Policy Gradient Methods in Reinforcement Learning

Nicholas Corrado
Department of Computer Science
University of Wisconsin-Madison
ncorrado@wisc.edu

Abstract

Policy gradient methods are widely used for reinforcement learning problems in practice because they (1) directly optimize the quantity of interest, (2) easily handle both continuous and discrete state and action spaces, and (3) can be applied to any differentiable policy parameterization. However, we have a relatively coarse theoretical understanding of these methods, particularly when it comes to convergence [1]. Juxtaposing this with their empirical success makes them a particularly interesting area of study. In this survey, we describe classical results for vanilla policy gradient methods and then trace the development of actor-critic methods, natural policy gradient methods, and trust region methods. We end with a discussion on global convergence.

1 Reinforcement Learning Setting

We consider the standard reinforcement learning setting described in [2]. Let $M = (\mathcal{S}, \mathcal{A}, P, r, \gamma, p)$ be a Markov decision process with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ where $P(s' \mid s, a)$ describes the probability of transitioning to state s' after taking action a in state s, reward function $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ where $r(s, a)$ is the reward immediately received after taking action a in state s, and discounting factor $\gamma \in [0, 1]$. Let $\rho : \mathcal{S} \to [0, 1]$ be the starting state distribution over $\mathcal{S}$. Let $\pi : \mathcal{S} \to \mathcal{A}$ denote a policy where $\pi(a \mid s)$ is the probability of taking action a while in state s. We will generally be working with stochastic policies, so we write $A_t \sim \pi(\cdot \mid S_t)$, where we use capital letters to represent quantities sampled from the policy. We define the value function $V^\pi(s)$, the state-action value function $Q^\pi(s, a)$, and the advantage function $A^\pi(s, a)$ as follows:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid \pi, s_0 = s \right] \tag{1}$$

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid \pi, s_0 = s, a_0 = a \right] \tag{2}$$

$$A^\pi(s, a) = V^\pi(s) - Q^\pi(s, a) \tag{3}$$

The subscript π denotes that the expectation is taken assuming the agent follows policy π at all time steps. We additionally define $d^\pi(s)$, the probability of being in state s under policy π starting at state s_0.

$$d^\pi(s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s | s_0, \pi) \tag{4}$$

We overload notation and let $d^\pi_\mu(s)$ denote the probability of being in state s under policy π starting at a state s_0 sampled from a starting start distribution μ.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

2 The Policy Gradient Framework

In contrast to value-based methods, policy-based model-free methods such as policy gradient methods seek to learn the optimal policy directly by considering a class of (stochastic) policies $\{\pi_\theta\}$ parametrized with respect to a weight vector θ. Policy gradient methods improve policy performance by updating θ according to the gradient of an objective function $J(\theta)$ that maps policies to a scalar performance metric:

$$J(\theta) = \sum_s d^\pi(s) \sum_a Q^\pi(s,a)\pi(a|s,\theta) \tag{5}$$

$$\theta_{t+1} = \theta_t + \alpha\widehat{\nabla J(\theta_t)} \tag{6}$$

Hats denote estimated quantities. All gradients are assumed to be taken with respect to θ unless otherwise specified. To simplify notation, we exclude the subscript θ from the policy when using Eq. 1 through Eq. 4.

The Policy Gradient Theorem [3, 4] expresses the gradient without $\frac{\partial d^\pi(s)}{\partial \theta}$, the effect of policy changes on the distribution of states, and serves as the backbone for policy gradient methods:

Theorem 1 (Policy Gradient Theorem)

$$\nabla_\theta J(\theta) \propto \sum_s d^\pi(s) \sum_a Q^\pi(s,a)\nabla\pi(a \mid s,\theta) \tag{7}$$

In the episodic case, the constant of proportionality is the average length of an episode, and in the continuing case it is 1, so that the proportionality is actually an equality. Note that proportionality suffices in the episodic case, as any constant out in front can be subsumed into the step size parameter α. We will be working with the continuing case unless stated otherwise.

3 Vanilla Policy Gradient (REINFORCE)

REINFORCE [5] is a Monte-Carlo method that samples trajectories to estimate the return G_t and then updates the policy parameter θ using stochastic gradient ascent. From Theorem 1, we can derive a way to estimate the gradient via sampling. Observe that the sum over states in the policy gradient theorem is weighted by the probabilities $d^\pi(s)$, allowing us to replace the sum over s with an expectation over π, sampling S_t from π. We can similarly convert the sum over actions into an expectation if we introduce $\pi(a|S_t,\theta)$:

$$\begin{aligned}
\nabla_J(\theta) &= \sum_s d^\pi(s) \sum_a Q^\pi(s,a)\nabla\pi(a \mid s,\theta) \\
&= \mathbb{E}_\pi\left[\sum_a Q^\pi(S_t,a)\nabla\pi_\theta(a \mid S_t,\theta)\right] \\
&= \mathbb{E}_\pi\left[\sum_a \pi(a \mid S_t,\theta)Q^\pi(S_t,a)\frac{\nabla\pi(a \mid S_t,\theta)}{\pi(a \mid S_t,\theta)}\right] \\
&= \mathbb{E}_\pi\left[Q^\pi(S_t,A_t)\frac{\nabla\pi(A_t \mid S_t,\theta)}{\pi(A_t \mid S_t,\theta)}\right] \qquad \text{where } A_t \sim \pi(\cdot|S_t,\theta) \\
&= \mathbb{E}_\pi\left[G_t\nabla_\theta \log\pi(A_t \mid S_t,\theta)\right] \qquad \text{since } Q^\pi(S_t,A_t) = \mathbb{E}_\pi\left[G_t \mid S_t,A_t\right]
\end{aligned} \tag{8}$$

The argument of the expectation is an unbiased estimate of the gradient that can be sampled at each time step. We can now use stochastic gradient ascent to update θ:

$$\theta_{t+1} = \theta_t + \alpha G_t \nabla \log\pi(A_t \mid S_t,\theta_t) \tag{9}$$

Note that because we use G_t, the REINFORCE is well-defined only for the episodic case, as we must simulate full trajectories before performing an update. The reward can be thought of as a weighting that increases the derivative in the direction of greater rewards.

The gradient estimate in Eq. 8, though unbiased, may suffer from high variance. To reduce variance while keeping the estimate unbiased, we can subtract a baseline value $b(S_t)$ from the return G_t, typically an estimate of the value function $\widehat{V}^{\pi}(S_t, w)$ parameterize by a weight vector w. Note that because the baseline does not depend on the action, the policy gradient theorem still holds, *i.e.* the expectation remains unchanged. This leads to a similar update rule:

$$\begin{aligned} \theta_{t+1} &= \theta_t + \alpha_\theta (G_t - \widehat{V}^{\pi}(S_t)) \nabla \log \pi(A_t \mid S_t, \theta_t) \\ w_{t+1} &= w_t + \alpha_w (G_t - \widehat{V}^{\pi}(S_t, w)) \nabla \widehat{V}^{\pi}(S_t, w) \end{aligned} \tag{10}$$

Theorem 1 in [5] says that, in expectation, the updates in Eq 9 and 10 will indeed produce a direction, *i.e.* the average update vector in weight space lies in a direction for which $J(\theta)$ is increasing. However, [5] provides no theoretical characterization of the asymptotic behavior of REINFORCE.

4 Actor-Critic Methods

Note that REINFORCE estimates the value function baseline after a full episode has been sampled, using only the value of only the first state of each state transition. Because of the algorithm's episodic nature, the estimate is made prior to the transition's action and thus does not provide us with an evaluation on how good our actions are. An algorithm that performs estimations and updates in an online fashion would provide much better information.

In actor–critic methods [4, 6], we use the state-value function at future states to get a better estimate of the return. The value function thus behaves as a "critic" that assesses actions made by the policy and informs the "actor" how it should update the policy parameters. Updates are performed at each step instead of waiting until the end of an episode, making the baseline a better assessment of the action taken. REINFORCE [5] can be seen as an actor-only method [7].

The update we present here is based on [8] and described in Ch. 13 of [2]. Consider using the next state S_{t+1} to compute the one-step return $G_{t:t+1} = R_{t+1} + \gamma \widehat{V}^{\pi}(S_{t+1}, w)$. An actor-critic update would be

$$\begin{aligned} \theta_{t+1} &= \theta_t + \alpha \left(G_{t:t+1} - \widehat{V}^{\pi}(S_t, w) \right) \nabla \log \pi(A_t \mid S_t, \theta_t) \\ &= \theta_t + \alpha \left(R_{t+1} + \gamma \widehat{V}^{\pi}(S_{t+1}, w) - \widehat{V}^{\pi}(S_t, w) \right) \nabla \log \pi(A_t \mid S_t, \theta_t) \end{aligned} \tag{11}$$

However, the critic introduces bias and updates can fail to converge under general conditions. [6] provides a theorem similar to the one stated in [4] showing local convergence for actor-critic methods under similar conditions. The approach is different than that of REINFORCE, as [4] uses function approximation. Theorem 2 in [4] shows that for a *compatible* approximation $f_w(s, a)$ of $Q^{\pi}(s, a)$, we can rewrite the gradient of our objective as:

$$\nabla J(\theta) = \sum_s d^{\pi}(s) \sum_a f_w(s, a) \nabla \pi(a \mid s, \theta)$$

Under certain conditions stated in Theorem 3 of [4], running stochastic gradient ascent on θ and w with the above gradient and using $f_w(s, a)$ as a baseline yields local convergence.

4.1 Asynchronous Advantage Actor-Critic

Asynchronous Advantage Actor-Critic (A3C) [9] further extends the actor-critic approach by using multiple agents (workers) who independently interact with different copies of the environment in parallel, locally updating their own weights and asynchronously syncing with shared global parameters. Workers can thus explore a bigger part of the state-action space in much less time. The updates are happening asynchronously in HogWild! [10] fashion. After each update, the agents reset their parameters to match the global parameters. In this way, workers indirectly share information. It is possible to implement synchronous counterpart of A3C called often referred to as Advantage Actor-Critic (A2C) in literature [2, 11], where global parameters are updated only after all parallel workers have finished computation.

4.2 Off-Policy Actor-Critic

It is often desirable to evaluate a new policy before actually deploying it using historical data collected from a different policy. Such an approach would allow RL researchers to evaluate algorithm performance using the same data. Furthermore, trajectory sampling is typically cheap and easy to do in simulated environments, but in real-world environments (often experienced in robotics), sampling is expensive [12]. More importantly, a new policy could be dangerous to deploy if it performs worse than the currently deployed policy [13].

The primary issue is that we are unable to draw trajectories from the target policy distribution, as we only have access to data drawn from the distribution of some offline behavior policy [14]. Off-policy evaluation seek to address this problem using importance sampling [15, 2], a classical technique for handling just this mismatch by giving more weight to rare samples that occur under the offline behavior distribution but occur more frequently the target policy.

The off-policy approach does not require full trajectories and can reuse any past episodes for much better sample efficiency [16]. Since the offline data is sampled from a behavior policy that is different from the target policy, we also achieve better exploration.

Let $q(a \mid s)$ denote the behavior policy used to sample trajectories. Then our objective function sums up rewards over the state distribution defined by q instead of π:

$$\begin{aligned} J(\theta) &= \sum_s d^q(s) \sum_a Q^\pi(s,a)\pi(a \mid s,\theta) \\ &= \mathbb{E}_q \left[\sum_a Q^\pi(S_t,a)\pi(a \mid S_t,\theta) \right] \end{aligned} \tag{12}$$

This leads to a new formulation of the gradient:

$$\begin{aligned} \nabla_J(\theta) &= \mathbb{E}_q \left[\sum_a \left(Q^\pi(S_t,a)\nabla\pi(a \mid S_t,\theta) + \nabla Q^\pi(S_t,a)\pi(a \mid S_t,\theta) \right) \right] \\ &\approx \mathbb{E}_q \left[\sum_a Q^\pi(S_t,a)\nabla\pi(a \mid S_t,\theta) \right] \\ &= \mathbb{E}_q \left[\frac{\pi(a \mid s,\theta)}{q(a \mid s)} Q^\pi(S_t,A_t)\nabla \log \pi(A_t \mid S_t,\theta) \right] \end{aligned} \tag{13}$$

In the second line of Eq. 13 , we ignore the second term inside the sum, since $\nabla Q^\pi(S_t, A_t)$ is hard to compute. Call this approximation $g(\theta)$. Theorems 1 and 2 in [8] justify this approximation. Theorem 1 assures that there exists a small enough step size in the direction of $g(\theta)$ such that the objective increases. Theorem 2 says that the set of all stationary points with respect to $g(\theta)$ are also stationary points of the true gradient, so long as the value function can be represented by a linear approximation of $V^\pi(s)$. Thus, we can still use policy gradient methods in the off-policy setting by simply adjusting the policy probabilities using an importance sampling estimator $\frac{\pi(a|s,\theta)}{q(a|s)}$ to compensate for the difference in behavior policy and true policy.

5 Natural Policy Gradient

Vanilla policy gradient methods are generally slow to converge due to the vanishing gradient problem: when the objective function is nearly flat, the gradient is small, and thus policy parameter updates are also small. The slow convergence additionally results in high sample complexity. See chapter 10 in [17] for an example MDP that experiences this problem.

These issues arise due to the fact that Eq. 6 implicitly uses the Euclidean metric to measure the distance between policies. However, this is generally not a good metric. Consider updating $\mathcal{N}(0,1)$ to $\mathcal{N}(1,1)$ and updating $\mathcal{N}(0,0.01)$ to $\mathcal{N}(1,0.01)$ In the Euclidean sense, both pairs of distributions are 1 unit apart in parameter space. However, $\mathcal{N}(0,0.01)$ and $\mathcal{N}(1,0.01)$ are clearly much more disparate than $\mathcal{N}(0,1)$ and $\mathcal{N}(1,1)$. When using an inappropriate metric, single updates to policy parameters can change the policy drastically and have a negative impact on convergence even though

the gradient update rule only makes a small change in the parameter space. We need to instead define a metric based on the manifold that the policy coordinates parameterize rather than on the coordinates themselves [18]. The KL divergence is a much more reasonable distance measure, as it allows us to better compare changes in policy parameters to changes in the distributions they parameterize.

Theorem 1 in [19] formalizes the preceding intuition by showing that the gradient is the direction of steepest ascent only if the parameter space parameterizes a Euclidean metric. Chapter 3 in [19] shows that the Fisher information matrix is the appropriate metric for the space of probability distribution parameters. Referring back to Theorem 1 in [19] and [18], we can write the update for Natural Policy Gradient (NPG):

$$\begin{aligned} F(\theta) &= \mathbb{E}_\pi \left[\nabla \log \pi(A_t \mid, S_t, \theta) (\nabla \log \pi(A_t \mid, S_t, \theta))^\top \right] \\ \theta_{t+1} &= \theta_t + \alpha F(\theta_t)^{-1} \nabla J(\theta_t) \end{aligned} \tag{14}$$

By using the curvature information contained in the Fisher Information matrix F, our steps will be larger in flat regions compared to vanilla policy gradient, the magnitude of the gradient will be larger in flat regions (where we are presumably far from a maximum) and smaller in very steep regions (near a maximum). Therefore, NPG tends to converge faster.However, F is expensive to compute, and matrix inversion is an $O(N^3)$ operation. We can speed this up by using the conjugate gradient method [20] to compute $F^{-1}g$ without inverting F. By running fixed iteration CG as the inner loop, we get Truncated Natural Policy Gradient [18, 21].

6 Trust Region Policy Optimization

Trust Region Policy Optimization (TRPO) [21] is similar to NPG in that it seeks to avoid updates that change the policy too much at each step. Define our objective as

$$J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \right] \tag{15}$$

The algorithm hinges on an identity originally stated in Lemma 6.1 in [22], restated in Eq. 1 in [21]. Consider the update of policy π with parameters θ to a new policy π_{old} with parameters θ_{old}. Then, we have the following identity. By rewriting the result stated in the lemma as a sum over states instead of time steps, we have

$$J(\theta) = J(\theta_{old}) + \sum_s d^\pi(s) \sum_a \pi(a|s, \theta) A^{\pi_{old}}(s, a) \tag{16}$$

This shows that any policy update from π_{old} to π that has a nonnegative expected advantage at every state is guaranteed to increase policy performance J or leave it unchanged. The dependency of $d^\pi(s)$ on π makes it difficult to optimize Eq. 16 directly, so we instead define a local approximation $J_{\theta_{old}}(\theta)$ by replacing $d^\pi(s)$ with $d^{\pi_{old}}(s)$ in Eq. 16, ignoring the changes in state visitation density due to changes in policy. This is a first order approximation of J (see Eq. 4 in [22]), so a sufficiently small step in the direction of the gradient will give us improvement. We now must determine how to choose our step size.

Let $D_{KL}(\theta_{old}, \theta) \coloneqq D_{KL}(\pi(\cdot|s, \theta_{old}) \mid\mid \pi(\cdot|s, \theta))$ denote the KL divergence between our old and new policies. We penalize the objective using $D_{KL}^{\max}(\theta_{old}, \theta) = \max_s D_{KL}(\theta_{old}, \theta)$ to make sure our policy does not change too drastically between updates. Making use of the lower bound in Eq. 6 of [21] (originally derived in [22]), our problem now becomes

$$\text{maximize}_\theta \quad J_{\theta_{old}}(\theta) - C D_{KL}^{\max}(\theta_{old}, \theta) \tag{17}$$

where $C = \frac{4\varepsilon\gamma}{(1-\gamma)^2}$ and $\varepsilon = \max_{s,a} |A^\pi(s, a)|$. Eq. 9 in [21] guarantees that policies improve monotonically when solving Eq. 17. This choice of C causes us to take small steps, so in practice, we rewrite this as a constrained optimization problem. We additionally use the average KL divergence $\bar{D}_{KL}(\theta_{old}, \theta)$, as the problem is otherwise impractical to solve due to the number of constraints. Our simplified problem is thus

$$\begin{aligned} \text{maximize}_\theta \quad & J_{\theta_{old}}(\theta) \\ \text{such that} \quad & \bar{D}_{KL}(\theta_{old}, \theta) \leq \delta \end{aligned} \tag{18}$$

We now rewrite our problem using sample averages and an empirical estimate for $Q^{\pi}(s,a)$ using the single path sampling technique described in [21], which estimates $Q^{\pi}(s,a)$ in a Monte Carlo fashion.

$$\begin{aligned} \text{maximize}_{\theta} \quad & \mathbb{E}_{\pi_{old}}\left[\frac{\pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta_{old})}\widehat{Q}^{\pi_{old}}(S_t,A_t)\right] \\ \text{such that} \quad & \mathbb{E}_{\pi_{old}}\left[D_{KL}(\theta_{old},\theta)\right] \leq \delta \end{aligned} \tag{19}$$

Note that we replace the sum over actions with an importance sampling estimator similar to the work shown in Eq. 13. We can recover NPG as a special case of TRPO if we use a linear approximation to J and a quadratic approximation to $\bar{D}_{KL}$ (see Eq. 17 in [21]). It is also possible to recover vanilla policy gradient by using a Euclidean constraint on the difference in policy parameters (see Eq. 18 in [21]).

6.1 Proximal Polixy Optimization

Proximal Policy Optimization [23] achieves the data efficiency and reliability of TRPO using only first-order optimization. Instead of using an explicit constraint on KL divergence, PPO uses so-called clipped probability ratios to constrain policy updates. Define $r_t(\theta) = \frac{\pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta_{old})}$. The objective that TRPO maximizes can then be written as

$$J(\theta) = \mathbb{E}_{\pi_{old}}\left[r_t(\theta)A^{\pi_{\theta_{old}}}(S_t,A_t)\right] \tag{20}$$

Rather than imposing a KL divergence constraint, PPO proposes a new a objective

$$J^{CLIP}(\theta) = \mathbb{E}_{\pi_{old}}\left[\min(r_t(\theta)A^{\pi_{\theta_{old}}}(s,a), \text{clip}(r_t(\theta),1-\epsilon,1+\epsilon)\right] \tag{21}$$

where $\text{clip}(r_t(\theta),1-\epsilon,1+\epsilon)$ forces $r_t(\theta)$ to remain within the interval $[1-\epsilon,1+\epsilon]$. In other words, it prevents the the policy from changing too much in one update while still making sure we move the policy in the right direction. Taking the minimum assures that the objective is indeed a lower bound on the TRPO objective. Thus, the clipped term is only used when the change $r_t(\theta)$ would make the objective worse. This allows us to take large steps whenever the policies are not that different. Using this objective increases performance and greatly simplifies the implementation of TRPO by removing the KL penalty.

6.2 Phasic Policy Gradient

In the aforementioned methods, we must decide whether or not some policy parameters should be shared with value function parameters. Sharing parameters is reasonable considering that the policy and the value function are related, but it them becomes possible for the optimization of one to interfere with the optimization of the other. Phasic Policy Gradient (PPG) [24] uses separate parameters while still reaping the benefits of sharing parameters by using a two-phase approach to PPO. PPG further improves upon PPO by tuning the level of sample reuse used to learn the policy parameters vs the value function parameters, making it more sample efficient.

The first phase, the policy phase, optimizes the same objective L^{CLIP} in Eq. 21 to train the policy and uses the squared loss to the learn the value function. In the auxiliary phase, the policy network is optimized using

$$L^{joint} = L^{aux} + \beta_{clone}\mathbb{E}_{\pi_{old}}\left[KL(\theta_{old}|\theta)\right]$$

where L^{aux} can be any auxiliary objective. Note that this objective resembles Eq. 17 used in TRPO [21]. We essentially optimize L^{aux} while trying to preserve the original policy, as indicated by the KL divergence term. PPG is conjectured to outperform PPO in high dimensional input spaces, where sharing parameters matters [24].

6.3 Actor-Critic with Experience Replay

Recall A3C [9] from Section 4. A3C suffers from poor sample complexity; in order for workers to explore their respective environments, each must perform an expensive simulation step. Actor-Critic with Experience Replacy (ACER) [25], the off-policy counterpart of A3C, reduces the cost of simulation by decreasing the number of simulation steps needed for learning via *experience*

replay [16]. With experience replay, workers reuse past experience (*i.e. past trajectories*) on their respective learning algorithms.

While off-policy learning and experience replay can improve sample efficiency, it is difficult to control the variance and stability of off-policy estimators [25]. ACER combats this first by using the Retrace algorithm [26] to estimate $Q^\pi(s, a)$, an off-policy algorithm with low variance and strong convergence guarantees to reduce bias in the gradient and speed up the critic's learning.

The importance sampling estimator is also clipped above at some constant c to prevent to ratio from getting too large. An additional correction term is added back in to keep the gradient estimate unbiased (see Eq. 7 in [25]. This is called the *truncation with bias correction* trick.

Instead of using TRPO as written and computing the average KL divergence, an average policy network that represents a running average of past policies is used. Policy updates are required to not deviate too far from this average. It is worth noting that PPO algorithm has displayed the better performance on continuous control tasks and almost matches ACER's performance on Atari, despite being far simpler to implement [23].

7 Global Convergence of Policy Gradient Methods

7.1 Global convergence of Projected Gradient Ascent and Natural Policy Gradient

[1] provides a detailed study of the convergence properties of policy gradient algorithms in the tabular setting. The paper shows that when the objective function has wide coverage over the state space (*i.e.* the state space is already well-explored), we have global convergence. Otherwise, we experience the aforementioned vanishing gradient problem. A gradient domination-like property [27] is key to deriving these convergence results.

For now, we focus on projected gradient ascent over the probability simplex using the softmax parameterization. Assume we have access to exact gradients, and let μ be a starting state distribution. The gradient with the softmax parameterization is then

$$\frac{\partial V^\pi(\mu)}{\partial \theta_{s,a}} = d^\pi_\mu(s)\pi(a|s,\theta)A^\pi(s,a)$$

Intuitively, if the advantage for an action is positive, the probability of that action will be improved. If $\pi(a|s,\theta)$ is zero, then there is will be no update. It follows that we can have non-optimal stationary points if actions are not explored by the policy itself. If we assume $\mu(s) > 0$ for all states s, then Theorem 5.1 in [1] shows that our value function estimate under NPG converges to the optimal value, even though the optimization is non-convex.

However, the convergence is only asymptotic; if $\pi(a|s,\theta)$ is small, the update to the action's probability will be small, and convergence will be slow. Entropy-based regularization can help by encouraging exploration, preventing probabilities from becoming too small [28, 9]. Theorem 5.2 in [1] shows that under log-barrier regularization, stationary points of the objective are approximately globally optimal. The proof indeed relies on showing that $\pi(a|s,\theta)$ never gets too small. Corollary 5.1 then provides an iteration complexity bound.

The $d^{\pi_\theta}_\mu(s)$ factor in Theorem 5.1 of [1] is holding back the convergence rate by scaling the gradient. In particular, if we got rid of this term, we would get a large signal when the advantage is large, allowing for faster progress. If we consider NPG with the softmax parameterization, the update for π becomes identical to the multiplicative weight update from online linear optimization over the probability simplex [29], and does not depend on $d^{\pi_\theta}_\mu(s)$. This give a dimension free, fast convergence rate of $O(\frac{1}{1-\gamma)^2\epsilon})$ derived using a mirror descent-style analysis [30] (which interesting still applies even though the problem is nonconvex). This improves upon results in [30] and [31]. It is important to again emphasize that non-convex is not an issue for the methods discussed.

[1] further studies vanilla policy gradient and NPG in the function approximation setting with a restricted class of log-linear policies and additionally derives sample complexity bounds when using approximate gradients. For brevity, we do not discuss these here.

7.2 Global Convergence for Linear Quadratic Regulators

The global convergence of policy gradient for Linear Quadratic Regulators (LQR) [32] can be shown using an approach similar to the approach used in [1]. In particular, the convergence results for LQRs also rely on the gradient domination property [27].

We consider the infinite horizon LQR problem defined in [32],

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty}(x_t^T Q x_t + u_t^R T u_t)\right] \\ \text{such that} \quad & x_{t+1} = Ax_t + Bu_t, \quad x_0 \sim \mathcal{D}, \end{aligned} \tag{22}$$

where x_t is the state, u_t is the control (action), Q and R are positive definite matrices parameterizing quadratic costs, and A and B are transition matrices. The initial state x_0 is assumed to be randomly distributed according to some distribution $\mathcal{D}$. Classical results tell us that the optimal control input (our policy's action) can be written as $u_t = -K^* x_t$ [33] for a matrix K^*. The goal is to learn K^* using policy gradient.

Consider the class of policies parametrized by a matrix K generating the controls $u_t = -Kx_t$. Denote the cost of K as the objective function $J(K) = \mathbb{E}_{x_0\sim\mathcal{D}}\left[\sum_{t=0}^{\infty}(x_t^T Q x_t + u_t^R T u_t)\right]$. Our update rule for K is then $K_{t+1} = K_t - \alpha\nabla J(K_t)$. Define the unnormalized state visitation covariances as

$$\Sigma_K = \mathbb{E}\left[\sum_{t=0}^{\infty} x_t x_t^\top\right]. \tag{23}$$

If the initial covariance matrix is full rank (forcing the system to start in different directions), then subsequent Σ_K will be full rank, which then tells us that all stationary points are global optima. Corollary 5 in [32] summarizes this resul by stating that $J(K)$ is gradient dominated when $\mathbb{E}_{x_0\sim\mathcal{D}}\left[x_0 x_0^\top\right]$ is full rank. Clearly, initialization is crucial to this problem. If $J(K)$ additionally has the smoothness property, then it is easy to show that gradient dominance implies linear convergence to a global optimum [27] However, the LQR is not globally smooth. To get around this, a looser smoothness property can be used (Lemma 6 [32]):

Lemma 1 ("Almost" smoothness) *J(K) satisfies*

$$\begin{aligned} J(K') - J(K) - 2\mathit{Tr}(\Sigma_{K'}(K'-K)^\top E_K) \\ = \mathit{Tr}(\Sigma_{K'}(K-K')^\top(R + B^\top P_K B)(K-K')) \end{aligned} \tag{24}$$

where P_K is the solution to

$$P_K = Q + K^\top R K + (A - BK)^\top P_K (A - BK)$$

and

$$E_K = (R + B^\top P_K B)K - B^\top P_K A$$

We rearrange the terms in the original statement of the lemma to emphasize its relationship to smoothness. In particular, recall that a function f is said to be smooth if $\exists\beta \in \mathbb{R}$ such that $|f(x) - f(y) - \nabla f(y)^\top(x-y)| \leq \frac{\beta}{2}||x-y||^2$. When K' is sufficiently close to K, $2\text{Tr}(\Sigma_{K'}(K'-K)^\top E_K)$ behaves like $\text{Tr}((K'-K)^\top \nabla C_K)$, and the term on the right hand side is $O(||K-K'||^2)$ From "almost" smoothness with gradient dominance, global convergence can be shown despite $J(K)$ being non-convex for more than 3 dimensions (Lemma 3 in [32]), though the proof is quite technical. [32] additionally derives global convergence results for the Gauss-Newton and natural policy gradient methods, but we omit discussion on this.

References

[1] Alekh Agarwal et al. "Optimality and Approximation with Policy Gradient Methods in Markov Decision Processes". In: *CoRR* (2019).

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

[3] Peter Marbach and John N Tsitsiklis. "Simulation-based optimization of Markov reward processes". In: *IEEE Transactions on Automatic Control* (2001).

[4] R. Sutton et al. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *NIPS*. 1999.

[5] Ronald J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: (1992).

[6] Vijay R Konda and John N Tsitsiklis. "Actor-critic algorithms". In: *Advances in neural information processing systems*. 2000.

[7] Ivo Grondman et al. "A survey of actor-critic reinforcement learning: Standard and natural policy gradients". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.6 (2012), pp. 1291–1307.

[8] Thomas Degris, Martha White, and Richard S Sutton. "Off-policy actor-critic". In: *arXiv preprint arXiv:1205.4839* (2012).

[9] Volodymyr Mnih et al. "Asynchronous Methods for Deep Reinforcement Learning". In: *CoRR* (2016).

[10] Benjamin Recht et al. "Hogwild!: A lock-free approach to parallelizing stochastic gradient descent". In: *Advances in neural information processing systems* 24 (2011), pp. 693–701.

[11] Jane X Wang et al. "Learning to reinforcement learn". In: *arXiv preprint arXiv:1611.05763* (2016).

[12] Alex Irpan et al. "Off-Policy Evaluation via Off-Policy Classification". In: *CoRR* abs/1906.01624 (2019). arXiv: 1906.01624. URL: http://arxiv.org/abs/1906.01624.

[13] Philip S Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. "High-confidence off-policy evaluation". In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.

[14] Doina Precup. "Eligibility traces for off-policy policy evaluation". In: *Computer Science Department Faculty Publication Series* (2000), p. 80.

[15] Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*. Vol. 10. John Wiley & Sons, 2016.

[16] Long-Ji Lin. "Self-improving reactive agents based on reinforcement learning, planning and teaching". In: *Machine learning* 8.3-4 (1992), pp. 293–321.

[17] Alekh Agarwal, Nan Jiang, and Sham M Kakade. *Reinforcement learning: Theory and algorithms*. Tech. rep. 2019.

[18] Sham M. Kakade. "A Natural Policy Gradient". In: *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*. Ed. by Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani. 2001.

[19] Shun-Ichi Amari. "Natural gradient works efficiently in learning". In: *Neural computation* (1998).

[20] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[21] John Schulman et al. "Trust region policy optimization". In: *International conference on machine learning*. 2015.

[22] Sham Kakade and John Langford. "Approximately Optimal Approximate Reinforcement Learning". In: *Proceedings of the Nineteenth International Conference on Machine Learning*. 2002.

[23] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[24] Karl Cobbe et al. *Phasic Policy Gradient*. 2020. arXiv: 2009.04416 [cs.LG].

[25] Ziyu Wang et al. "Sample efficient actor-critic with experience replay". In: *arXiv preprint arXiv:1611.01224* (2016).

[26] Rémi Munos et al. "Safe and efficient off-policy reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1054–1062.

[27] Elijah Polak. "An historical survey of computational methods in optimal control". In: *SIAM review* 15.2 (1973), pp. 553–584.

[28] Ronald J Williams and Jing Peng. "Function optimization using connectionist reinforcement learning algorithms". In: *Connection Science* 3.3 (1991), pp. 241–268.

[29] Shai Shalev-Shwartz et al. “Online learning and online convex optimization”. In: (2011).

[30] Eyal Even-Dar, Sham M Kakade, and Yishay Mansour. “Online Markov decision processes”. In: *Mathematics of Operations Research* 34.3 (2009), pp. 726–736.

[31] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. “A theory of regularized markov decision processes”. In: *arXiv preprint arXiv:1901.11275* (2019).

[32] Maryam Fazel et al. “Global convergence of policy gradient methods for the linear quadratic regulator”. In: *arXiv preprint arXiv:1801.05039* (2018).

[33] Dimitri P Bertsekas. *Dynamic programming and optimal control*. Vol. 1. 2.