Management Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Approximation Benefits of Policy Gradient Methods with Aggregated States

Daniel Russo

To cite this article:
Daniel Russo (2023) Approximation Benefits of Policy Gradient Methods with Aggregated States. Management Science

Published online in Articles in Advance 28 Jun 2023

. https://doi.org/10.1287/mnsc.2023.4788

https://pubsonline.informs.org/journal/mnsc

MANAGEMENT SCIENCE
Articles in Advance, pp. 1–14
ISSN 0025-1909 (print), ISSN 1526-5501 (online)

Approximation Benefits of Policy Gradient Methods with Aggregated States

Daniel Russo[a]

[a] Graduate School of Business, Columbia University, New York, New York 10027
Contact: djr2174@gsb.columbia.edu, https://orcid.org/0000-0001-5926-8624 (DR)

Received: January 7, 2021
Revised: June 22, 2022
Accepted: August 23, 2022
Published Online in Articles in Advance: June 28, 2023

https://doi.org/10.1287/mnsc.2023.4788

Abstract. Folklore suggests that policy gradient can be more robust to misspecification than its relative, approximate policy iteration. This paper studies the case of state-aggregated representations, in which the state space is partitioned and either the policy or value function approximation is held constant over partitions. This paper shows a policy gradient method converges to a policy whose regret per period is bounded by ϵ, the largest difference between two elements of the state-action value function belonging to a common partition. With the same representation, both approximate policy iteration and approximate value iteration can produce policies whose per-period regret scales as $\epsilon/(1-\gamma)$, where γ is a discount factor. Faced with inherent approximation error, methods that locally optimize the true decision objective can be far more robust.

History: Accepted by Hamid Nazerzadeh, data science.
Supplemental Material: Data are available at https://doi.org/10.1287/mnsc.2023.4788.

Keywords: reinforcement learning • approximate dynamic programming • policy gradient methods • state aggregation

1. Introduction

The fields of approximate dynamic programming and reinforcement learning (RL) offer algorithms that can produce effective performance in complex control problems in which large state spaces render exact computation intractable. Two of the classic methods in this area, approximate value iteration (AVI) and approximate policy iteration (API), modify value and policy iteration by fitting parametric approximation to the value function. In recent years, an alternative class of algorithms known as policy gradient methods has surged in popularity. These algorithms search directly over a parameterized subclass of policies by applying variants of gradient ascent to an objective function measuring the total expected reward accrued. An appealing feature of these methods is that, even though parametric approximations may introduce unavoidable error, they still directly optimize the true decision objective. AVI and API, by contrast, pick parameters by minimizing a measure of error in the value function approximation that may not be well-aligned with the decision objective.

Numerous papers find empirically that direct policy search eventually converges to superior policies. For example, consider a long sequence of works that apply approximate dynamic programming techniques to Tetris.[1] After Bertsekas and Ioffe (1996) applied approximate policy iteration to the problem, Kakade (2002), Szita and Lörincz (2006), and Furmston and Barber (2012) attained much higher scores using methods that directly search over a class of policies. This is not unique to Tetris. A similar phenomenon was observed in an ambulance redeployment problem by Maxwell et al. (2013) and a battery storage problem by Scott et al. (2014). Experiments with deep reinforcement learning tend to be less transparent, but policy gradient methods are extremely popular (see, e.g., Schulman et al. 2015, 2017b).

Kakade (2002), Szita and Lörincz (2006), Furmston and Barber (2012), Maxwell et al. (2013), and Scott et al. (2014) all search over the class of policies that are induced by (soft) maximization with respect to some parameterized value function. In a sense, these methods tune the parameters of the value function approximation but do so aiming to directly improve the total expected reward earned rather than to minimize a measure of prediction error. As a result, any gap in performance cannot be due to the approximation architecture and instead is caused by the procedure that sets the parameters.

There is very limited theory formalizing this phenomenon. Several works provide broad performance guarantees for each type of algorithm. In the case of API, Munos (2003), Antos et al. (2008), and Lazaric et al. (2012) build on the original analysis of Bertsekas and Tsitsiklis (1996). An intellectual foundation for studying policy gradient methods is laid by Kakade and Langford (2002), who analyze a conservative policy iteration algorithm (CPI). Scherrer and Geist (2014) observe that guarantees similar to those for CPI can be provided for some idealized policy gradient methods, and recently Agarwal et al. (2019b) develop approximation guarantees and convergence rates for a much broader class of policy gradient algorithms. There

are few lower bounds, but comparing available upper bounds is suggestive. The results for incremental algorithms such as CPI depend on a certain distribution shift term that is typically smaller than the so-called concentrability coefficients in Munos (2003), Antos et al. (2008), and Lazaric et al. (2012). See Scherrer (2014).

This paper provides a specialized study of algorithms that use state-aggregated representations, under which the state space is partitioned and either the policy or value function approximation does not distinguish between states in a common partition. State aggregation is a very old idea in approximate dynamic programming and reinforcement learning (Whitt 1978, Bean et al. 1987, Gordon 1995, Singh et al. 1995, Tsitsiklis and Van Roy 1996, Rust 1997, Li et al. 2006, Jiang et al. 2015, Abel et al. 2016), leading to tractable algorithms for problems with low-dimensional continuous state spaces in which it is believed that nearby states are similar. We measure the inherent error of a state aggregation procedure by the largest difference between two elements of the state-action value function belonging to a common partition, denoted ϵ_ϕ. (Here ϕ denotes a particular state aggregation.)

We show that any policy that is a stationary point of the policy gradient objective function has per-period regret less than ϵ_ϕ. Many variants of policy gradient algorithms, being first order methods, are ensured to converge (often efficiently) to stationary points, so this provides a guarantee on the quality of an ultimate policy produced with this approximation architecture. This guarantee is a substantial improvement over past work. The recent results of Bhandari and Russo (2019) translate into limiting per-period regret of $\kappa_\rho \epsilon_\phi/(1-\gamma)$, where γ is a discount factor and κ_ρ is a complex term that captures distribution shift. Critically, here, even per-period regret scales with the effective horizon. Other available bounds (Kakade and Langford 2002, Scherrer and Geist 2014, Agarwal et al. 2019b) are at least as bad.[2] Building on an example of Bertsekas and Tsitsiklis (1996), we give an example in which API produces policies whose per-period regret scales as $\epsilon_\phi/(1-\gamma)$, hence establishing formally that policy gradient methods converge to a far better policy with the same approximation architecture.

The large performance gap between API and policy gradient is surprising because they are known to be closely related (Konda and Tsitsiklis 2000, Sutton et al. 2000). There is a particularly precise and simple connection in the case of state aggregation: Theorem 8 shows that a Frank–Wolfe (Frank and Wolfe 1956) policy gradient method is equivalent to a version of API that (i) estimates an approximate value function by minimizing a loss function that weighs errors at states in proportion to how often those states are visited under the current policy and (ii) makes soft or local updates to the policy in each iteration. With these modifications, each iteration of API locally optimizes a first order approximation to the true decision objective, leading to much greater robustness to approximation errors. See Section 6. Section 5 also studies a version of API that makes only the first change—using an on-policy state weighting—but not the second. A different counterexample (Example 2) is constructed to show that this variant can be as brittle as a standard version of API that makes neither change.

The limited scope of this work should be highlighted. We have in mind settings in which policy gradient is applied in simulation, which is how they are currently employed in nearly all applications. In this case, it is feasible to employ an exploratory initial distribution as is assumed in this work. This choice can have a critical impact on the optimization landscape (see, e.g., Agarwal et al. 2019b). By focusing on the quality of stationary points, the paper sidesteps issues of (i) optimization error (i.e., error because of executing only a finite number of gradient steps), (ii) statistical error (i.e., error because of estimating gradients with limited simulated rollouts), and (iii) precisely which policy gradient variant and step sizes are employed. The treatment of API is similarly stylized. This choice allows for a crisp presentation focused on one insight that is missing in the current literature.

1.1. Further Discussion of Related Literature

Dong et al. (2019) prove that a state-aggregated and optimistic variant of Q-learning efficiently approaches limiting per-period regret smaller than a measure of inherent aggregation error similar to ϵ_ϕ. Van Roy (2006) previously shows that approximate value iteration with fixed state-relevance weights could suffer per-period regret that scales with the effective time horizon. Van Roy (2006) also observes that the robustness of policies derived from solutions of state-aggregated Bellman equations can depend critically on the choice of state-relevance weights. Whereas this paper studies different algorithms and gives proofs that bear little resemblance to Van Roy (2006) and Dong et al. (2019), their study of the robustness difference between two closely related algorithms inspired my own.

A number of recent works study the convergence rates of policy gradient methods in Markov decision processes (MDPs) with finite state and action spaces under the assumption that the policy class can represent all stochastic policies. Examples include Agarwal et al. (2019b), Mei et al. (2020), Zhang et al. (2021), Cen et al. (2021), Bhandari and Russo (2021), and Khodadadian et al. (2021). The fastest convergence rates seem to be attained by policy gradient variants that behave just like policy iteration (Bhandari and Russo 2021). Such theory, therefore, does not offer insight into the advantages of policy gradient methods over classic algorithms. This paper helps close that intellectual gap in the literature.

In a fascinating paper, Scherrer and Lesner (2012) observe that the horizon dependence of approximate value and policy iteration can be improved by modifying them to use nonstationary policies. It is unclear if there is

a connection between their work and this paper's finding about policy gradient methods for optimizing over the class of stationary policies.

What is called state aggregation in this paper is sometimes called "hard" state aggregation. Generalizations allow a state to have an affinity or partial association with several different regions of the state space. Singh et al. (1995) propose a soft state aggregation method, which avoids discontinuities in the approximate value function at the boundaries of partitions. The theory of Tsitsiklis and Van Roy (1996) holds for a related class of approximations they call interpolators. A popular approximation method called tile coding, which is closely related to state aggregation, is discussed in the textbook of Sutton and Barto (2018). This paper does not pursue such extensions, focusing on the simplest variant of state aggregation. The current proof significantly relies on the structure of hard state aggregation.

The current paper assumes a state-aggregation rule is fixed and given, then studying the quality of the policies various algorithms produce with this approximation. A long series of works focus on how a state aggregation can be learned or adaptively constructed (Bertsekas and Castanon 1989, Singh et al. 1995, Dean and Givan 1997, Jiang et al. 2015, Duan et al. 2019, Misra et al. 2020).

The bounds for policy gradient provided in this work depend on a notion of maximal error because of state aggregation (see Definition 2). Contemporaneously, Agarwal et al. (2020) looked at policy gradient with state aggregated representations. Dependence on the time horizon is not a focus of their work. Instead, they focus on providing an upper bound that depends on a softer notion of approximation error that averages across partitions. It seems that the two measures coincide in Example 1, but the measure of Agarwal et al. (2020) can offer an important improvement in other examples. Their upper bounds are not matched by those previously established for alternative algorithms and are suggestive of another provable benefit of using policy gradient methods in this setting.

2. Problem Formulation

We consider a Markov decision process $M = (\mathcal{S}, \mathcal{A}, r, P, \gamma, \rho)$, which consists of a finite state space $\mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$, finite action space $\mathcal{A} = \{1, \ldots, |\mathcal{A}|\}$, reward function r, transition kernel P, discount factor $\gamma \in (0,1)$, and initial distribution ρ. For any finite set $\mathcal{X} = \{1, \ldots, |\mathcal{X}|\}$, we let $\Delta(\mathcal{X}) = \{d \in \mathbb{R}_+^{|\mathcal{X}|} : \sum_{x\in\mathcal{X}} d_x = 1\}$ denote the set of probability distributions over $\mathcal{X}$. A stationary randomized policy is a mapping $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. We use $\pi(s,a)$ to denote the ath component of $\pi(s)$. Let Π denote the set of all stationary randomized policies. Conditioned on the history up to that point, an agent who selects action a in state $s \in \mathcal{S}$ earns a reward in that period with expected value $r(s, a)$ and transitions randomly to a next state, in which $P(s'|s,a)$ denotes the probability of transitioning to state $s' \in \mathcal{S}$. To treat randomized policies, we overload notation, defining for $d \in \Delta(\mathcal{A})$, $r(s,d) = \sum_{a=1}^{|\mathcal{A}|} r(s,a)d_a$ and $P(s'|s,d) = \sum_{a=1}^{|\mathcal{A}|} P(s'|s,a)d_a$. Notice that, if $e_a \in \Delta(\mathcal{A})$ is the ath standard basis vector, then $r(s,e_a) = r(s,a)$.

2.1. Value Functions and Bellman Operators

We define, respectively, the value function associated with a policy π and the optimal value function by

$$V_\pi(s) = \mathbb{E}_s^\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right], \qquad V^*(s) = \max_{\pi\in\Pi} V_\pi(s).$$

The notation $\mathbb{E}_s^\pi[\cdot]$ denotes expectations taken over the sequence of states when $s_0 = s$ and policy π is applied. A policy π^* is said to be optimal if $V_{\pi^*}(s) = V^*(s)$ for every $s \in \mathcal{S}$. It is known that an optimal deterministic policy exists. Throughout this paper, we use π^* to denote some optimal policy. There could be multiples, but this does not change the results. The Bellman operator $T_\pi : \mathbb{R}^n \to \mathbb{R}^n$ associated with a policy $\pi \in \Pi$ maps a value function $V \in \mathbb{R}^n$ to a new value function $T_\pi V \in \mathbb{R}^n$ defined by $(T_\pi V)(s) = r(s,\pi(s)) + \gamma\sum_{s'\in\mathcal{S}} P(s'|s,\pi(s))V(s')$. The Bellman optimality operator $T : \mathbb{R}^n \to \mathbb{R}^n$ is defined by

$$TV(s) = \max_{\pi\in\Pi}(T_\pi V)(s) = \max_{d\in\Delta(\mathcal{A})} r(s,d) + \gamma\sum_{s'\in\mathcal{S}} P(s'|s,d)V(s').$$

It is well-known that T and T_π are contraction mappings with respect to the maximum norm. Their unique fixed points are V^* and V_π, respectively. For a state $s \in \mathcal{S}$, policy $\pi \in \Pi$, and action distribution $d \in \Delta(\mathcal{A})$, define the state-action value function $Q_\pi(s,d) = r(s,d) + \gamma\sum_{s'\in\mathcal{S}} P(s'|s,d) V_\pi(s)$, which measures the expected total discounted reward of sampling an action from d in state s and applying π thereafter. When d is deterministic, meaning $d_a = 1$ for some $a \in \mathcal{A}$, we denote this simply by $Q_\pi(s,a)$. Define $Q^*(s,d) = Q_{\pi^*}(s,d)$ for some optimal policy π^*. These obey the relations

$$Q_\pi(s,\pi'(s)) = (T_{\pi'}V_\pi)(s) \quad \max_{d\in\Delta(\mathcal{A})} Q_\pi(s,d) = (TV_\pi)(s). \tag{1}$$

2.2. Geometric Average Rewards and Occupancies

Policy gradient methods are first order optimization algorithms applied to optimize a scalar objective that measures expected discounted reward earned from a random initial state given by

$$J(\pi) = (1-\gamma)\sum_{s\in\mathcal{S}} \rho(s)V_\pi(s). \tag{2}$$

Another critical object is the discounted state occupancy measure

$$\eta_\pi = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \rho P_\pi^t \in \Delta(\mathcal{S}),$$

where $P_\pi \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ is the Markov transition matrix under policy π and $\rho \in \mathbb{R}^{|\mathcal{S}|}$ is viewed as a row vector.

Here, $\eta_\pi(s)$ gives the geometric average time spent in state s when the initial state is drawn from ρ. These two are related as $J(\pi) = \sum_{s\in\mathcal{S}} \eta_\pi(s) r(s,\pi(s))$.

The factor of $(1-\gamma)$ in the definitions of η_π and $J(\pi)$ serves to normalize these quantities and gives them a natural interpretation in terms of average reward problems. In particular, consider, just for the moment, a problem with modified transition probabilities $\tilde{P}(s'|s,a) = (1-\gamma)\rho(s') + \gamma P(s'|s,a)$. That is, in each period, there is a $1-\gamma$ chance that the system resets in a random state drawn from ρ. Otherwise, the problem continues with the next state drawn according to P. One can show that $J(\pi)$ denotes the average reward earned by π and $\eta_\pi(s)$ is average fraction of time spent in state s under policy π in this episodic problem. Undiscounted average reward problems are often constructed by studying $J(\pi)$ as the discount factor approaches one (Bertsekas 1995, Puterman 2014).

3. State Aggregation

A state aggregation is defined by a function $\phi : \mathcal{S} \to \{1,\ldots,m\}$ that partitions the state space into m segments. We call $\phi^{-1}(j) = \{s \in \mathcal{S} : \phi(s) = j\}$ the jth segment of the partition. Typically, we have in mind problems in which the state space is enormous (effectively infinite), but it is tractable to store and loop over vectors of length m. Tractable algorithms can then be derived by searching over approximate transition kernels, value functions, or policies that don't distinguish between states belonging to a common segment. Our hope is that states in a common segment are sufficiently similar, for example, because of smoothness in the transition dynamics and rewards so that these approximations still allow for effective decision making.

To make this idea formal, let us define the set of approximate value functions and policies induced by a state aggregation ϕ,

$$\begin{aligned}\mathcal{Q}_\phi &= \{Q \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|} : Q(s,a) = Q(s',a) \text{ for all } a \in \mathcal{A},\\ &\qquad \text{and all } s,s' \in \mathcal{S} \text{ such that } \phi(s) = \phi(s')\}\\ \Pi_\phi &= \{\pi \in \Pi : \pi(s) = \pi(s') \text{ for all } s,s' \in \mathcal{S}\\ &\qquad \text{such that } \phi(s) = \phi(s')\}.\end{aligned}$$

It should be emphasized that practical algorithms do not require, for example, actually storing $|\mathcal{S}| \cdot |\mathcal{A}|$ numbers in order to represent an element $Q \in \mathcal{Q}_\phi \subset \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$. Instead, one stores just $m \cdot |\mathcal{A}|$ numbers, one per segment.

Should we approximate the value function or the policy? In this setting, there is a broad equivalence. This is important to the interpretation of the paper's results because it implies that any benefit of policy gradient methods is not attributable to its approximation architecture, but instead is attributable to the way it searches over the parameters of an approximation. A complete proof of these statements is omitted, but some details are given Appendix B.

Remark 1 (Equivalence of Aggregated-State Approximations). The set of randomized state-aggregated policies Π_ϕ is equal to the set of policies formed by softmax optimization with respect to state-aggregated value functions:

$$\begin{aligned}\Pi_\phi = \text{closure}\{\pi \in \Pi : Q \in \mathcal{Q}_\phi,\ \ \pi(s,a)\\ = e^{Q(s,a)} \Big/ \sum_{a'\in\mathcal{A}} e^{Q(s,a')} \quad \forall s \in \mathcal{S}, a \in \mathcal{A}\}.\end{aligned} \tag{3}$$

Moreover, the set of deterministic policies contained in Π_ϕ is isomorphic to

$$\{f \in \mathcal{A}^{|\mathcal{S}|} : Q \in \mathcal{Q}_\phi, f(s) = \min\left\{\operatorname*{argmax}_{a\in\mathcal{A}} Q(s,a)\right\}\right\}, \tag{4}$$

the set of greedy policies[3] with respect to some state-aggregated value function.

4. Approximation via Policy Gradient with State-Aggregated Policy Classes

4.1. Convergence to Stationary Points

Policy gradient methods are first order optimization methods applied to maximize $J(\pi)$ over the constrained policy class Π_ϕ. Of course, just as there is an ever-growing list of first order optimization procedures, there are many policy gradient variants. How do we provide insights relevant to this whole family of algorithms? Were $J(\pi)$ concave, we would expect that sensible optimization method to converge to the solution of $\max_{\pi\in\Pi_\phi} J(\pi)$, allowing us to abstract away the details of the optimization procedure and study instead the quality of decisions that can be made using a certain constrained policy class. Unfortunately, $J(\pi)$ is nonconcave (Agarwal et al. 2019b, Bhandari and Russo 2019). It is, however, smooth (see Lemma 1). In smooth nonconcave optimization, we expect sensible first order methods to converge to a first order stationary point (Bertsekas 1997). Studying the quality of policies that are stationary points of $J(\cdot)$ then gives broad insight into how the use of restricted policy classes affects the limiting performance reached by policy gradient methods.

As defined, a policy is a first order stationary point if, based on a first order approximation to $J(\cdot)$, there is no feasible direction that improves the objective value. Local search algorithms generally continue to increase the objective value until reaching a stationary point. A first order stationary point could be a local maximum or a saddle point. The latter presents no additional complications for applying Theorem 3. Throughout this section, we view each $\pi \in \Pi$ as a stacked vector $\pi = (\pi(s,a) : s \in \mathcal{S}, a \in \mathcal{A}) \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$. It may also be natural to view π as an $|\mathcal{S}| \times |\mathcal{A}|$ dimensional matrix whose rows are probability vectors. In that case, all results are equivalent if one views inner products as the standard inner product on square matrices given by $\langle A, B\rangle = \text{Trace}(A^\top B)$ and all norms as the Frobenius norm.

Definition 1. A policy $\pi \in \Pi$ is a first order stationary point of $J : \Pi \to \mathbb{R}$ on the subset $\Pi_\phi \subset \Pi$ if

$$\langle \nabla J(\pi), \pi' - \pi \rangle \leq 0 \qquad \forall \pi' \in \Pi_\phi.$$

The following smoothness result is shown by a short calculation[4] in Agarwal et al. (2019b).

Lemma 1. *For every* $\pi, \pi' \in \Pi$, $\|\nabla J(\pi) - \nabla J(\pi')\|_2 \leq L \|\pi - \pi'\|_2$, *where* $L = \frac{2\gamma|\mathcal{A}|\|r\|_\infty}{(1-\gamma)^2}$.

We've claimed that we expect first order methods applied to smooth nonconvex optimization are expected to converge to first order stationary points. This is a standard subject in nonlinear optimization (see, e.g., Bertsekas 1997), and the recent literature proposes stochastic first order methods with fast convergence rates to stationary points (see e.g. Ghadimi and Lan 2013, 2016; Defazio et al. 2014; Xiao and Zhang 2014; Reddi et al. 2016a, b, c; Davis and Grimmer 2019). As an illustration, we show a convergence result for an idealized policy gradient method with exact gradient evaluations and a direct parameterization. Armed with such a result, we focus on the quality of stationary points.

Recall that a point π_∞ is a limit point of a sequence if some subsequence converges to π_∞. Bounded sequences have convergent subsequences, so limit points exist for the sequence $\{\pi_t\}$ in Lemma 2. The operator $\text{Proj}_{2,\Pi_\phi}(\pi) = \text{argmin}_{\pi' \in \Pi_\phi} \|\pi' - \pi\|_2^2$ denotes orthogonal projection onto the convex set Π_ϕ. This exact lemma statement can be found in Bhandari and Russo (2019), and similar statements appear in nonlinear optimization textbooks (Bertsekas 1997, Beck 2017).

Lemma 2 (Convergence to Stationary Points). *For any* $\pi_1 \in \Pi$ *and* $\alpha \in \left(0, \frac{1}{L}\right]$, *let*

$$\pi_{t+1} = \text{Proj}_{2,\Pi_\phi}(\pi_t + \alpha \nabla J(\pi_t)) \qquad t = 1, 2, 3 \ldots \tag{5}$$

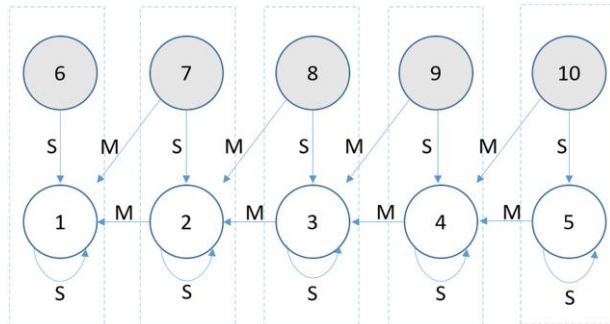
If π_∞ *is a limit point of* $\{\pi_t : t \in \mathbb{N}\}$, *then* π_∞ *is a stationary point of* $J(\cdot)$ *on* Π_ϕ *and*

$$\lim_{t \to \infty} J(\pi_t) = J(\pi_\infty).$$

Remark 2 (Steepest Feasible Ascent). It is well-known (see, e.g., Beck 2017) that policies generated by (5) satisfy $\pi^{t+1} = \text{argmin}_{\pi \in \Pi_\phi}(\pi^t + \langle \nabla J(\pi^t), \pi - \pi^t \rangle + \frac{1}{2\alpha}\|\pi - \pi^t\|_2^2)$. When the step size α is small, a projected gradient update essentially moves in the steepest feasible ascent direction until reaching a stationary point from which there are no feasible ascents.

Remark 3 (Practical Implementation). The appendix provides many extra details related to this algorithm. It explains that this projection can be computed using simple soft-thresholding operations and the whole algorithm can be implemented efficiently when storing only a parameter $\theta \in \mathbb{R}^{m \cdot |\mathcal{A}|}$. This stores one value per state segment and action rather than one per state. The appendix also shows how to generate unbiased stochastic gradients of $J(\cdot)$. The body of this paper instead focuses on the quality of the stationary points of $J(\cdot)$, abstracting away the specifics of which policy gradient method is used.

Figure 1. (Color online) A Bad Example for API



Note. The actions `move` and `stay` are denoted by M and S.

4.2. Quality of Stationary Points

We measure the accuracy of a state-aggregation $\phi(\cdot)$ through the maximal difference between state-action values with states belonging to the same segment of the state space. This notion is weaker than alternatives that explicitly assume transition probabilities, and rewards are uniformly close within segments—usually by imposing a smoothness condition (see, e.g., Rust 1996). But it is a stronger requirement than the recent one in Dong et al. (2019), which only looks at the gap between state-action values under the optimal value function. The current proof requires bounding the aggregation error of some policy π_∞ that is a stationary point of $J(\cdot)$, so it does not seem possible to give guarantees for policy gradient methods if we replace Q_π with Q^* in the following definition. At the same time, performance bounds that depend on ϵ_ϕ can be quite conservative. See, for instance, Figures 2 and 4. It is an open question whether this definition can be relaxed in a meaningful way. Li et al. (2006) provides a comparison of different measures of the approximation error of a state-aggregated representation.

Figure 2. (Color online) Performance of Approximate Policy Iteration and Policy Gradient in Example 1 with $n = 200$ States, Discount Factor $\gamma = 0.99$ and $\epsilon_\phi = 1$

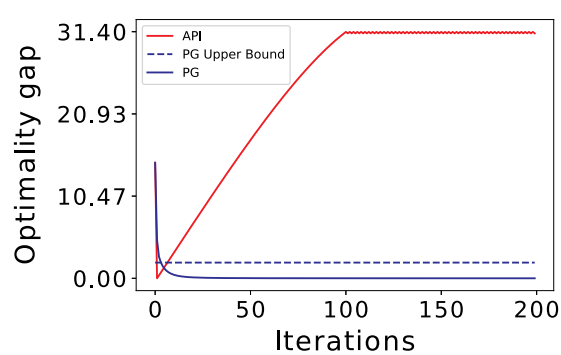
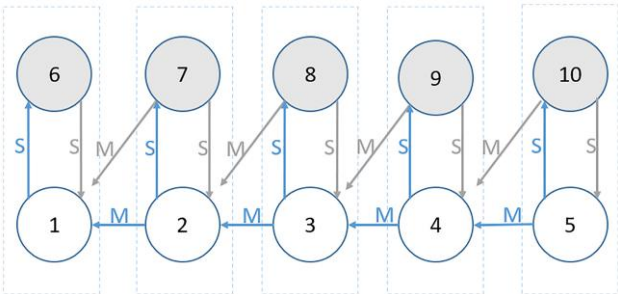


Figure 3. (Color online) A Bad Example for API with Adaptive Weights



Note. The actions `move` and `stay` are denoted by M and S.

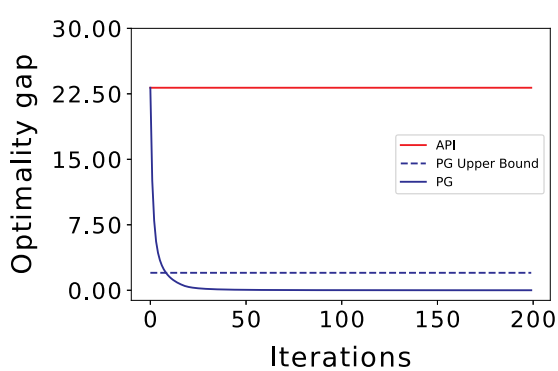
Definition 2 (Inherent State Aggregation Error). Let $\epsilon_\phi \in \mathbb{R}$ be the smallest scalar satisfying

$$|Q_\pi(s,a) - Q_\pi(s',a)| \le \epsilon_\phi$$

for every $\pi \in \Pi_\phi$, $a \in \mathcal{A}$ and all $s, s' \in \mathcal{S}$ such that $\phi(s) = \phi(s')$.

Despite the nonconcavity of $J(\cdot)$, one can give guarantees on the quality of its stationary points. The next result does so under the requirement that each state-space segment has positive probability under the initial weighting ρ. Similar assumptions appear in Bhandari and Russo (2019) and Agarwal et al. (2019b), and they each discuss its necessity at some length. Whereas the next result holds for ρ that is nearly degenerate, it should be emphasized that the convergence rates of many policy gradient methods depend inversely on $\min_i \rho(\phi^{-1}(i))$. An exploratory initial distribution is critical to these algorithm's practical success. Recall that $J(\cdot)$, as defined in (2), is normalized so that it represents the average rather than cumulative reward earned. Recall also that $\pi^* \in \Pi$ denotes some optimal policy, which, by definition, satisfies $V_{\pi^*}(s) = V^*(s) \quad \forall s \in \mathcal{S}$. Such a π^* is also an unconstrained maximizer of the policy gradient objective $J(\cdot)$.

Figure 4. (Color online) Performance of Algorithm 2 and Policy Gradient in Example 2 with $n = 400$ States, Discount Factor $\gamma = 0.99$, $\epsilon_\phi = 1$, and $c = 1/3$



Note. The initial distribution has the form $\rho(s) = C \cdot 20s$ and $\rho(s+m) = C \cdot s$, where $s \in \{1, \ldots, m\}$ and C is a normalizing constant.

Theorem 3 (Quality of Stationary Points). *Suppose $\rho(\phi^{-1}(i)) > 0$ for each $i \in \{1, \ldots, m\}$. If π_∞ is a stationary point of $J(\cdot)$ on Π_ϕ, then*

$$J(\pi^*) - J(\pi_\infty) \le (1-\gamma)\|V_{\pi_\infty} - V^*\|_\infty \le 2\epsilon_\phi.$$

For purposes of comparison, let us provide an alternative result, which can be derived by specializing a result in Bhandari and Russo (2019). At the time when this paper was initially written and posted, the following result was the best available bound. See the subsequent remark for a detailed comparison with a contemporaneous statement by Agarwal et al. (2019b). See the conclusion for discussion around how the special structure of state aggregation seems to drive the improvements in Theorem 3.

Theorem 4 (Earlier Result by Bhandari and Russo (2019)). *If π_∞ is a stationary point of $J(\cdot)$ on Π_ϕ, then*

$$J(\pi^*) - J(\pi_\infty) \le \kappa_\rho \frac{\epsilon_\phi}{(1-\gamma)},$$

where

$$\kappa_\rho \le \max_{i \in \{1,\ldots,m\}} \frac{\eta_{\pi^*}(\phi^{-1}(i))}{\rho(\phi^{-1}(i))}.$$

Here, κ_ρ captures whether the weight the initial distribution ρ places on each segment of the state partition is aligned with the occupancy measure under an optimal policy. The form here is somewhat stronger than the simple one in Kakade and Langford (2002), which does not aggregate across segments, but it is still problematic. Without special knowledge about the optimal policy, it is impossible to guarantee that κ_ρ is smaller than the number of segments m. Worse perhaps is the dependence on the effective horizon $1/(1-\gamma)$. Recall from Section 2 that $J(\pi) \in [0,1]$ has the interpretation of a geometric average reward per decision. The optimality gap $J(\pi^*) - J(\pi_\infty)$ then represents a kind of average per-decision regret produced by a limiting policy. The dependence on $1/(1-\gamma)$ on the right-hand side is then highly problematic, suggesting performance degrades entirely in a long-horizon regime. Whereas undesirable, this horizon dependence is unavoidable under some classic approximate dynamic programming procedures. This is shown for approximate value iteration (with a fixed state-weighting) by Van Roy (2006). In the next section, we show this is true for approximate policy iteration as well.

Remark 4 (Strengthened Results in Agarwal et al. (2019b)). Initially, a similar result to Theorem 4 could be found in Agarwal et al. (2019b) although with an even worse dependence on the discount factor. However, contemporaneously with this paper, their paper's results were updated and stated in terms of a notion called transfer error. Specializing this bound to our setting shows that the limiting optimality gap is bounded by $2\epsilon_\phi\sqrt{k}$ under a state-aggregated natural policy gradient method. That

bound avoids poor dependence on the problem's time horizon. However, relative to Theorem 3, it still has a poor dependence on the number of actions. Such an upper bound cannot be compared cleanly against the lower bound for API that we provide in Theorem 7, so for the purposes of this paper, the tighter result in Theorem 3 is critical.

Proof of Theorem 3. The next lemma is a version of the policy gradient theorem (Sutton and Barto 2018) that applies with directly parameterized policies. It is easy to deduce this formula from ones in Agarwal et al. (2019b) for example. The inner product interpretation in the statement is inspired by Konda and Tsitsiklis (2000). For any given state-relevance weights $w \in \mathbb{R}^{|\mathcal{S}|}$, define the inner product $\langle \cdot, \cdot \rangle_{w\times 1}$ on $\mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ by

$$\langle Q, Q' \rangle_{w\times 1} = \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} w(s) Q(s,a) Q'(s,a). \tag{6}$$

□

Lemma 5 (Policy Gradient Theorem for Directional Derivatives). *For each* $\pi, \pi' \in \Pi$,

$$\begin{aligned}\langle \nabla J(\pi), \pi' - \pi \rangle &= \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \eta_\pi(s) Q_\pi(s,a)(\pi'(s,a) - \pi(s,a)) \\ &= \langle Q_\pi, \pi' - \pi \rangle_{\eta_\pi \times 1}. \end{aligned} \tag{7}$$

Proof Sketch. The proof sketch here gives insight into the second order remainder error term in a first order Taylor expansion of $J(\cdot)$. We have

$$\begin{aligned} J(\pi') - J(\pi) &= \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \eta_{\pi'}(s)(\pi'(s,a) - \pi(s,a)) Q_\pi(s,a) \\ &= \langle Q_\pi, \pi' - \pi \rangle_{\eta_\pi \times 1} \\ &\quad + \underbrace{\sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} (\eta_{\pi'}(s) - \eta_\pi(s))(\pi'(s,a) - \pi(s,a)) Q_\pi(s,a)}_{=O(\|\pi'-\pi\|^2)}. \end{aligned} \tag{8}$$

The first equality is a simple but powerful result known in the RL literature as the performance difference lemma (Kakade and Langford 2002). That the remainder term is second order uses that $\pi \longmapsto P_\pi$ is linear, and therefore, $\eta_\pi = (1-\gamma)\rho(I - \gamma P_\pi)^{-1}$ is differentiable in π. □

Because $Q(s,d) = \sum_a Q(s,a) d_a$ for any action distribution $d \in \Delta(\mathcal{A})$, this formula can be written as $\langle \nabla J(\pi), \pi' - \pi \rangle = \mathbb{E}_{S\sim\eta_\pi}[Q_\pi(S, \pi'(S)) - Q_\pi(S, \pi(S))]$. One can interpret $Q_\pi(s,a)$ as measuring the benefit of switching from π to action a for a single period. The policy gradient Equation (7) says that the infinite-horizon impact of a local policy change in the direction of π' is equal to the average benefit of switching to policy π' for a single period and at a random state.

The next lemma is a special case of one in Bhandari and Russo (2019). This simplified setting allows for an extremely simple proof, so we include it for completeness. Equation (9) can be viewed as an approximate Bellman equation within the restricted class of policies.

Lemma 6 (An Approximate Bellman Equation for Stationary Points). *If* π_∞ *is a stationary point of* $J(\cdot)$ *on* Π_ϕ, *then*

$$\mathbb{E}[V_{\pi_\infty}(S)] = \max_{\pi\in\Pi_\phi} \mathbb{E}[T_\pi V_{\pi_\infty}(S)] \quad \textit{where } S \sim \eta_{\pi_\infty}. \tag{9}$$

Proof. Continue to let S denote a random draw from η_{π_∞}. For every $\pi \in \Pi_\phi$, we have

$$\begin{aligned} 0 \geq \langle \nabla J(\pi_\infty), \pi - \pi_\infty \rangle &= \langle Q_{\pi_\infty}, \pi - \pi_\infty \rangle_{\eta_{\pi_\infty}\times 1} \\ &= \mathbb{E}[Q_{\pi_\infty}(S, \pi(S))) - Q_{\pi_\infty}(S, \pi_\infty(S))] \\ &= \mathbb{E}[(T_\pi V_{\pi_\infty})(S) - V_{\pi_\infty}(S)]. \end{aligned}$$

The second equality uses (1). The reverse inequality uses that $\pi_\infty \in \Pi_\phi$ along with the Bellman equation $V_{\pi_\infty} = T_{\pi_\infty} V_{\pi_\infty}$. □

We are now ready to prove Theorem 3.

Proof of Theorem 3. We apply Lemma 6 and several times use the connection between Q functions and Bellman operators in (1). For notational convenience, throughout let S denote a random draw from η_{π_∞} and let $\mathcal{S}_i = \phi^{-1}(i)$ denote the ith segment of the state space. Because $\mathbb{E}[T_{\pi_\infty} V_{\pi_\infty}(S)] = \max_{\pi\in\Pi_\phi} \mathbb{E}[T_\pi V_{\pi_\infty}(S)]$, we have

$$\begin{aligned} \pi_\infty &\in \operatorname*{argmax}_{\pi\in\Pi_\phi} \mathbb{E}[T_\pi V_{\pi_\infty}(S)] = \operatorname*{argmax}_{\pi\in\Pi_\phi} \mathbb{E}[Q_{\pi_\infty}(S, \pi(S))] \\ &= \operatorname*{argmax}_{\pi\in\Pi_\phi} \sum_{i=1}^m \mathbb{E}[Q_{\pi_\infty}(S, \pi(S)) | S \in \mathcal{S}_i] \mathbb{P}(S \in \mathcal{S}_i). \end{aligned}$$

Let a_i^∞ denote the action selected by policy π_∞ at any state $s \in \phi^{-1}(i)$ in segment i. The vector $(a_1^\infty, \ldots, a_m^\infty)$ provides a full description of the policy π_∞. The optimization problem decomposes across segments of the state space, implying that

$$a_i^\infty \in \operatorname*{argmax}_{a\in\mathcal{A}} \mathbb{E}[Q_{\pi_\infty}(S,a) | S \in \mathcal{S}_i] \qquad i = 1, \ldots, m.$$

Here, we use implicitly that $\mathbb{P}(S \in \mathcal{S}_i) > 0$, which is assured by our assumption that $\rho(\mathcal{S}_i) > 0$. Now, we use the definition of ϵ_ϕ to show a_i^∞ must be near optimal at every state in partition i. Pick

$$(s_i^*, a_i^*) \in \operatorname*{argmax}_{s\in\mathcal{S}_i, a\in\mathcal{A}} Q_{\pi_\infty}(s,a).$$

By the optimality of a_i^∞, there must exist some $\tilde{s} \in \mathcal{S}_i$ such that $Q_{\pi_\infty}(\tilde{s}, a_i^\infty) \geq Q_{\pi_\infty}(\tilde{s}, a_i^*)$. For any other $s \in \mathcal{S}_i$, we have

$$\begin{aligned} Q_{\pi_\infty}(s, a_i^\infty) &\geq Q_{\pi_\infty}(\tilde{s}, a_i^\infty) - \epsilon_\phi \geq Q_{\pi_\infty}(\tilde{s}, a_i^*) - \epsilon_\phi \\ &\geq Q_{\pi_\infty}(s_i^*, a_i^*) - 2\epsilon_\phi \geq \max_{a\in\mathcal{A}} Q_{\pi_\infty}(s,a) - 2\epsilon_\phi. \end{aligned}$$

Observe that $Q_{\pi_\infty}(s, a_i^\infty) = Q_{\pi_\infty}(s, \pi_\infty(s)) = V_{\pi_\infty}(s)$ and $\max_{a\in\mathcal{A}} Q_{\pi_\infty}(s,a) = TV_{\pi_\infty}(s)$. Because s is arbitrary, this

gives element-wise inequality $V_{\pi_\infty} \geq TV_{\pi_\infty} - 2\epsilon_\phi e$, where e denotes a vector of ones. Using the monotonicity of Bellman operators and the fact that $T(V + ce) = TV + \gamma ce$ (Bertsekas 1995), we have

$$V_{\pi_\infty} \geq TV_{\pi_\infty} - 2\epsilon_\phi e \geq T^2 V_{\pi_\infty} + 2\gamma\epsilon_\phi e - 2\epsilon_\phi e$$
$$\geq \cdots \geq V^* - \frac{2\epsilon_\phi}{1-\gamma} e. \quad \square$$

5. Horizon-Dependent per-Period Regret Under API

Approximate policy iteration is one of the classic approximate dynamic programming algorithms. It has deep connections to popular methods today, such as Q-learning with target networks that are infrequently updated (Mnih et al. 2015). Approximate policy iteration is presented in Algorithm 1. The norm there is the one induced by the inner product in (6) defined by $\|Q\|_{2,w\times 1} = \sqrt{\sum_s \sum_a w(s) Q(s,a)^2}$. The procedure mimics the classic policy iteration algorithm (Puterman 2014) except it uses a regression-based approximation in the policy evaluation step, aiming to select a state-aggregated value function that is close to the true one in terms of mean squared error. It is worth noting that this is a somewhat idealized form of the algorithm. Practical algorithms use efficient sample-based approximations to the least-squares problem defining $\hat{Q}$. See Bertsekas and Tsitsiklis (1996) or Bertsekas (2011) for an introduction.

How does this algorithm perform? Our main result in this section is captured by the following proposition, giving a lower bound on performance that is worse than the result in Theorem 3 by a factor of the effective horizon $1/(1-\gamma)$. Recall from Section 3 that there is a broad equivalence between searching over the restricted class of value functions in $\mathcal{Q}_\phi$ and searching over the restricted class of policies Π_ϕ. Any advantage in the limiting performance of policy gradient methods is due to the way in which it searches over policies and not an advantage in representational power.

Theorem 7. *There exists an MDP, a state-aggregation ϕ, and initial policy π_1 such that, if $\{\pi_t\}_{t\in\mathbb{N}}$ is generated by Algorithm 1 with inputs given by ϕ, π_1, and uniform weighting $w(s) = 1/|\mathcal{S}|$ $\forall s$, then*

$$\liminf_{t\to\infty} J(\pi^*) - J(\pi_t) \geq \frac{\gamma\epsilon_\phi/4}{(1-\gamma)}. \tag{10}$$

Later in this section, we study a variant of API that adapts state-relevance weights across iterations.

Note that a classic result of Bertsekas and Tsitsiklis (1996, proposition 6.2), specialized to this setting, can be used to show a bound in the other direction that matches (10) up to a numerical constant. Technically, the analysis of Bertsekas and Tsitsiklis (1996) applies to value functions and not state-action value functions. The reader can find the same proof written in terms of state-action value functions in Agarwal et al. (2019a).

Theorem 7 is established through Example 1, which synthesizes an example from Bertsekas and Tsitsiklis (1996) with an example of Van Roy (2006). The latter work studies approximate value iteration in state-aggregated problems, establishing a result such as Theorem 7. The former work gives an example in which the optimality gap under API exhibits poor dependence on the horizon, but that example does not treat state aggregation. Surprisingly, although the state-aggregated problem in Example 1 is similar to the example of Bertsekas and Tsitsiklis (1996), the conclusion is stronger. The result of Bertsekas and Tsitsiklis (1996) is analogous to (10) but with the limit-inferior replaced with a limit-superior. In their example, API cycles endlessly between policies, sometimes selecting the optimal policy and sometimes selecting disastrous ones. In addition to applying in state-aggregated problems, Theorem 7 strengthens the conclusion by showing that API consistently selects disastrous policies in the limit.

Algorithm 1 (API)

input: $w \in \Delta(\mathcal{S})$, $\pi_1 \in \Pi$, ϕ
(1) **for** $t = 1, 2, \ldots,$ **do**
  `/* Approximate policy evaluation step */`
(2) $\hat{Q}_t \in \operatorname{argmin}_{\hat{Q}\in\mathcal{Q}_\phi} \|\hat{Q} - Q_{\pi_t}\|_{2,w\times 1}$;
  `/* Policy improvement step */`
(3) $\pi_{t+1}(s) \in \operatorname{argmax}_{a\in\mathcal{A}} \hat{Q}_t(s,a)$ $\forall s$;
(4) **end**.

Example 1. Consider an MDP with $n = 2m$ states, depicted in Figure 1 for $n = 10$ and $m = 5$. For $s \in \{1, \ldots, m\}$, we have $\phi(s) = \phi(s+m) = s$. This means that the algorithms don't distinguish between s and $s+m$. In state $s \in \{2, \ldots, m\}$, there are two possible actions: `move`, which moves the agent to state $s-1$ and generates a reward $r(s, \texttt{Move}) = 0$, and `stay`, which keeps the agent in the same state with reward $r(s, \texttt{Stay})$. Rewards obey the recursion

$$r(1, \texttt{Stay}) = 0 \qquad r(s, \texttt{Stay}) = \gamma r(s-1, \texttt{Stay}) - c$$
$$\text{for } s \in \{2, \ldots, n\},$$

and the formula $r(s, \texttt{Stay}) = -c\sum_{i=2}^{s} \gamma^{i-2}$. The negative reward for the action `stay` can be thought of as a cost. State 1 has only the costless action `stay`. (Or one can think of `move` as being identical to `stay` in state 1).

Transition probabilities from state $s+m$ are identical to those at state s, and $r(s+m, \texttt{Move}) = r(s, \texttt{Move}) = 0$, but $r(s+m, \texttt{Stay}) = r(s, \texttt{Stay}) + \epsilon_\phi$, where $\epsilon_\phi > 0$. Pick $c = \epsilon_\phi/2$. The optimal policy plays `move` from every state $s \in \{2, \ldots, m\}$.

Figure 2 displays simulation results. The dashed horizontal line represents $2\epsilon_\phi$, the upper bound on the limiting optimality gap proved in Theorem 3. In particular, any optimization method that is guaranteed to

reach a stationary point of $J(\cdot)$ has a limiting optimality gap below the dashed blue line. The simulation results show that a particular variant, projected gradient ascent with small constant step size, in fact, converges gracefully to optimality in this example. The performance of approximate policy iteration is far worse. In the second iteration, it actually reaches an optimal policy, but from there, performance continues to degrade. In the limit, it cycles endlessly between two policies. That cycling behavior is common with API and is confirmed analytically as follows.

Proof of Theorem 7. Consider API applied to Example 1 with uniform weighting, that is, $w(s) = 1/|\mathcal{S}|$, and an initial policy π_1 with $\pi_1(s) = \texttt{Stay}$ for $s \in \{2,4,6,\ldots, m-1\}$ and $\pi_1(s) = \texttt{Move}$ for $s \in \{3,5,7,\ldots,m\}$. For simplicity, we assume m is an odd number. The policy is state aggregated, so $\pi_1(s+m) = \pi_1(s)$ for $s \le m$. We show that the next policy produced by API, π_2, plays `move` at states $\{2,4,\ldots,m-1\}$ but plays `stay` at states $\{3,5,7,\ldots,m\}$. Proceeding in this manner, one finds that $\pi_3 = \pi_1$, $\pi_4 = \pi_2$, and the policies cycle endlessly.

Assuming for the moment that this result holds, let us consider the optimality gap under any initial distribution with $\rho(m) > 1/2$. We find $J(\pi^*) - J(\pi_t) > (1/2)(1-\gamma)(V^*(m) - V_{\pi_t}(m))$. Observe that an optimal policy π^*, which chooses `move` in every state, incurs cost $V_{\pi^*}(m) = 0$. On the other hand, $V_{\pi_t}(m) = \frac{r(m,\texttt{Stay})}{1-\gamma}$ for t even and $V_{\pi_t}(m) = 0 + \gamma \cdot \frac{r(m-1,\texttt{Stay})}{1-\gamma}$ for t odd. These formulas reflect that either policy moves at most once before staying perpetually at one of the rightmost states. We find

$$J(\pi^*) - J(\pi_t) > -\frac{\gamma}{2} \cdot r(m-1,\texttt{Stay}) = \frac{c\gamma}{2} \cdot \sum_{i=2}^{m-1} \gamma^{i-2} \overset{m\to\infty}{\to} \frac{c\gamma}{2(1-\gamma)} = \frac{\epsilon_\phi \gamma}{4(1-\gamma)}.$$

This establishes that Theorem 7 holds for problem instances with m sufficiently large.

We now turn to verifying that policies cycle in the manner described. The weighted least-squares problem solved by API has a particularly simple form in this case. It is straightforward to show that the problem defining $\hat{Q}_t$ decomposes across segments of the state space and, as the conditional mean minimizes squared loss, has the form

$$\begin{aligned}\hat{Q}_t(s,a) &= \mathbb{E}_{S\sim w}[Q_{\pi_t}(S,a) \mid S \in \phi^{-1}(s)] \\ &= \frac{Q_{\pi_t}(s,a) + Q_{\pi_t}(s+m,a)}{2} \qquad \forall s \le m, a \in \mathcal{A} \\ &= Q_{\pi_t}(s,a) + (\epsilon_\phi/2)\mathbf{1}(a = \texttt{Stay}) \qquad \forall s \le m, a \in \mathcal{A}.\end{aligned}$$

That is, in each segment, the value function of the current policy is overestimated by $\epsilon_\phi/2$ at state s and underestimated by $\epsilon_\phi/2$ in state $s+m$.

We verify that π_2 has the form conjectured. The proof for π_3 is uses the same ideas. Under π_1 and for $s \in \{4,6,8,\ldots\}$, we have $V_{\pi_1}(s) = r(s,\texttt{Stay})/(1-\gamma)$ and $V_{\pi_1}(s-1) = 0 + \gamma V_{\pi_1}(s-2) = \gamma r(s-2,\texttt{Stay})/(1-\gamma)$. Then,

$$\begin{aligned}Q_{\pi_1}(s,\texttt{Stay}) &= r(s,\texttt{Stay}) + \gamma V_{\pi_1}(s) = r(s,\texttt{Stay})/(1-\gamma) \\ Q_{\pi_1}(s,\texttt{Move}) &= r(s,\texttt{Move}) + \gamma V_{\pi_1}(s-1) \\ &= \gamma^2 r(s-2,\texttt{Stay})/(1-\gamma).\end{aligned}$$

Then, the least squares approximation gives $\hat{Q}_1(s,\texttt{Stay}) = r(s,\texttt{Stay})/(1-\gamma) + \epsilon_\phi/2$ and $\hat{Q}_1(s,\texttt{Move}) = \gamma^2 r(s-2,\texttt{Stay})/(1-\gamma)$. By plugging in $\epsilon_\phi = c/2$, one can verify that $\hat{Q}_1(s,\texttt{Move}) > \hat{Q}_2(s,\texttt{Stay})$, so π_2 plays `move` from states $s \in \{4,6,8,\ldots m\}$. The edge case of $s=2$ needs to be handled separately. One finds $\hat{Q}_1(2,\texttt{Move}) = 0$ and $\hat{Q}_1(2,\texttt{Stay}) = -c + \epsilon_\phi/2 = 0$. Breaking ties[5] in favor of the action `stay`, we get $\pi_2(s) = \texttt{Stay}$ for $s \in \{2,4,\ldots,m\}$.

Under π_1 and for $s \in \{3,5,7,\ldots\}$, we have $V_{\pi_1}(s-1) = r(s-1,\texttt{Stay})/(1-\gamma)$ and $V_{\pi_1}(s) = 0 + \gamma V_{\pi_1}(s-1) = \gamma r(s-1,\texttt{Stay})/(1-\gamma)$. Then,

$$\begin{aligned}Q_{\pi_1}(s,\texttt{Stay}) &= r(s,\texttt{Stay}) + \gamma V_{\pi_1}(s) = r(s,\texttt{Stay}) \\ &\quad + \gamma^2 r(s-1,\texttt{Stay})/(1-\gamma) \\ Q_{\pi_1}(s,\texttt{Move}) &= r(s,\texttt{Move}) + \gamma V_{\pi_1}(s-1) \\ &= \gamma r(s-1,\texttt{Stay})/(1-\gamma).\end{aligned}$$

Then, the least squares approximation gives $\hat{Q}_1(s,\texttt{Stay}) = Q_{\pi_1}(s,\texttt{Stay}) + \epsilon_\phi/2$ and $\hat{Q}_1(s,\texttt{Move}) = Q_{\pi_1}(s,\texttt{Move})$. Now, the misestimation error $\epsilon_\phi/2$ is enough cause the algorithm to select the decision `stay`. In particular,

$$\begin{aligned}\hat{Q}_1(s,\texttt{Stay}) - \hat{Q}_1(s,\texttt{Move}) &= \frac{\epsilon_\phi}{2} + r(s,\texttt{Stay}) - \gamma r(s-1,\texttt{Stay}) \\ &= \frac{\epsilon_\phi}{2} + (-c + \gamma r(s-1,\texttt{Stay})) - \gamma r(s-1,\texttt{Stay}) \\ &= \frac{\epsilon_\phi}{2} - c = 0.\end{aligned}$$

Breaking ties in favor of the actions `Stay`, we find that π_2 plays `stay` in states $\{3,5,7,\ldots\}$. □

5.1. Brittle Behavior of API with On-Policy State Relevance Weights

We illustrate that policy gradient sometimes dramatically outperforms a version of API that uses a fixed state weighting. Algorithm 2 presents another natural form of API in which these weights are adapted over time. At iteration t, it weighs states according to the occupancy measure η_{π_t}, prioritizing accuracy at states that are visited often. This choice arises organically if the data used to approximate Q_{π_t} is generated by applying π_t in the environment.

This modification to API seems to address Example 1, but it exhibits similarly brittle behavior in other examples. This possibility is validated through the numerical simulation of Example 2, depicted in Figure 4. An analytical proof is likely possible by following the argument in Theorem 7.

Algorithm 2 (API with Adaptive State Weighting)

input: $\pi_1 \in \Pi$, ϕ

(1) **for** $t = 1, 2, \ldots,$ **do**

/* policy evaluation */

(2) $\hat{Q}_t \in \operatorname{argmin}_{\hat{Q} \in \mathcal{Q}_\phi} \|\hat{Q} - Q_{\pi_t}\|_{2,\eta_{\pi_t} \times 1}$;

/* Policy improvement */

(3) $\pi_{t+1}(s) \in \operatorname{argmax}_{a \in \mathcal{A}} \hat{Q}_t(s,a) \ \forall s$;

(4) **end**.

Example 2. Consider an MDP with $n = 2m$ states, depicted in Figure 3 for $n = 10$ and $m = 5$. For $s \in \{1, \ldots, m\}$, we have $\phi(s) = \phi(s+m) = s$. This means that the algorithms don't distinguish between s and $s+m$. In a state with $\phi(s) \in \{2, \ldots, m\}$, there are two possible actions: `move`, which moves the agent to state $s-1$, and `stay`, which keeps the agent in the same state partition but brings an agent in state s to $s+m$ and one who is in state $s+m$ to state s. The action `move` generates zero reward. Rewards for the action `stay` at the states depicted on the top of Figure 3 obey the recursion

$$r(1+m, \texttt{Stay}) = 0$$
$$r(s+m, \texttt{Stay}) = \gamma r(s+m-1, \texttt{Stay}) - c \quad \text{for } s \in \{2, \ldots, m\},$$

and the formula $r(s+m, \texttt{Stay}) = -c \sum_{i=2}^{s} \gamma^{i-2}$. Playing the action `stay` generates a higher reward in the states depicted at the bottom of Figure 3 with $r(s, \texttt{Stay}) = r(s+m, \texttt{Stay}) + \epsilon_\phi$.

Example 2 involves two careful modifications to Example 1. First, in Example 2, selecting `stay` repeatedly causes the system to cycle between two states in a common partition. Second, the reward generated from playing `stay` is higher in a state $s \in \{2, \ldots, m\}$ (bottom row of Figure 3) than in the corresponding state $s+m$ (top row of Figure 3). These modifications are designed to ensure that Algorithm 2 cycles between bad policies. See Appendix C for further discussion.

6. What Drives the Performance Gap? The Importance of On-Policy State Weighting and Incremental Updates

Policy gradient methods are intimately related to API, making the performance difference between them all the more striking. To make their connections clear, we establish in Theorem 8 a precise equivalence between two algorithms: one is a Frank–Wolfe (Frank and Wolfe 1956, Jaggi 2013) variant of policy gradient, and the other is a form of API that uses online state-relevance weights and soft policy updates. After giving a short proof, we turn to discussion of the insights this equivalence yields.

Theorem 8. *Suppose Algorithms 3 and 4 are applied with the same inputs and the optimization problems in step 2 of Algorithm 3 and step 3 of Algorithm 4 always have unique solutions. Then, each algorithm produces an identical sequence of policies.*

Algorithm 3 (Frank–Wolfe Policy Gradient)

input: $\alpha, \pi_1 \in \Pi$, ϕ

(1) **for** $t = 1, 2, \ldots,$ **do**

/* Maximize linearization */

(2) $\tilde{\pi}_{t+1} = \operatorname{argmax}_{\pi \in \Pi_\phi} \langle \nabla J(\pi_t), \pi - \pi_t \rangle$;

/* Soft policy update */

(3) $\pi_{t+1} = \alpha \tilde{\pi}_{t+1} + (1-\alpha)\pi_t$;

(4) **end**.

Algorithm 4 (Soft API with Adaptive Weighting)

input: $\alpha, \pi_1 \in \Pi$, ϕ

(1) **for** $t = 1, 2, \ldots,$ **do**

/* Occupancy weighted policy evaluation */

(2) $\hat{Q}_t \in \operatorname{argmin}_{\hat{Q} \in \mathcal{Q}_\phi} \|\hat{Q} - Q_{\pi_t}\|_{2,\eta_{\pi_t} \times 1}$;

/* Policy improvement */

(3) $\tilde{\pi}_{t+1}(s) \in \operatorname{argmax}_{a \in \mathcal{A}} \hat{Q}_t(s,a) \ \forall s$;

/* Soft policy update */

(4) $\pi_{t+1} = \alpha \tilde{\pi}_{t+1} + (1-\alpha)\pi_t$;

(5) **end**.

Theorem 8 appears to be new, but related results appear several times in the literature. Algorithm 2 is often called the conservative policy iteration and is first proposed by Kakade and Langford (2002) based on considerations similar to policy gradient methods. In cases without approximation, Vieillard et al. (2019) and Bhandari and Russo (2021) observe that the Frank–Wolfe algorithm is equivalent to Algorithm 4. O'Donoghue et al. (2017) and Schulman et al. (2017a) study a related equivalence when entropy regularization is applied.

6.1. Proof of Theorem 8 Using the Actor–Critic Theorem

To compare policy gradient and API, we rely on the theory of actor–critic methods, which uses estimated value functions in evaluating gradients of $J(\cdot)$. To make this precise, recall the policy gradient expression in Lemma 5 expresses directional derivatives as a certain weighted inner product, $\langle \nabla J(\pi), \pi' - \pi \rangle = \langle Q_\pi, \pi' - \pi \rangle_{\eta_\pi \times 1}$. Actor–critic methods replace the true value function Q_π with some parametrized approximation, producing an approximate gradient.

An extremely elegant result of Konda and Tsitsiklis (2000) and Sutton et al. (2000) shows that compatible value function approximation produces no error in evaluating the gradient in feasible ascent directions. We identify the form of compatible function approximation in our setting. As in the previous section, $\|Q\|_{2,\eta_\pi \times 1}$ denotes the norm induced by the inner product $\langle \cdot, \cdot \rangle_{\eta_\pi \times 1}$ defined in (6).

Lemma 9 (Compatible Function Approximation). *If* $\hat{Q}_\pi = \operatorname{argmin}_{\hat{Q} \in \mathcal{Q}_\phi} \|\hat{Q} - Q_\pi\|_{2,\eta_\pi \times 1}$, *then,*

$$\langle \nabla J(\pi), \pi' - \pi \rangle = \langle \hat{Q}_\pi, \pi' - \pi \rangle_{\eta_\pi \times 1} \qquad \forall \pi' \in \Pi_\phi.$$

Proof. Observe that $\mathcal{Q}_\phi = \text{Span}(\Pi_\phi)$, where $\text{Span}(\Pi_\phi)$ consists of all vectors of the form $\sum_{i=1}^{I} c_i \pi^{(i)}$, where each $c_i \in \mathbb{R}$ is a scalar and $\pi^{(i)} \in \Pi_\phi$. Then, $\hat{Q}_\pi$ is the orthogonal projection of Q_π onto $\text{Span}(\Pi_\phi)$ with respect to the norm induced by the inner product $\langle \cdot\,,\,\cdot \rangle_{2,\eta_\pi \times 1}$. This means the error vector $Q_\pi - \hat{Q}_\pi$ is orthogonal to the subspace $\text{Span}(\Pi_\phi)$ with respect to $\langle \cdot\,,\,\cdot \rangle_{2,\eta_\pi \times 1}$, implying

$$\langle Q_\pi\,,\,\tilde{\pi} \rangle_{\eta_\pi \times 1} = \langle \hat{Q}_\pi\,,\,\tilde{\pi} \rangle_{\eta_\pi \times 1} \quad \forall \tilde{\pi} \in \text{Span}(\Pi_\phi).$$

Combined with Lemma 5, this yields the result. □

Theorem 8 is a simple corollary of Lemma 9. By Lemma 9, step 2 of Algorithm 3 is equivalent to

$$\begin{aligned}\tilde{\pi}_{t+1} &\in \operatorname*{argmax}_{\pi \in \Pi_\phi} \langle \nabla J(\pi_t)\,,\,\pi - \pi_t \rangle \\ &= \operatorname*{argmax}_{\pi \in \Pi_\phi} \langle \hat{Q}_t\,,\,\pi - \pi_t \rangle_{\eta_{\pi_t} \times 1},\end{aligned}$$

where $\hat{Q}_t$ has the same definition as in Algorithm 4.

6.2. Discussion

In light of Example 1, Theorem 8 reveals that two changes to API together have an enormous impact. One is to use on-policy state-relevance weights in the choice of loss function that is minimized to select an approximate value function. Lemma 9 shows that using this estimation loss orients estimation toward accurate evaluation of the decision objective, at least locally. A poor choice of state weighting seems to have contributed to the poor performance of Algorithm 1 in Example 1. The "top states" $(m+1, \ldots, 2m)$, displayed shaded in the top half of Figure 1, were given the same weight as the "bottom states." With on-policy state-relevance weights, little weight is given to the top states as they are visited infrequently. This allows for an accurate representation at the more important bottom states, dampening the propagating errors from state aggregation that are shown in the proof of Theorem 7. Van Roy (2006) previously observed that the robustness of policies derived from solutions of state-aggregated Bellman equations can depend critically on the choice of state-relevance weights. It is an open question whether his theory can be connected formally to the analysis in this paper.

The other change to API is to use soft, or local, changes to the policy. In the tth iteration of Algorithm 4, the state-relevance weights η_{π_t} capture relevance under the policy π_t by design. But if the step size is large, they may no longer reflect the relevance of states under the policy π_{t+1} over which the algorithm is optimizing. Equation (8) shows that the second order error term, $J(\pi_{t+1}) - J(\pi_t) - \langle \nabla J(\pi_t)\,,\,\pi_{t+1} - \pi_t \rangle$, depends on the magnitude of distribution shift, $\eta_{\pi_{t+1}} - \eta_{\pi_t}$. Example 2 shows that this issue can severely impact decision quality. In that example, the relevance of top states changes substantially across iterations, and this causes Algorithm 2 to cycle endlessly between policies with poor performance. Other works in the literature focus on ensuring that policy changes are small enough that performance improves strictly in each iteration (Kakade and Langford 2002, Schulman et al. 2015), but I am not aware of any results that show such severe degradation in performance is possible otherwise.

7. Conclusion

A surge of recent papers on the theory of reinforcement learning establish convergence rates and sample complexity bounds for different algorithms. Few, however, elucidate the subtle impact of algorithmic design choices on robustness to approximation errors. This paper provides one such case study, focused on the comparison between approximate policy iteration and policy gradient when applied with state-aggregated representations. The main contribution is providing a short, self-contained treatment with a transparent gap between provable upper and lower bounds.

One open question, highlighted in Section 4, is whether the notion of approximation error in Definition 2 can be relaxed to depend only on the optimal value function as shown for optimistic Q-learning in Dong et al. (2019). For simplicity and brevity, this paper focuses on the quality of stationary points and, at times, on the simple projected policy gradient method. Convergence rates for policy gradient methods are given, for example, in Agarwal et al. (2019b), Bhandari and Russo (2019), and Shani et al. (2020), and it seems that similar finite time bounds could be developed here.

Another open direction is to generalize these results beyond the case of state aggregation. The critical feature of state-aggregated representations is that they can be adjusted locally without impacting the approximation in other regions of the state space. By contrast, in general, linear models can be quite rigid with local changes influencing the approximation at distant states. This rigidity seems to drive a dependence of past guarantees for policy gradient methods on certain distribution shift terms that were avoided in Theorem 3, such as the κ_ρ term in Theorem 4. Indirectly, poor dependence on the problem's time horizon was avoided for the same reason.[6] The deep neural representations that are popular today seem to have elements of both state-aggregation and global linear approximation. A crisp understanding of local methods such as state aggregation may provide some useful intuition for the study of neural networks.

Acknowledgments

The author thanks Jalaj Bhandari for an earlier research collaboration that shaped the author's thinking around policy gradient methods.

Appendix A. Implementing Projected Policy Gradient with Aggregated State Approximations

Conceptually, the simplest policy gradient method is the projected gradient ascent iteration:

$$\pi^{t+1} = \mathrm{Proj}_{2,\Pi_\phi}\left(\pi^t + \alpha \nabla J(\pi^t)\right) = \operatorname*{argmax}_{\pi\in\Pi_\phi}\left(\pi^t + \langle \nabla J(\pi^t),\ \pi - \pi^t\rangle - \frac{1}{2\alpha}\|\pi - \pi^t\|_2^2\right) \quad t \in \mathbb{N},$$

where any policy $\pi \in \Pi$ is viewed as a stacked $|\mathcal{S}|\cdot|\mathcal{A}|$-dimensional vector satisfying $\sum_{a\in\mathcal{A}}\pi(s,a) = 1$ and $\pi \geq 0$. The operator $\mathrm{Proj}_{2,\Pi_\phi}(\pi) = \mathrm{argmin}_{\pi'\in\Pi_\phi}\|\pi' - \pi\|_2^2$ denotes orthogonal projection onto the convex set Π_ϕ with respect to the Euclidean norm. The second equality is a well-known "proximal" interpretation of the projected update (Beck 2017). Although the optimization problem

$$\operatorname*{argmax}_{\pi\in\Pi_\phi}\left(\pi^t + \langle \nabla J(\pi^t),\ \pi - \pi^t\rangle - \frac{1}{2\alpha}\|\pi - \pi^t\|_2^2\right)$$

appears to involve $|\mathcal{S}|\cdot|\mathcal{A}|$ decision variables, it is equivalent to one involving $m\cdot|\mathcal{A}|$ decision variables.

Algorithm A.1 uses $\theta \in \mathbb{R}^{m\times|\mathcal{A}|}$ to denote the parameter of a state-aggregated policy, in which $\pi_\infty(s,a) = \theta_{i,a}$ is the probability of selecting action a for a state $s \in \phi^{-1}(i)$ in segment i. The projection has a simple solution, involving projecting the vector $\tilde{\theta}_{s,:}$ corresponding to partition i onto the space of action distributions $\Delta(\mathcal{A})$. Projection onto the simplex can be executed with a simple soft thresholding procedure. In particular, the projection $\hat{y} \in \Delta(\mathcal{A})$ of a vector $y \in \mathbb{R}^{|\mathcal{A}|}$ satisfies $\hat{y}_i = \max\{y_i - \beta, 0\}$, where β is chosen so that $\sum_i \hat{y}_i = 1$. The pseudocode in Algorithm A.2, taken from Duchi et al. (2008), shows how β can be found efficiently. The algorithm runs in $O(|\mathcal{A}|\log(|\mathcal{A}|))$ time with the bottleneck being the sorting of the vector y. Duchi et al. (2008) shows how this can be reduced to an $O(|\mathcal{A}|)$ runtime.

Algorithm A.1 (Projected Policy Gradient)

input: $\theta \in \mathbb{R}^{m\times|\mathcal{A}|}$, step size α
for $t = 1, 2, \ldots,$ **do**
  Get gradient $g = \nabla_\theta J(\pi_\theta)$;
  Form target $\tilde{\theta} = \theta + \alpha\theta$;
  `/* Project onto simplex */`
  for $i = 1, \ldots, m$ **do**
    $\theta_{i,:} \leftarrow \min_{d\in\Delta(\mathcal{A})}\|d - \tilde{\theta}_{i,:}\|_2^2$
  end
end.

Algorithm A.2 (Projection onto the Probability Simplex)

input: Vector $y \in \mathbb{R}^k$
Sort y into μ with $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_k$;
Find $J = \max\left\{j \in [k] : \mu_j - \frac{1}{j}\left(\sum_{r=1}^j \mu_r - 1\right)\right\} > 0$;
Define $\beta = \frac{1}{J}\left(\sum_{i=1}^J \mu_i - 1\right)$;
output: $\hat{y} \in \mathbb{R}^k$, where $\hat{y}_i = \max\{y_i - \beta, 0\}$.

Algorithm A.3 provides an unbiased Monte Carlo policy gradient estimator. It is based on the formula

$$\frac{\partial J(\pi)}{\partial \pi(s,a)} = Q_\pi(s,a)\eta_\pi(s),$$

which can be derived from the standard policy gradient theorem by picking a direct policy parameterization (see, e.g., Agarwal et al. 2019b). Rewriting this, if $\tilde{s}_0 \sim \eta_\pi$ and $\tilde{a}_0|\tilde{s}_0 \sim \mathrm{Uniform}(1,\ldots,k)$, then

$$\frac{\partial J(\pi)}{\partial \pi(s,a)} = Q_\pi(s,a)\mathbb{P}(\tilde{s}_0 = s) = |\mathcal{A}|Q_\pi(s,a)\mathbb{P}(\tilde{s}_0 = s, \tilde{a}_0 = a).$$

Using the chain rule, we have

$$\frac{\partial J(\pi_\theta)}{\partial \theta_{i,a}} = \sum_{s\in\phi^{-1}(i)} \frac{\partial J(\pi)}{\partial \pi(s,a)} = |\mathcal{A}|\mathbb{E}[Q_\pi(\tilde{s}_0,a)1(\tilde{s}_0 \in \phi^{-1}(i), \tilde{a}_0 = a)].$$

Algorithm A.3 shows how this formula can be used, together with a simulator of the environment, to generate stochastic gradient $\hat{g}$ with $\mathbb{E}[\hat{g}] = \nabla_\theta J(\pi_\theta)$. This can be used in stochastic gradient schemes or, by averaging across many independent simulation runs, used to estimate $\nabla_\theta J(\pi_\theta)$ accurately. The algorithm begins by drawing a state $\tilde{s}_0$ from $\eta_\pi(\cdot)$ and then an action $\tilde{a}_0$ uniformly at random (see Remark A.1). Then, it uses a Monte Carlo rollout to estimate $Q_\pi(\tilde{s}_0, \tilde{a}_0)$. To give unbiased estimates of infinite horizon discounted sums underlying η_π and Q_π, it leverages a well-known equivalence between geometric discounting and the use of a random geometric horizon. For any scalar random variables $\{X_t\}_{t=0,1,\ldots}$, one has

$$\mathbb{E}\left[\sum_{t=0}^\infty \gamma^t X_t\right] = \mathbb{E}\left[\sum_{t=0}^\tau X_t\right],$$

where $\tau \sim \mathrm{Geometric}(1-\gamma)$ has distribution that is independent of $\{X_t\}$. The equivalence is because $\mathbb{P}(\tau \geq t) = \gamma^t$.

Algorithm A.3 (Simple Unbiased Gradient)

input: H, S, A, tuning parameters $\{\beta_k\}_{k\in\mathbb{N}}$
`/* Sample` $\tilde{s}_0 \sim \eta_\pi$ `*/`
(1) Sample $\tau \sim \mathrm{Geometric}(1-\gamma)$;
(2) Sample initial $s_0 \sim \rho$;
(3) Apply policy π for τ time steps;
(4) Observe $(s_0, a_0, r_0, \ldots, a_{\tau-1}, r_{\tau-1}, s_\tau)$;
(5) Set $\tilde{s}_0 = s_\tau$;
`/* Draw uniform random action */`
(6) Sample $\tilde{a}_0 \sim \mathrm{Uniform}\{1,\ldots,|\mathcal{A}|\}$;
`/* Unbiased estimate of` $Q_\pi(\tilde{a}_0, \tilde{s}_0)$ `*/`
(7) Sample $\tilde{\tau} \sim \mathrm{Geometric}(1-\gamma)$;
(8) Apply action $\tilde{a}_0$ and observe $(\tilde{r}_0, \tilde{s}_1)$;
(9) **if** $\tilde{\tau} > 0$ **then**
(10) Apply policy π for $\tilde{\tau}$ periods from $\tilde{s}_1$;
(11) Observe: $(\tilde{s}_1, \tilde{a}_1, \tilde{s}_2, \ldots, \tilde{a}_{\tau-1}, \tilde{r}_{\tau-1}, \tilde{s}_\tau)$;
(12) **end**
(13) Set $\hat{Q} = \tilde{r}_0 + \cdots + \tilde{r}_\tau$;
(14) Find state segment $I = \phi^{-1}(\tilde{s}_0)$;
(15) Set $\hat{g}(i,a) = \begin{cases} |\mathcal{A}|\cdot\hat{Q} & \text{if } i = I, a = \tilde{a}_0 \\ 0 & \text{otherwise} \end{cases}$;
output: $\hat{g} \in \mathbb{R}^{m\times|\mathcal{A}|}$.

Remark A.1. A more common presentation of unbiased policy gradient estimation uses a kind of inverse propensity estimate in which $\tilde{a}_0$ is sampled from the policy being evaluated (Williams 1992). This can have very large variance if the policy is nearly deterministic. The form presented ensures the variance of the sampled gradient is uniformly bounded.

Appendix B. Some Details on Remark 1

Equation (3) can be established as follows. If $Q \in \mathcal{Q}_\phi$ is state aggregated, then it is immediate that π defined by $\pi(s,a) = e^{Q(s,a)}/\sum_{a' \in \mathcal{A}} e^{Q(s,a')}$ is state aggregated. This shows that soft-maximization with respect to some state-aggregated value function yields a state-aggregated policy that assigns nonzero probability to each action. Now, we need to show that every strictly stochastic state-aggregated policy can be generated this way. Consider any policy $\pi \in \Pi_\phi$ with $\pi(s,a) > 0$ for all $a \in \mathcal{A}$. Picking $Q(s,a) = \log\pi(s,a)$ yields $\pi(s,a) = e^{Q(s,a)}/\sum_{a' \in \mathcal{A}} e^{Q(s,a')}$.

Results for policies that assign zero probability to some action are attained by taking limits of strictly stochastic policies that approach them. The minimum in (4) can be replaced with any deterministic tie-breaking mechanism. This is needed because, if ties are broken differently at states sharing a common segment, the induced policy would not be constant across segments.

Appendix C. Further Discussion of Example 2

As in the proof of Theorem 7, imagine Algorithm 2 is applied in Example 2 with initial policy π_1 that selects $\pi_1(s) = \texttt{Stay}$ for $s \in \{2,4,6,\ldots,m-1\}$ and $\pi_1(s) = \texttt{Move}$ for $s \in \{3,5,7,\ldots,m\}$. One can show, and this is validated numerically with open-source code, that the next policy produced by API, π_2, plays `move` at states $\{2,4,\ldots,m-1\}$ and plays `stay` at states $\{3,5,7,\ldots,m\}$. This cycle continues with $\pi_3 = \pi_1$, $\pi_4 = \pi_2$ and so on. The performance of these policies is depicted in Figure 4.

What drives this? Consider the policy π_1 described and some $s \in \{3,5,7,\ldots\}$ so $\pi_1(s) = \texttt{Move}$. Policy iteration considers the value of deviating from the prescribed action of π_1 for only a single period. So the agent is essentially comparing (A) picking `move` and then continually selecting `stay` at states $\{s-1, s-1+m\}$ and (B) picking `stay` and transitioning to state $s+m$ and then moving to $s-1$ and selecting `stay` thereafter. Because the initial distribution is much more likely to place the agent at s (top of Figure 3) than at $s+m$ (bottom of Figure 3) and the agent plays `move`; API with on-policy state-weighting essentially ignores behavior at $s+m$ when fitting an approximate Q-function. The problem is constructed so that there is a higher reward to playing `stay` at state s than at $s+m$, and this is enough to cause the agent to estimate that (B) is preferable to (A). The reverse of this logic plays out for states $s \in \{2,4,6,\ldots\}$ with $\pi_1(s) = \texttt{Stay}$.

Endnotes

[1] See Gabillon et al. (2013) for a full account of the history.

[2] See Remark 4 for discussion of updated results in Agarwal et al. (2019b).

[3] Here, we have broken ties deterministically in favor of the actions with a smaller index. If there are multiple optimal actions and ties are broken differently at states sharing a common segment, the induced policy would not be constant across segments.

[4] In Arxiv version 2, this is lemma E.3. To translate their result to our formulation, one must multiply the statement in lemma E.3 by $(1-\gamma)$ as in the definition $J(\pi) = (1-\gamma)\rho V_\pi$. They also have normalized so that $|r(s,a)| \le 1$. That is the reason $\|r\|_\infty$ does not appear in their expression.

[5] This example can be easily modified by taking c to be infinitesimally smaller than $\epsilon_\phi/2$, in which case no tie-breaking mechanism is needed.

[6] Past work, such as Bhandari and Russo (2019) and Agarwal et al. (2019b) uses the initial distribution ρ to control likelihood ratio terms as $\eta_{\pi^*}(s)/\eta_\pi(s) \ge (1-\gamma)(\eta_{\pi^*}(s)/\rho(s))$. Avoiding a dependence of limiting approximation error on distribution shift terms avoided an extra dependence on the effective time horizon, $(1-\gamma)^{-1}$.

References

Abel D, Hershkowitz DE, Littman ML (2016) Near optimal behavior via approximate state abstraction. *Internat. Conf. Machine Learn.* (JMLR), 2915–2923.

Agarwal A, Jiang N, Kakade SM (2019a) Reinforcement learning: Theory and algorithms. Technical report. Accessed April 21, 2023, https://rltheorybook.github.io.

Agarwal A, Henaff M, Kakade S, Sun W (2020) PC-PG: Policy cover directed exploration for provable policy gradient learning. *Adv. Neural Inform. Processing Systems* 33:13399–13412.

Agarwal A, Kakade SM, Lee JD, Mahajan G (2019b) Optimality and approximation with policy gradient methods in Markov decision processes. Preprint, submitted August 1, https://arxiv.org/abs/1908.00261.

Antos A, Szepesvári C, Munos R (2008) Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learn.* 71(1):89–129.

Bean JC, Birge JR, Smith RL (1987) Aggregation in dynamic programming. *Oper. Res.* 35(2):215–220.

Beck A (2017) *First-Order Methods in Optimization* (SIAM-Society for Industrial and Applied Mathematics, Philadelphia).

Bertsekas D, Castanon D (1989) Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Trans. Automatic Control* 34(6):589–598.

Bertsekas DP (1995) *Dynamic Programming and Optimal Control*, vol. 2 (Athena Scientific, Belmont, MA).

Bertsekas DP (1997) Nonlinear programming. *J. Oper. Res. Soc.* 48(3):334.

Bertsekas DP (2011) Approximate policy iteration: A survey and some new methods. *J. Control Theory Appl.* 9(3):310–335.

Bertsekas DP, Ioffe S (1996) Temporal differences-based policy iteration and applications in neuro-dynamic programming. Laboratory for Information and Decision Systems Report LIDS-P-2349, MIT, Cambridge, MA.

Bertsekas DP, Tsitsiklis JN (1996) *Neuro-Dynamic Programming* (Athena Scientific, Belmont, MA).

Bhandari J, Russo D (2019) Global optimality guarantees for policy gradient methods. Preprint, submitted June 5, https://arxiv.org/abs/1906.01786.

Bhandari J, Russo D (2021) On the linear convergence of policy gradient methods for finite MDPs. *Internat. Conf. Artificial Intelligence Statist.* (PMLR), 2386–2394.

Cen S, Cheng C, Chen Y, Wei Y, Chi Y (2021) Fast global convergence of natural policy gradient methods with entropy regularization. *Oper. Res.* 70(4):2563–2578.

Davis D, Grimmer B (2019) Proximally guided stochastic subgradient method for nonsmooth, nonconvex problems. *SIAM J. Optim.* 29(3):1908–1930.

Dean T, Givan R (1997) Model minimization in Markov decision processes. *Proc. AAAI Conf. Artificial Intelligence* (AAAI, Washington, DC), 106–111.

Defazio A, Bach F, Lacoste-Julien S (2014) SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. *Adv. Neural Inform. Processing Systems* 27:1646–1654.

Dong S, Van Roy B, Zhou Z (2019) Provably efficient reinforcement learning with aggregated states. Preprint, submitted December 13, https://arxiv.org/abs/1912.06366.

Duan Y, Ke T, Wang M (2019) State aggregation learning from Markov transition data. *Adv. Neural Inform. Processing Systems* 33:4486–4495.

Duchi J, Shalev-Shwartz S, Singer Y, Chandra T (2008) Efficient projections onto the ℓ_1-ball for learning in high dimensions. *Internat. Conf. Machine Learn.* (JMLR), 272–279.

Frank M, Wolfe P (1956) An algorithm for quadratic programming. *Naval Res. Logist. Quart.* 3(1–2):95–110.

Furmston T, Barber D (2012) A unifying perspective of parametric policy search methods for Markov decision processes. *Adv. Neural Inform. Processing Systems* 25:2717–2725.

Gabillon V, Ghavamzadeh M, Scherrer B (2013) Approximate dynamic programming finally performs well in the game of Tetris. *Adv. Neural Inform. Processing Systems* 26:1754–1762.

Ghadimi S, Lan G (2013) Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM J. Optim.* 23(4): 2341–2368.

Ghadimi S, Lan G (2016) Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Math. Programming* 156(1–2):59–99.

Gordon GJ (1995) Stable function approximation in dynamic programming. *Machine Learn. Proc.* (Morgan Kaufmann, San Francisco), 261–268.

Jaggi M (2013) Revisiting Frank-Wolfe: Projection-free sparse convex optimization. *Internat. Conf. Machine Learn.* (PMLR), 427–435.

Jiang N, Kulesza A, Singh S (2015) Abstraction selection in model-based reinforcement learning. *Internat. Conf. Machine Learn.* (PMLR), 179–188.

Kakade S, Langford J (2002) Approximately optimal approximate reinforcement learning. *ICML*, vol. 2, 267–274.

Kakade SM (2002) A natural policy gradient. *Adv. Neural Inform. Processing Systems* 14:1531–1538.

Khodadadian S, Jhunjhunwala PR, Varma SM, Maguluri ST (2021) On the linear convergence of natural policy gradient algorithm. *60th IEEE Conf. Decision Control* (IEEE, Piscataway, NJ), 3794–3799.

Konda VR, Tsitsiklis JN (2000) Actor-critic algorithms. *Adv. Neural Inform. Processing Systems* 1008–1014.

Lazaric A, Ghavamzadeh M, Munos R (2012) Finite-sample analysis of least-squares policy iteration. *J. Machine Learn. Res.* 13:3041–3074.

Li L, Walsh TJ, Littman ML (2006) Toward a unified theory of state abstraction for MDPs. *Proc. Ninth Intern. Sympos. Artificial Intelligence and Math.* (ISAIM), 1–10.

Maxwell MS, Henderson SG, Topaloglu H (2013) Tuning approximate dynamic programming policies for ambulance redeployment via direct search. *Stochastic Systems* 3(2):322–361.

Mei J, Xiao C, Szepesvari C, Schuurmans D (2020) On the global convergence rates of softmax policy gradient methods. *Internat. Conf. Machine Learn.* (PMLR), 6820–6829.

Misra D, Henaff M, Krishnamurthy A, Langford J (2020) Kinematic state abstraction and provably efficient rich-observation reinforcement learning. *Internat. Conf. Machine Learn.* (PMLR), 6961–6971.

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, et al. (2015) Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Munos R (2003) Error bounds for approximate policy iteration. *ICML*, vol. 3, 560–567.

O'Donoghue B, Munos R, Kavukcuoglu K, Mnih V (2017) Combining policy gradient and q-learning. *Fifth Internat. Conf. Learn. Representations* (Appleton, WI), 1–13.

Puterman ML (2014) *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (John Wiley & Sons, New York).

Reddi SJ, Sra S, Póczos B, Smola A (2016a) Stochastic Frank-Wolfe methods for nonconvex optimization. *54th Annual Allerton Conf. Comm. Control. Comput.* (IEEE, Piscataway, NJ), 1244–1251.

Reddi SJ, Sra S, Poczos B, Smola AJ (2016b) Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. *Adv. Neural Inform. Processing Systems* 29:1145–1153.

Reddi SJ, Hefny A, Sra S, Póczos B, Smola A (2016c) Stochastic variance reduction for nonconvex optimization. *Internat. Conf. Machine Learn.* (JMLR), 314–323.

Rust J (1996) Numerical dynamic programming in economics. Amman H, Kendrik D, Rust J, eds. *Handbook of Computational Economics*, vol. 1 (Amsterdam), 619–729.

Rust J (1997) Using randomization to break the curse of dimensionality. *Econometrica* 65(3):487–516.

Scherrer B (2014) Approximate policy iteration schemes: A comparison. *Internat. Conf. Machine Learn.* (JMLR), 1314–1322.

Scherrer B, Geist M (2014) Local policy search in a convex space and conservative policy iteration as boosted policy search. *Joint Eur. Conf. Machine Learn. Knowledge Discovery Databases* (Springer Berlin, Heidelberg), 35–50.

Scherrer B, Lesner B (2012) On the use of non-stationary policies for stationary infinite-horizon Markov decision processes. *Adv. Neural Inform. Processing Systems* 25:1826–1834.

Schulman J, Chen X, Abbeel P (2017a) Equivalence between policy gradients and soft q-learning. Preprint, submitted April 21, https://arxiv.org/abs/1704.06440.

Schulman J, Levine S, Abbeel P, Jordan M, Moritz P (2015) Trust region policy optimization. *Internat. Conf. Machine Learn.* (JMLR), 1889–1897.

Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017b) Proximal policy optimization algorithms. Preprint, submitted July 20, https://arxiv.org/abs/1707.06347.

Scott WR, Powell WB, Moazehi S (2014) Least squares policy iteration with instrumental variables vs. direct policy search: Comparison against optimal benchmarks using energy storage. Preprint, submitted January 4, https://arxiv.org/abs/1401.0843.

Shani L, Efroni Y, Mannor S (2020) Adaptive trust region policy optimization: Global convergence and faster rates for regularized MDPs. *Proc. Conf. AAAI Artificial Intelligence*, vol. 34, 5668–5675.

Singh SP, Jaakkola T, Jordan MI (1995) Reinforcement learning with soft state aggregation. *Adv. Neural Inform. Processing Systems* 7:361–368.

Sutton RS, Barto AG (2018) *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA).

Sutton RS, McAllester DA, Singh SP, Mansour Y (2000) Policy gradient methods for reinforcement learning with function approximation. *Adv. Neural Inform. Processing Systems* 12:1057–1063.

Szita I, Lörincz A (2006) Learning Tetris using the noisy cross-entropy method. *Neural Comput.* 18(12):2936–2941.

Tsitsiklis JN, Van Roy B (1996) Feature-based methods for large scale dynamic programming. *Machine Learn.* 22(1–3):59–94.

Van Roy B (2006) Performance loss bounds for approximate value iteration with state aggregation. *Math. Oper. Res.* 31(2):234–244.

Vieillard N, Pietquin O, Geist M (2019) On connections between constrained optimization and reinforcement learning. Preprint, submitted October 18, https://arxiv.org/abs/1910.08476.

Whitt W (1978) Approximations of dynamic programs, I. *Math. Oper. Res.* 3(3):231–243.

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

Xiao L, Zhang T (2014) A proximal stochastic gradient method with progressive variance reduction. *SIAM J. Optim.* 24(4):2057–2075.

Zhang J, Kim J, O'Donoghue B, Boyd S (2021) Sample efficient reinforcement learning with reinforce. *Proc. Conf. AAAI Artificial Intelligence*, vol. 35, 10887–10895.