \sigma\sqrt{\epsilon}Z_m) + \frac{1}{2}\alpha''(\alpha^2\epsilon^2 + 2\alpha\sigma\epsilon^{3/2}Z_m + \sigma^2\epsilon Z_m^2)\right)\epsilon|s_m\right] + o(\epsilon^2)\\ =&\alpha'\alpha\epsilon^2 + \frac{1}{2}\alpha''\sigma^2\epsilon^2 + o(\epsilon^2).\end{aligned} \tag{A.4}$$

Here α, α', σ refers to the function's value at s_m, similar for $\sigma', \sigma'', \partial^i_{s_2} f$. We omit (s_m) when the functions has its value at s_m.

Using (A.3), one can estimate Δs^i_{m+1} for $i = 2, 3, 4$ as follows,

$$\begin{aligned}\Delta s^2_{m+1} =&\alpha(s_{m+1})^2\epsilon^2 + \sigma(s_{m+1})^2 Z^2_{m+1}\epsilon + 2\alpha(s_{m+1})\sigma(s_{m+1})Z_{m+1}\epsilon^{3/2}\\ =&\alpha^2\epsilon^2 + \left(\sigma^2 + (\sigma')^2\Delta s_m^2 + 2\sigma\sigma'\Delta s_m + \sigma\sigma''\Delta s_m^2\right)Z^2_{m+1}\epsilon\\ &+2\left(\alpha\sigma + \alpha\sigma'\Delta s_m + \alpha'\sigma\Delta s_m\right)Z_{m+1}\epsilon^{3/2} + o(\epsilon^2);\\ \Delta s^3_{m+1} =&3\alpha(s_{m+1})\sigma^2(s_{m+1})Z^2_{m+1}\epsilon^2 + \sigma^3(s_{m+1})Z^3_{m+1}\epsilon^{3/2} + o(\epsilon^2)\\ =&3\alpha\sigma^2 Z^2_{t+1}\epsilon^2 + \sigma^3(s_{m+1})Z^3_{m+1}\epsilon^{3/2} + o(\epsilon^2);\\ \Delta s^4_{m+1} =&\sigma^4(s_{m+1})Z^4_{m+1}\epsilon^2 + o(\epsilon^2) = \sigma^4 Z^4_{m+1}\epsilon^2 + o(\epsilon^2).\end{aligned}$$

Therefore,

$$\begin{aligned}\mathbb{E}\left[\Delta s^2_{m+1} - \Delta s^2_m|s_m\right] =&\mathbb{E}\left[\left(\sigma'^2\Delta s_m^2 + 2\sigma\sigma'\Delta s_m + \sigma\sigma''\Delta s_m^2\right)\epsilon|s_m\right] + o(\epsilon^2)\\ =&\mathbb{E}\left[\left(\left(\sigma'^2 + \sigma\sigma''\right)\left(\sigma^2 Z_m^2\epsilon + o(\epsilon)\right) + 2\sigma\sigma'(\alpha\epsilon + \sigma Z_m\sqrt{\epsilon})\right)\epsilon|s_m\right] + o(\epsilon^2)\\ =&\left(\sigma'^2 + \sigma\sigma''\right)\sigma^2\epsilon^2 + 2\sigma\sigma'\alpha\epsilon^2 + o(\epsilon^2);\\ \mathbb{E}\left[\Delta s^3_{m+1} - \Delta s^3_m|s_m\right] =&o(\epsilon^2);\\ \mathbb{E}\left[\Delta s^4_{m+1} - \Delta s^4_m|s_m\right] =&o(\epsilon^2).\end{aligned} \tag{A.5}$$

Hence, by inserting (A.4) and (A.5) into (A.2) gives,

$$\delta(\theta) = \left[\partial_{s_2} f\left(\alpha'\alpha + \frac{1}{2}\alpha''\sigma^2\right) + \partial_{s_2}^2 f\left(\sigma'^2\sigma^2 + \sigma\sigma''\sigma^2 + 2\sigma\sigma'\alpha\right)\right]\epsilon^2 + o(\epsilon^2). \tag{A.6}$$

As defined in (4.2) and (A.1), the completion of the proof is followed by,

$$\begin{aligned}\tilde{j} &= \frac{1}{2}\mathbb{E}[f(s_m, s_{m+1}; \theta)|s_m]\delta \leq C\epsilon^2 + o(\epsilon^2);\\ \tilde{J} &= \mathbb{E}\tilde{j} \leq C\epsilon^2 + o(\epsilon^2).\end{aligned}$$

■

Appendix B. Proof of Theorem 3

Proof We first observe

$$\frac{\hat{p}^\infty}{p^\infty} = \frac{Z}{\hat{Z}}\frac{e^{-\hat{\beta}(J(\theta)+\tilde{J}(\theta))}}{e^{-\beta J(\theta)}} = \frac{Z}{\hat{Z}}e^{-(\hat{\beta}-\beta)J(\theta)}e^{-\hat{\beta}\tilde{J}(\theta)} = \frac{Z}{\hat{Z}}e^{\frac{\tilde{\Sigma}}{\eta}\frac{2J(\theta)}{\Sigma(\Sigma+\tilde{\Sigma})}}e^{-\frac{\tilde{J}(\theta)}{\eta}\frac{2}{(\Sigma+\tilde{\Sigma})}}.$$

By the fact that $\tilde{\Sigma} \leq O(\epsilon^2), \tilde{J} \leq O(\epsilon^2)$, we have

$$\left\|\frac{\hat{p}^\infty}{p^\infty}\right\|_{L^\infty} \leq \frac{Z}{\hat{Z}}e^{O\left(\frac{\epsilon^2}{\eta}\right)} \leq \frac{Z}{\hat{Z}}\left(1 + O\left(\frac{\epsilon^2}{\eta}\right)\right).$$

Similarly, it is easy to see that $\frac{Z}{\hat{Z}} \sim (1 + O(\epsilon^2/\eta))$ because,

$$\frac{Z}{\hat{Z}} = \frac{\int_\Omega e^{-\beta J}d\theta}{\int_\Omega e^{-\beta J}e^{-(\hat{\beta}-\beta)J}e^{-\hat{\beta}\tilde{J}}d\theta} = \frac{\int_\Omega e^{-\beta J}d\theta}{\int_\Omega e^{-\beta J}\left(1 + O\left(\frac{\epsilon^2}{\eta}\right)\right)^2 d\theta} = 1 + O\left(\frac{\epsilon^2}{\eta}\right).$$

Therefore,

$$\left\|\frac{\hat{p}^\infty}{p^\infty}\right\|_{L^\infty} \leq \left(1 + O\left(\frac{\epsilon^2}{\eta}\right)\right)^2 \leq 1 + O\left(\frac{\epsilon^2}{\eta}\right).$$

■

Appendix C. Proof of Theorem 4

Proof Letting $h(t,\theta) = \hat{p}(t,\theta) - p(t,\theta)$ and subtracting (4.4) from (4.5) leads to

$$\begin{aligned}\partial_t h =& \nabla\cdot\left[(\nabla J + \nabla\tilde{J})\hat{p} + \frac{1}{\hat{\beta}}\nabla\hat{p}\right] - \nabla\cdot\left[\nabla J p + \frac{1}{\beta}\nabla p\right]\\ =& \nabla\cdot\left[\nabla J h + \frac{1}{\beta}\nabla h\right] + \nabla\cdot\left[\nabla\tilde{J}\hat{p} + \left(\frac{1}{\hat{\beta}} - \frac{1}{\beta}\right)\nabla\hat{p}\right]\\ =& \nabla\cdot\left[p^\infty\nabla\left(\frac{h}{p^\infty}\right)\right] + \nabla\cdot\left[\nabla\tilde{J}\hat{p} + \left(\frac{1}{\hat{\beta}} - \frac{1}{\beta}\right)\nabla\hat{p}\right].\end{aligned}$$

Multiply $\frac{h}{p^\infty}$ to the above equation, then integrate it over θ, one has,

$$\frac{1}{2}\partial_t \|h\|_*^2 = \frac{h}{p^\infty}\left((\nabla J + \nabla\tilde{J})\hat{p} + \frac{1}{\hat{\beta}}\nabla\hat{p}\right)\cdot\boldsymbol{n}\Big|_{\partial\Omega} - \frac{h}{p^\infty}\left(\nabla Jp + \frac{1}{\beta}\nabla p\right)\cdot\boldsymbol{n}\Big|_{\partial\Omega}$$
$$- \int\left[\nabla\left(\frac{h}{p^\infty}\right)\right]^2 p^\infty d\theta - \int\left[\nabla\tilde{J}\hat{p} + \left(\frac{1}{\hat{\beta}} - \frac{1}{\beta}\right)\nabla\hat{p}\right]\cdot\nabla\left(\frac{h}{p^\infty}\right)d\theta.$$

The first two terms on the RHS vanish because of the reflecting boundary condition (4.6). Since $\nabla\tilde{J}, \tilde{\Sigma} \le O(\epsilon^2)$ have been shown in Lemma 1 and (4.7), this leads to the two coefficients of the last term can be bounded by $\left\|\nabla\tilde{J}\right\|_{L^\infty} \le C_1\epsilon^2$, $\frac{1}{\hat{\beta}} - \frac{1}{\beta} = \eta\tilde{\Sigma}/2 \le C_2\eta\epsilon^2$. Therefore, applying Young's inequality to the last term gives,

$$\begin{aligned}
&\int\left[\nabla\tilde{J}\hat{p} + \left(\frac{1}{\hat{\beta}} - \frac{1}{\beta}\right)\nabla\hat{p}\right]\cdot\nabla\left(\frac{h}{p^\infty}\right)d\theta \\
\le&\left\|\nabla\tilde{J}\right\|_{L^\infty}\int\left|\hat{p}\cdot\nabla\left(\frac{h}{p^\infty}\right)\right|d\theta + \left(\frac{1}{\hat{\beta}} - \frac{1}{\beta}\right)\int\left|\nabla\hat{p}\cdot\nabla\left(\frac{h}{p^\infty}\right)\right|d\theta \\
\le&\frac{1}{2}C_1\epsilon^2\left(\|\nabla\hat{p}\|_*^2 + \int\left[\nabla\left(\frac{h}{p^\infty}\right)\right]^2 p^\infty d\theta\right) + \frac{1}{2}C_2\eta\epsilon^2\left(\|\nabla\hat{p}\|_*^2 + \int\left[\nabla\left(\frac{h}{p^\infty}\right)\right]^2 p^\infty d\theta\right).
\end{aligned}$$

The third term can be bounded according to the Poincare Inequality (4.8), thus one has

$$\begin{aligned}
&\frac{1}{2}\partial_t\|h\|_*^2 \\
\le&-\frac{\lambda}{2}\|h\|_*^2 - \frac{1}{2}\int\left[\nabla\left(\frac{h}{p^\infty}\right)\right]^2 p^\infty d\theta + \frac{\epsilon^2}{2}\left(C_1\|\hat{p}\|_*^2 + C_2\eta\|\nabla\hat{p}\|_*^2 + (C_1 + C_2\eta)\int\left[\nabla\left(\frac{h}{p^\infty}\right)\right]^2 p^\infty d\theta\right) \\
\le&-\frac{\lambda}{2}\|h\|_*^2 + \frac{\epsilon^2}{2}\left(C_1\|\hat{p}\|_*^2 + \eta C_2\|\nabla\hat{p}\|_*^2\right).
\end{aligned} \tag{C.1}$$

Since we only consider the case when $\epsilon \ll 1$, so the coefficient of $\int\left[\nabla\left(\frac{h}{p^\infty}\right)\right]^2 p^\infty d\theta$, $-(1 - \epsilon^2(C_1 + C_2\eta))/2$, is always negative, which gives the last inequality of the above estimates.

Therefore, as long as $\|\hat{p}\|_*^2, \|\nabla\hat{p}\|_*^2$ are bounded, we can bound $\|h\|_*^2$ from (C.1). We prove the boundedness of $\|\hat{p}\|_*^2, \|\nabla\hat{p}\|_*^2$ in Lemma 5 and Lemma 6 in Appendix D and E. Then, we can bound the last term of (C.1) by

$$\frac{\epsilon^2}{2}\left(C_1\|\hat{p}\|_*^2 + \eta C_2\|\nabla\hat{p}\|_*^2\right) \le \frac{C}{2}\left\|\frac{\hat{p}^\infty}{p^\infty}\right\|_{L^\infty}\epsilon^2,$$

for some constant C. Hence from (C.1), we have,

$$\partial_t\left(e^{\lambda t}\|h\|_*^2\right) \le e^{\lambda t}\left(C\left\|\frac{\hat{p}^\infty}{p^\infty}\right\|_{L^\infty}\epsilon^2\right),$$
$$e^{\lambda t}\|h\|_*^2 - \|h(0)\|_*^2 \le \frac{1}{\lambda}(e^{\lambda t} - 1)C\left\|\frac{\hat{p}^\infty}{p^\infty}\right\|_{L^\infty}\epsilon^2.$$

Since $p(0,\theta) = \hat{p}(0,\theta)$, $h(0,\theta) = 0$. Therefore,

$$\|h\|_*^2 \leq \frac{1}{\lambda}(1-e^{-\lambda t})C\left\|\frac{\hat{p}^\infty}{p^\infty}\right\|_{L^\infty}\epsilon^2 \leq \left(1+O\left(\frac{\epsilon^2}{\eta}\right)\right)O(\epsilon^2).$$

■

Appendix D.

Lemma 5 *The solution to (4.5) is bounded*

$$\|\hat{p}\|_*^2 \leq C\left\|\frac{\hat{p}^\infty}{p^\infty}\right\|_{L^\infty},$$

with some constant C related to the initial data.

Proof We first prove that the difference of $\hat{p}$ and $\hat{p}^\infty$ is exponentially decay. Define norm $\|g\|_{\hat{*}}^2$ as follows,

$$\|g\|_{\hat{*}}^2 = \int g^2 \frac{1}{\hat{p}^\infty} d\theta,$$

where $\hat{p}^\infty$ is defined in Theorem 3. Let $g = \hat{p} - \hat{p}^\infty$, then g satisfies,

$$\partial_t g = \nabla \cdot \left[\hat{p}^\infty \nabla\left(\frac{g}{\hat{p}^\infty}\right)\right]. \tag{D.1}$$

Multiplying $\frac{g}{\hat{p}^\infty}$, and integrating it over θ, after integration by parts, one has

$$\frac{1}{2}\partial_t \|g\|_{\hat{*}}^2 = -\int \hat{p}^\infty \left[\nabla\left(\frac{g}{\hat{p}^\infty}\right)\right]^2 d\theta \leq -\lambda \|g\|_{\hat{*}}^2,$$

where the last inequality follows from the Poincare inequality (4.8) and the fact that $\int g d\theta = 1-1 = 0$. Solve the above ODE, one has,

$$\|g(t)\|_{\hat{*}}^2 \leq e^{-2\lambda t}\|g(0)\|_{\hat{*}}^2 \tag{D.2}$$

Therefore, one can bound $\hat{p}$ by

$$\begin{aligned}\|\hat{p}\|_*^2 = \left\|\hat{p}\sqrt{\frac{\hat{p}^\infty}{p^\infty}}\right\|_{\hat{*}}^2 &\leq \left\|\frac{\hat{p}^\infty}{p^\infty}\right\|_{L^\infty}\|\hat{p}\|_{\hat{*}}^2 \leq \left\|\frac{\hat{p}^\infty}{p^\infty}\right\|_{L^\infty}\left(\|\hat{p}-\hat{p}^\infty\|_{\hat{*}}^2 + \|\hat{p}^\infty\|_{\hat{*}}^2\right) \\ &\leq \left\|\frac{\hat{p}^\infty}{p^\infty}\right\|_{L^\infty}\left(e^{-2\lambda t}\|g(0)\|_{\hat{*}}^2 + 1\right).\end{aligned} \tag{D.3}$$

■

Appendix E.

Lemma 6 *The gradient of the solution to (4.5) is bounded*

$$\|\nabla \hat{p}\|_*^2 \leq C \left\| \frac{\hat{p}^\infty}{p^\infty} \right\|_{L^\infty},$$

with some constant C related to the initial data.

Proof Similar to (D.3) in the proof of Lemma 5, it is sufficient to prove this Lemma if we get the estimation for $\|\nabla g(t)\|_{\hat{*}}^2$ with $g = \hat{p} - \hat{p}^\infty$, because

$$\|\nabla \hat{p}\|_*^2 \leq \left\| \frac{\hat{p}^\infty}{p^\infty} \right\|_{L^\infty} \left(\|\nabla g\|_{\hat{*}}^2 + 1 \right) \tag{E.1}$$

First notice that reflecting boundary condition also holds for $\partial_{\theta_i} \hat{p}$, that is,

$$\partial_{\theta_i} \left((\nabla J + \nabla \tilde{J}) \hat{p} + \frac{\eta}{2} \nabla \cdot (\Sigma + \tilde{\Sigma} \hat{p}) \right) \cdot \boldsymbol{n} \Big|_{\partial\Omega} = 0,$$

Then take ∂_{θ_i} to (D.1), one has,

$$\partial_t \partial_{\theta_i} g = \nabla \cdot \left[\hat{p}^\infty \nabla \left(\frac{\partial_{\theta_i} g}{\hat{p}^\infty} \right) \right] + \nabla \cdot \left[\nabla \left(\partial_{\theta_i} \hat{J} \right) g \right].$$

Multiplying $\frac{\partial_{\theta_i} g}{\hat{p}^\infty}$, and summing it over i, integrating it over θ gives

$$\begin{aligned}
\frac{1}{2} \partial_t \|\nabla g\|_{\hat{*}}^2 = {} & \partial_{\theta_i} \left((\nabla J + \nabla \tilde{J}) g + \frac{\eta}{2} \nabla \cdot (\Sigma + \tilde{\Sigma} g) \right) \cdot \boldsymbol{n} \Big|_{\partial\Omega} \\
& - \sum_i \int \hat{p}^\infty \left[\nabla \left(\frac{\partial_{\theta_i} g}{\hat{p}^\infty} \right) \right]^2 d\theta - \sum_i \int \left[\nabla \left(\partial_{\theta_i} \hat{J} \right) g \right] \cdot \nabla \left(\frac{\partial_{\theta_i} g}{\hat{p}^\infty} \right) d\theta \\
\leq {} & - \frac{\lambda}{2} \sum_i \|\partial_{\theta_i} g\|_{\hat{*}}^2 - \frac{1}{2} \sum_i \int \hat{p}^\infty \left[\nabla \left(\frac{\partial_{\theta_i} g}{\hat{p}^\infty} \right) \right]^2 d\theta \\
& + \frac{1}{2} \sum_i \int \left[\nabla \left(\partial_{\theta_i} \hat{J} \right) g \right]^2 \frac{1}{\hat{p}^\infty} d\theta + \frac{1}{2} \sum_i \int \hat{p}^\infty \left[\nabla \left(\frac{\partial_{\theta_i} g}{\hat{p}^\infty} \right) \right]^2 d\theta \\
\leq {} & - \frac{\lambda}{2} \|\nabla g\|_{\hat{*}}^2 + \frac{1}{2} \sum_i \int \left[\nabla \left(\partial_{\theta_i} \hat{J} \right) g \right]^2 \frac{1}{\hat{p}^\infty} d\theta \leq - \frac{\lambda}{2} \|\nabla g\|_{\hat{*}}^2 + \frac{M}{2} \|g\|_{\hat{*}}^2,
\end{aligned}$$

where the second assumption in Assumption 2 is applied to obtain the last inequality. Use the estimation of $\|g\|_{\hat{*}}^2$ in (D.2) to solve the above ODE,

$$\begin{aligned}
& \partial_t \left(e^{\lambda t} \|\nabla g\|_{\hat{*}}^2 \right) \leq M e^{\lambda t} \left(e^{-2\lambda t} \|g(0)\|_{\hat{*}}^2 \right) \leq M e^{-\lambda t} \|g(0)\|_{\hat{*}}^2, \\
& e^{\lambda t} \|\nabla g\|_{\hat{*}}^2 - \|\nabla g(0)\|_{\hat{*}}^2 \leq M \|g(0)\|_{\hat{*}}^2 \frac{1}{\lambda} (1 - e^{-\lambda t}), \\
& \|\nabla g\|_{\hat{*}}^2 \leq e^{-\lambda t} \left(\|\nabla g(0)\|_{\hat{*}}^2 + \frac{M}{\lambda} \|g(0)\|_{\hat{*}}^2 \right).
\end{aligned}$$

Therefore, by (E.1), one has

$$\|\nabla \hat{p}\|_*^2 \leq \left\| \frac{\hat{p}^\infty}{p^\infty} \right\|_{L^\infty} \left(e^{-\lambda t} \left(\|\nabla g(0)\|_{\hat{*}}^2 + \frac{M}{\lambda} \|g(0)\|_{\hat{*}}^2 \right) + 1 \right).$$

■