Characterizing the Gap Between Actor-Critic and Policy Gradient

Junfeng Wen [1] **Saurabh Kumar** [2] **Ramki Gummadi** [3] **Dale Schuurmans** [1 3]

Abstract

Actor-critic (AC) methods are ubiquitous in reinforcement learning. Although it is understood that AC methods are closely related to policy gradient (PG), their precise connection has not been fully characterized previously. In this paper, we explain the gap between AC and PG methods by identifying the exact adjustment to the AC objective/gradient that recovers the true policy gradient of the cumulative reward objective (PG). Furthermore, by viewing the AC method as a two-player Stackelberg game between the actor and critic, we show that the Stackelberg policy gradient can be recovered as a special case of our more general analysis. Based on these results, we develop practical algorithms, *Residual Actor-Critic* and *Stackelberg Actor-Critic*, for estimating the correction between AC and PG and use these to modify the standard AC algorithm. Experiments on popular tabular and continuous environments show the proposed corrections can improve both the sample efficiency and final performance of existing AC methods.

1. Introduction

Policy gradient (PG) (Marbach and Tsitsiklis, 2001) is the foundation of many reinforcement learning (RL) algorithms (Sutton and Barto, 2018). The basic PG method (Sutton et al., 2000) requires access to the state-action values of the current policy, which is challenging to obtain. For instance, Monte-Carlo value estimates (Williams, 1992) are unbiased but suffer from high variance and low sample efficiency. To address this issue, actor-critic (AC) methods learn a parametrized value function (critic) to estimate the state-action values. This approach has inspired many successful algorithms (Mnih et al., 2016; Lillicrap et al., 2016; Haarnoja et al., 2018a) that achieve impressive performance on a range of challenging tasks. Despite the success of AC methods, AC and PG have subtle differences that are only partially characterized in the literature (Konda and Tsitsiklis, 2000; Sutton et al., 2000). In particular, the distinction between PG and practical AC methods which use an arbitrarily parametrized critic (e.g. high-capacity function approximators such as neural networks) is unclear. In this paper, we investigate the intuition that explicitly quantifying this difference and minimizing it to increase AC's fidelity to PG can benefit practical AC methods.

A key difficulty in understanding the difference between AC and PG is the use of non-linear function approximation. A non-linear parametrization violates the compatibility requirement between the actor and critic (Sutton et al., 2000) needed to ensure equivalence of PG and the policy improvement step in AC. Additionally, the critic in AC, which estimates the policy's state-action values, can be highly inaccurate since it may suffer from bias and is not optimized to convergence. Consequently, the policy improvement step of AC may differ substantially from the corresponding PG update which uses the true state-action values.

In this paper, we precisely characterize the gap between these two policy improvement updates. We start by investigating natural objective functions for AC and show that several classic algorithms can just be seen as alternative schedules for the actor and critic updates under these objectives (Table 1). From these observations, we then quantify the gap between AC and PG from both objective and gradient perspectives. From the objective perspective, we calculate the difference between the actor objective and the original cumulative reward objective used to derive PG; while from the gradient perspective, we identify the difference of the policy improvement update in AC when using an arbitrary critic versus following the true policy gradient.

Given this understanding of the difference between PG and AC, we propose two solutions to close their gap and reduce the bias introduced by the critic in practice. First, we develop novel update rules for AC that estimate this gap and add a correction to standard updates used in AC; in particular, we propose a new AC framework, which we call Residual Actor-Critic (Res-AC). Second, we explore AC from a game-theoretic perspective and treat AC as a Stackelberg game (Fiez et al., 2020; Sinha et al., 2017). By treating

[1]Department of Computing Science, University of Alberta, Edmonton, Canada [2]Stanford University [3]Google Brain. Correspondence to: Junfeng Wen <junfengwen@gmail.com>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021.

the actor and critic as two players, we propose a second novel AC framework, Stackelberg Actor-Critic (Stack-AC), and prove that the Stack-AC updates can also close the gap between AC and PG under certain assumptions. We implement the Res-AC and Stack-AC update rules by applying them to Soft Actor-Critic (Haarnoja et al., 2018a). We present empirical results that show these modifications can improve sample efficiency and final performance in both a tabular domain as well as continuous control tasks which require neural networks to approximate the actor and critic.

2. Background

Throughout this paper we exploit a matrix-vector notation that significantly simplifies the calculations and clarifies the exposition. Therefore, we first need to introduce the notation and relevant concepts we leverage in some detail. Beyond matrix-vector notations, Appendix A provides alternative representations for some of the key concepts in this paper.

Markov Decision Process. Let $\mathbb{R}_+ = \{x \in \mathbb{R} | x \geqslant 0\}$. A *Markov Decision Process* (MDP) is defined as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathbf{r}, P, \mu_0, \gamma\}$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathbf{r} = [\mathbf{r}_{s_1}^\top, \ldots, \mathbf{r}_{s_{|\mathcal{S}|}}^\top]^\top \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times 1}$ is reward vector/function,[1] $P \in \mathbb{R}_+^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}|}$ is the transition matrix where $P_{sa,\tilde{s}} = \Pr(\tilde{s}|s,a)$ for some state $s \in \mathcal{S}$, action $a \in \mathcal{A}$ and next state $\tilde{s} \in \mathcal{S}$, $\mu_0 \in \mathbb{R}_+^{|\mathcal{S}|\times 1}$ is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. It is clear that $\mu_0^\top \mathbf{1}_{|\mathcal{S}|} = 1$ and $P\mathbf{1}_{|\mathcal{S}|} = \mathbf{1}_{|\mathcal{S}||\mathcal{A}|}$ since μ_0 and P represent probabilities, where $\mathbf{1}$ is the vector of all ones and the subscript represents its dimensionality.

Actor and Critic. A *policy* or *actor* can be represented as $\boldsymbol{\pi} \in \mathbb{R}_+^{|\mathcal{S}||\mathcal{A}|\times 1}$, and $\boldsymbol{\pi}_{s_1} \in \mathbb{R}_+^{|\mathcal{A}|\times 1}$ denotes the policy for the first state s_1. We denote the expanded matrix of $\boldsymbol{\pi}$ as

$$\Pi = \begin{bmatrix} \boldsymbol{\pi}_{s_1}^\top & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \boldsymbol{\pi}_{s_{|\mathcal{S}|}}^\top \end{bmatrix} \in \mathbb{R}_+^{|\mathcal{S}|\times|\mathcal{S}||\mathcal{A}|} \quad (1)$$

We also use $\boldsymbol{\pi}_\theta$ to denote a parametrized policy with parameters θ. In the case of a tabular softmax policy, $\boldsymbol{\pi}_\theta$ are the per-state softmax transformation of the logits $\theta \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times 1}$. We assume that $\boldsymbol{\pi}_\theta$ is properly parametrized so that it is normalized for every state.[2] A policy's state-action values (a.k.a. Q-values) are denoted as $\mathbf{q}_\theta = \sum_{i=0}^{\infty}(\gamma P \Pi_\theta)^i \mathbf{r} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times 1}$. We construct an block-diagonal matrix $Q_\theta \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}||\mathcal{A}|}$ similar to Eq. (1) for $\mathbf{q}_\theta$. In general $\mathbf{q}_\theta$ is not available, so one may approximate it using a *critic* $\mathbf{q}_\phi$ with parameters $\boldsymbol{\phi}$. In the tabular case, $\mathbf{q}_\phi = \boldsymbol{\phi} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times 1}$ is a lookup table. The actor and critic Jacobian matrices are respectively

$$(H_\theta)_{i,sa} = \left[\frac{\partial(\boldsymbol{\pi}_\theta)_{sa}}{\partial\theta_i}\right] \qquad (H_\phi)_{i,sa} = \left[\frac{\partial(\mathbf{q}_\phi)_{sa}}{\partial\phi_i}\right] \quad (2)$$

An important concept that will be used repeatedly throughout the paper is the (discounted) stationary distribution of a policy $\boldsymbol{\pi}_\theta$ over all states and actions in the MDP, where initial states are sampled from μ_0. This distribution is denoted as $\mathbf{d}_\theta \in \mathbb{R}_+^{|\mathcal{S}||\mathcal{A}|\times 1}$ and it is defined as follows:

$$d_\theta(s,a) = (1-\gamma)\sum_{i=0}^{\infty}\gamma^i \Pr(S_i = s, A_i = a | P, \boldsymbol{\pi}_\theta, S_0 \sim \mu_0).$$

A policy's stationary distribution satisfies the following recursion (Wang et al., 2007):

$$\mathbf{d}_\theta = (1-\gamma)\Pi_\theta^\top \mu_0 + \gamma \Pi_\theta^\top P^\top \mathbf{d}_\theta \quad (3)$$

We use $\mathbf{d}_{\mathcal{S},\theta} \in \mathbb{R}_+^{|\mathcal{S}|\times 1}$ to denote the stationary distribution over the states instead of the state-action pairs. More precisely, $\mathbf{d}_{\mathcal{S},\theta} = \Xi \mathbf{d}_\theta$, where Ξ is the marginalization matrix

$$\Xi = \begin{bmatrix} \mathbf{1}_{|\mathcal{A}|}^\top & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{1}_{|\mathcal{A}|}^\top \end{bmatrix} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}||\mathcal{A}|} \quad (4)$$

Furthermore, we use $D_\theta = \Delta(\mathbf{d}_\theta)$ where $\Delta(\cdot)$ maps a vector to a diagonal matrix with its elements on the main diagonal.

Policy Objective and Policy Gradient. The cumulative (discounted) reward objective, or *policy objective*, of a policy $\boldsymbol{\pi}_\theta$ can be written as (Puterman, 2014)

$$\max_\theta \; J(\theta) := (1-\gamma)\mu_0^\top \Pi_\theta \sum_{i=0}^{\infty}(\gamma P \Pi_\theta)^i \mathbf{r} \quad (5)$$

$$= (1-\gamma)\mu_0^\top \Pi_\theta \mathbf{q}_\theta \quad (6)$$

$$= \mathbf{d}_\theta^\top \mathbf{r} \quad (7)$$

where the last equation is due to duality (Wang et al. 2007, Lemma 9; Puterman 2014, Sec. 6.9).

Although the term *policy gradient* can mean any gradient of a policy in the literature, we use this term to specifically refer to the gradient of Eq. (6) throughout the paper. It is given by (Sutton et al., 2000, Thm.1)

$$\nabla_\theta J = \sum_s d_{\mathcal{S},\theta}(s) \sum_a q_\theta(s,a) \nabla_\theta \pi_\theta(s,a) \quad (8)$$

One can replace the Q-values with a critic in the policy gradient to obtain:[3]

$$\nabla_\theta^\phi J := \sum_s d_{\mathcal{S},\theta}(s) \sum_a q_\phi(s,a) \nabla_\theta \pi_\theta(s,a) \quad (9)$$

[1] A vector can represent a concept in tabular (finite dimension) or continuous space (infinite dimension). Correspondingly, the inner product $\mathbf{x}^\top \mathbf{y}$ can be interpreted as summation or integral.

[2] For example, using softmax transformation for discrete action or Beta distribution for box-constrained action (Chou et al., 2017).

[3] $\nabla_\theta^\phi J$ matches $\nabla_\theta J$ under certain technical assumptions, see Sutton et al. (2000, Thm.2) for further discussion.

An *actor-critic* (AC) method alternates between improving the policy (actor) using the critic, and estimating the policy's Q-values with a critic. AC methods are typically derived from *policy iteration* (Sutton and Barto, 2018), which alternates between policy evaluation and policy improvement. Eq. (9) can be used to update the actor in the policy improvement step.

3. Unifying Classical Algorithms

We consider a unified perspective on AC (and related) algorithms that will allow us to better understand their relationships and ultimately characterize the key differences. This will set the stage for the main contributions to follow, although the perspectives are independently useful.

In particular, we consider AC (Sutton and Barto, 2018) from two perspectives: (1) starting from the cumulative reward objective (objective perspective) and (2) from the policy gradient (gradient perspective). These two perspectives differ in where the policy's Q-values are approximated with a parametrized critic; in the first case, this is done in the policy objective (Eq. (6)) *before* computing the gradient with respect to the policy parameters, while in the second case, the approximation appears in the gradient expression (Eq. (8)) *after* taking the gradient of the policy objective.

3.1. Actor-Critic from the Objective Perspective

Consider the two standard actor-critic objectives:

$$\text{Actor: } \max_{\theta} \; J_\pi(\theta, \boldsymbol{\phi}) := (1-\gamma)\boldsymbol{\mu}_0^\top \Pi_\theta \mathbf{q}_\phi \tag{10}$$

$$\text{Critic: } \min_{\boldsymbol{\phi}} \; J_q(\theta, \boldsymbol{\phi}) := \frac{1}{2}\|\mathbf{r} + \gamma P \Pi_\theta \mathbf{q}_\phi - \mathbf{q}_\phi\|_{\mathbf{d}}^2 \tag{11}$$

$$= \frac{1}{2}(\mathbf{r} - \Psi_\theta \mathbf{q}_\phi)^\top D(\mathbf{r} - \Psi_\theta \mathbf{q}_\phi)$$

$$\text{where} \qquad \Psi_\theta := I - \gamma P \Pi_\theta$$

The *actor objective* is an approximation of Eq. (6) using a parameterized critic $\mathbf{q}_\phi$ in place of $\mathbf{q}_\theta$. To learn a critic that estimates the policy's Q-values, the *critic objective* minimizes the Bellman residual weighted by a state-action distribution $\mathbf{d}$, which is assumed to have full support. From a practical standpoint, $\mathbf{d}$ is equivalent to a (fixed) replay buffer distribution from which the state-actions are sampled. An on-policy assumption is equivalent to setting $\mathbf{d} = \mathbf{d}_\theta$, the current policy's stationary distribution. Given these objectives, the partial derivatives are[4]

$$\partial_\theta J_\pi = (1-\gamma)\sum_s \mu_0(s) \sum_a q_\phi(s,a) \nabla_\theta \pi_\theta(s,a) \tag{12}$$

$$\partial_\phi J_q = H_\phi \Psi_\theta^\top D(\Psi_\theta \mathbf{q}_\phi - \mathbf{r}) = -H_\phi \Psi_\theta^\top D \boldsymbol{\delta}_{\theta,\phi} \tag{13}$$

[4]Unlike Eq. (8), we use ∂_θ instead of ∇_θ when the quantity in question depends on both θ and $\boldsymbol{\phi}$.

$\boldsymbol{\phi}$ Update / θ Update	$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \partial_\phi J_q$	$\mathbf{q}_\phi \leftarrow \mathbf{q}_\theta$
$\theta \leftarrow \theta + \partial_\theta J_\pi$	Actor$_o$-Critic	Policy Gradient$_o$
$\theta \leftarrow \theta + \nabla_\theta^\phi J$	Actor$_g$-Critic	Policy Gradient$_g$
$\theta \leftarrow \theta^*(\mathbf{q}_\phi)$	Q-Learning†	Policy Iteration

Table 1: Algorithms based on update rules. Note that $\theta \leftarrow \theta^*(\mathbf{q}_\phi)$ is finding the greedy policy w.r.t. current $\mathbf{q}_\phi$ (Policy Improvement), while $\mathbf{q}_\phi \leftarrow \mathbf{q}_\theta$ is learning the on-policy values of current $\boldsymbol{\pi}_\theta$ (Policy Evaluation). † indicates that the semi-gradient is used for the critic.

where $\boldsymbol{\delta}_{\theta,\phi} := \mathbf{r} - \Psi_\theta \mathbf{q}_\phi$ is the *residual* of the critic. From this objective perspective, actor-critic (which we refer to as Actor$_o$-Critic) alternates between ascent on the actor and descent on the critic using their respective gradients[5]:

$$\text{Actor}_o\text{:} \qquad \theta \leftarrow \theta + \alpha\, \partial_\theta J_\pi \tag{14}$$

$$\text{Critic:} \qquad \boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha\, \partial_\phi J_q \tag{15}$$

As an example, Soft Actor-Critic (Haarnoja et al., 2018a) can be considered as using variants of these updates (with entropy regularization, as shown in Appendix C).

3.2. Actor-Critic from the Gradient Perspective

Alternatively, one can consider $\nabla_\theta^\phi J$ in Eq. (9) which applies the critic *after* the gradient is derived. A2C (Mnih et al., 2016) is an example of such an approach. A key observation is that $\nabla_\theta^\phi J$ differs from $\partial_\theta J_\pi$ (Eq. (12)) in terms of the state distribution they consider: $\partial_\theta J_\pi$ uses the initial state distribution, whereas $\nabla_\theta^\phi J$ uses the policy's stationary distribution. Therefore, using $\nabla_\theta^\phi J$, one can define an alternative actor update

$$\text{Actor}_g\text{:} \qquad \theta \leftarrow \theta + \alpha\, \nabla_\theta^\phi J. \tag{16}$$

To distinguish this from Actor$_o$-Critic, we refer to the updates (15)–(16) collectively as Actor$_g$-Critic (since the actor update is derived from a gradient perspective). In the literature, AC typically refers to what we call Actor$_g$-Critic (Sutton et al., 2000; Konda and Tsitsiklis, 2000).

3.3. Unifying and Relating Classical Algorithms

Given these two perspectives, we can now show how classical algorithms can simply be interpreted as alternative interplays between the actor and critic updates; see Table 1.

Policy Iteration, for example, alternates between policy evaluation and policy improvement. *Policy evaluation* corresponds to fully optimizing the critic to becoming the policy's

[5]For notational simplicity, we use the same learning rate α for both the actor and the critic even though this is not needed for any of our conclusions.

Q-values: $\mathbf{q}_\phi \leftarrow \mathbf{q}_\theta$. *Policy improvement* updates the policy to be greedy with respect to its Q-values. Since the critic is fully optimized, it is equivalent to a policy that is greedy with respect to the current critic: $\theta \leftarrow \theta^*(\mathbf{q}_\phi)$. As noted in Section 2, AC methods are typically derived from policy iteration. What is often overlooked, however, is that Actor_o-Critic and Actor_g-Critic present distinct instances of this general framework.

Policy Gradient$_g$. When the critic is fully optimized for every actor step (i.e., policy evaluation with infinite critic capacity $\mathbf{q}_\phi \leftarrow \mathbf{q}_\theta$), we have $\nabla_\theta^\phi J = \nabla_\theta J$, which corresponds to the classical policy gradient method (Eq. (8)). Note that this is different from **Policy Gradient**$_o$, which is defined to be $\partial_\theta J_\pi$ (Eq. (12)) with $\mathbf{q}_\phi = \mathbf{q}_\theta$ (similar to Actor_o from the objective perspective). The key difference is that Policy Gradient_g uses the the on-policy distribution $d_{S,\theta}(s)$ to weight the states, whereas Policy Gradient_o uses the initial state distribution $\mu_0(s)$.

Q-Learning. The critic gradient in Eq. (13) is the gradient of the expected squared Bellman residual, which involves double-sampling (Baird, 1995). Therefore, a *semi-gradient* (Sutton and Barto, 2018, Sec.9.3) is used in practice. If the critic is directly parametrized (i.e., $\mathbf{q}_\phi = \boldsymbol{\phi} = \mathbf{q} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times 1}$), the critic update becomes

$$\mathbf{q} \leftarrow \mathbf{q} - \alpha\, \partial_\phi^{\text{semi}} J_q = (I - \alpha D)\mathbf{q} + \alpha D \mathbf{q}' \tag{17}$$

$$\text{where} \quad \partial_\phi^{\text{semi}} J_q := -D\boldsymbol{\delta}_{\theta,\phi} \tag{18}$$

and $\mathbf{q}' := \mathbf{r} + \gamma P \Pi_\theta \mathbf{q}$ is the *target value*, whose gradient is ignored. When the policy is fully optimized w.r.t. $\mathbf{q}$ (last row of Table 1), $\gamma P\Pi_\theta \mathbf{q}$ will choose the maximum next-state value, making $\mathbf{q}'$ the usual Q-Learning target. When $D = D_\theta$ in Eq. (17), $\mathbf{q}$ is updated according to on-policy experience, yielding the on-policy Q-Learning algorithm.

4. Residual Actor-Critic

Based on the unified perspective, we now present one of our key contributions, which is a characterization of the gap between AC and PG methods, both in terms of the objectives (Section 4.1) and the gradients (Section 4.2). Then we propose a practical algorithm to reduce the gap/bias introduced by the critic in Section 4.3.

To begin, note that the actor objective (10) differs from the policy objective (6) in that the critic value function $\mathbf{q}_\phi$ is independent of the policy. Therefore, there is a discrepancy between the policy gradient $\nabla_\theta J$ and the partial derivative $\partial_\theta J_\pi$ of the actor objective.

4.1. Objective Perspective

To characterize the gap from the objective perspective, consider the difference between the policy objective (6) and the actor objective (10) in Actor_o-Critic. Using the dual formulation of the policy objective (7), the difference is:

$$\mathbf{d}_\theta^\top \mathbf{r} - (1-\gamma)\boldsymbol{\mu}_0^\top \Pi_\theta \mathbf{q}_\phi = \mathbf{d}_\theta^\top \mathbf{r} - \mathbf{d}_\theta^\top \Psi_\theta \mathbf{q}_\phi = \mathbf{d}_\theta^\top \boldsymbol{\delta}_{\theta,\phi} \tag{19}$$

where the first equality replaces $(1-\gamma)\boldsymbol{\mu}_0^\top \Pi_\theta$ with $\mathbf{d}_\theta^\top \Psi_\theta$ using Eq. (3). Therefore, the difference is $\mathbf{d}_\theta^\top \boldsymbol{\delta}_{\theta,\phi}$, which is the inner product between the stationary distribution of $\boldsymbol{\pi}_\theta$ and the on-policy residual of the critic $\mathbf{q}_\phi$ under $\boldsymbol{\pi}_\theta$. This also implies that the difference between the Actor_o-gradient, $\partial_\theta J_\pi$, and the policy gradient, $\nabla_\theta J$, is exactly equal to $\partial_\theta(\mathbf{d}_\theta^\top \boldsymbol{\delta}_{\theta,\phi})$, which we analyze below.

Note that by the product rule,

$$\partial_\theta(\mathbf{d}_\theta^\top \boldsymbol{\delta}_{\theta,\phi}) = \partial_\theta((\mathbf{d}'_\theta)^\top \boldsymbol{\delta}_{\theta,\phi}) + \partial_\theta(\mathbf{d}_\theta^\top \boldsymbol{\delta}'_{\theta,\phi}) \tag{20}$$

where $\mathbf{d}'_\theta$ and $\boldsymbol{\delta}'_{\theta,\phi}$ are treated as being independent of the policy parameter θ (i.e., without computing gradients). Then the policy gradient can be expressed as

$$\nabla_\theta J = \partial_\theta J_\pi + \partial_\theta((\mathbf{d}'_\theta)^\top \boldsymbol{\delta}_{\theta,\phi}) + \partial_\theta(\mathbf{d}_\theta^\top \boldsymbol{\delta}'_{\theta,\phi}) \tag{21}$$

The first term is given by Eq. (12), and the second term is

$$\begin{aligned}&\mathbb{E}_{(S,A)\sim d_\theta}[\partial_\theta \delta_{\theta,\phi}(S,A)]\\ &= \mathbb{E}_{(S,A)\sim d_\theta, \tilde{S}\sim P_{SA}}\left[\gamma \partial_\theta \sum_{\tilde{A}} \pi_\theta(\tilde{S},\tilde{A}) q_\phi(\tilde{S},\tilde{A})\right]\end{aligned} \tag{22}$$

To further simplify, one can use the log derivative trick (i.e., $\nabla_\theta \pi_\theta = \pi_\theta \nabla_\theta \log \pi_\theta$) so that

$$\begin{aligned}\partial_\theta J_\pi &= (1-\gamma)\mathbb{E}_{S\sim\mu_0, A\sim\pi_S}[q_\phi(S,A)\nabla_\theta \log \pi_\theta(S,A)]\\ \partial_\theta((\mathbf{d}'_\theta)^\top \boldsymbol{\delta}_{\theta,\phi}) &= \gamma \mathbb{E}_{(S,A)\sim d_\theta, \tilde{S}\sim P_{SA}, \tilde{A}\sim\pi_{\tilde{S}}}[\\ &\qquad q_\phi(\tilde{S},\tilde{A})\nabla_\theta \log \pi_\theta(\tilde{S},\tilde{A})]\end{aligned} \tag{23}$$

Note that these two terms can be combined, using the recursive definition of $\mathbf{d}_\theta$ (Eq. (3)), as

$$\begin{aligned}&\partial_\theta[J_\pi + (\mathbf{d}'_\theta)^\top \boldsymbol{\delta}_{\theta,\phi}]\\ &= \mathbb{E}_{(S,A)\sim d_\theta}[q_\phi(S,A)\nabla_\theta \log \pi_\theta(S,A)]\\ &= \sum_s d_\theta(s) \sum_a q_\phi(s,a)\nabla_\theta \pi_\theta(s,a)\end{aligned} \tag{24}$$

which is equivalent to $\nabla_\theta^\phi J$ in Eq. (9). This is somewhat surprising. Now one can see that $\nabla_\theta^\phi J$ is in fact maximizing $J_\pi + (\mathbf{d}'_\theta)^\top \boldsymbol{\delta}_{\theta,\phi}$, which is nearly identical to the policy objective except that it ignores the last term of (21), $\partial_\theta(\mathbf{d}_\theta^\top \boldsymbol{\delta}'_{\theta,\phi})$ (i.e., the dependence of $\mathbf{d}_\theta$ on θ). The following theorem summarizes the gap between policy gradient and actor gradients.

Theorem 1. *The gap between the policy gradient $\nabla_\theta J$ and $\partial_\theta J_\pi$ used in the Actor_o update is given by*

$$\nabla_\theta J - \partial_\theta J_\pi = \partial_\theta \mathbb{E}_{(S,A)\sim d_\theta}[\delta_{\theta,\phi}(S,A)]$$

and the gap between the policy gradient $\nabla_\theta J$ and $\nabla_\theta^\phi J$ used in the Actor$_g$ update is given by

$$\nabla_\theta J - \nabla_\theta^\phi J = \partial_\theta \mathbb{E}_{(S,A)\sim d_\theta}[\delta'_{\theta,\phi}(S,A)].$$

We will discuss how to estimate $\partial_\theta(\mathbf{d}_\theta^\top \boldsymbol{\delta}'_{\theta,\phi})$ in Section 4.3.

4.2. Gradient Perspective

Additional insight is gained by considering the difference between AC and PG from the gradient perspective. This section provides a different and possibly more direct way to correct $\nabla_\theta^\phi J$, where the critic replaces the on-policy values *after* the policy gradient is derived. Recall the policy gradient and $\nabla_\theta^\phi J$, and observe that they can be rewritten respectively as (see Appendix A):

$$\nabla_\theta J = H_\theta \Delta(\Xi^\top \mathbf{d}_{\mathcal{S},\theta})\mathbf{q}_\theta, \; \nabla_\theta^\phi J = H_\theta \Delta(\Xi^\top \mathbf{d}_{\mathcal{S},\theta})\mathbf{q}_\phi \quad (25)$$

Clearly, $\nabla_\theta^\phi J \neq \nabla_\theta J$ unless $\mathbf{q}_\phi$ satisfies specific conditions (Sutton et al., 2000, Thm.2). Their difference is:

$$\begin{aligned} \nabla_\theta J - \nabla_\theta^\phi J &= H_\theta \Delta(\Xi^\top \mathbf{d}_{\mathcal{S},\theta})(\mathbf{q}_\theta - \mathbf{q}_\phi) && (26)\\ &= H_\theta \Delta(\Xi^\top \mathbf{d}_{\mathcal{S},\theta})\Psi_\theta^{-1}(\mathbf{r} - \Psi_\theta \mathbf{q}_\phi) && (27)\\ &= H_\theta \Delta(\Xi^\top \mathbf{d}_{\mathcal{S},\theta})\Psi_\theta^{-1}\boldsymbol{\delta}_{\theta,\phi} && (28)\end{aligned}$$

To further simplify, consider the following theorem.

Theorem 2. *[Stationary distribution derivative] Let the derivative matrix Υ of the stationary distribution w.r.t. the policy parameters to be $\Upsilon_{i,sa} = \frac{\partial d_\theta(s,a)}{\partial \theta_i}$. Then*

$$\Upsilon = H_\theta \Delta(\Xi^\top \mathbf{d}_{\mathcal{S},\theta})\Psi_\theta^{-1}.$$

All proofs can be found in Appendix B. Using this theorem, one can see that Eq. (28) is in fact

$$\nabla_\theta J - \nabla_\theta^\phi J = \Upsilon \boldsymbol{\delta}_{\theta,\phi} = \partial_\theta(\mathbf{d}_\theta^\top \boldsymbol{\delta}'_{\theta,\phi}), \quad (29)$$

which provides an alternative way to prove the gap between PG and AC as shown in Theorem 1.

Theorem 2 in itself can be useful in some other contexts. A related result can be found in Morimura et al. (2010, Eq.(3.5)), which is a recursive form for the derivative of the log stationary distribution. Our theorem here provides a direct and explicit form for the derivative of the distribution.

4.3. Residual Actor-Critic Update Rules

The key insight from both Sections 4.1 and 4.2 is that bridging the gap between AC and PG requires the computation of $\partial_\theta(\mathbf{d}_\theta^\top \boldsymbol{\delta}'_{\theta,\phi})$. Our next main contribution is to develop a practical strategy to estimate this gap, which will reduce the bias introduced by the critic and bring the actor update in AC closer to the true policy gradient. This results in a new AC framework we call *Residual Actor-Critic*.

To develop a practical estimator, first note that $\mathbf{d}_\theta^\top \boldsymbol{\delta}'_{\theta,\phi}$ can be treated as a dual policy objective (Eq. (7)), where the environment's reward $\mathbf{r}$ is replaced by the residual $\boldsymbol{\delta}'_{\theta,\phi}$ of the critic $\mathbf{q}_\phi$. The corresponding primal objective is

$$\max_\theta \; J_\delta(\theta) := (1-\gamma)\boldsymbol{\mu}_0^\top \Pi_\theta \mathbf{w}_\theta \quad (30)$$

where $\mathbf{w}_\theta = \sum_{i=0}^\infty (\gamma P \Pi_\theta)^i \boldsymbol{\delta}'_{\theta,\phi}$ is the on-policy Q-value associated with the residual reward. The gradient of (30) is precisely the desired correction, $\partial_\theta(\mathbf{d}_\theta^\top \boldsymbol{\delta}'_{\theta,\phi})$. Computing its gradient requires $\mathbf{w}_\theta$, which can be approximated by introducing a *residual-critic* (or *res-critic*) $\mathbf{w}_\psi$ with parameters $\boldsymbol{\psi}$. Concretely, the res-critic solves the following problem:

$$\min_{\boldsymbol{\psi}} \; J_w(\boldsymbol{\psi}; \theta, \boldsymbol{\phi}) := \frac{1}{2}\|\boldsymbol{\delta}'_{\theta,\phi} + \gamma P \Pi_\theta \mathbf{w}_\psi - \mathbf{w}_\psi\|_{\mathbf{d}}^2 \quad (31)$$

Once we have a relatively accurate res-critic $\mathbf{w}_\psi$, we apply the PG$_g$ method (see Table 1) and use the following to approximate $\partial_\theta(\mathbf{d}_\theta^\top \boldsymbol{\delta}'_{\theta,\phi})$

$$\nabla_\theta^\psi J_\delta := \sum_s d_\theta(s) \sum_a w_\psi(s,a) \partial_\theta \pi_\theta(s,a) \quad (32)$$

Combining these update rules for the actor and res-critic with the standard AC update rules results in our Residual Actor-Critic (Res-AC) framework, which can be summarized as follows:

$$\begin{aligned} &\text{Actor:} && \theta \leftarrow \theta + \alpha\,(\nabla_\theta^\phi J + \nabla_\theta^\psi J_\delta) && (33)\\ &\text{Critic:} && \boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha\,\partial_\phi J_q && (34)\\ &\text{Res-Critic:} && \boldsymbol{\psi} \leftarrow \boldsymbol{\psi} - \alpha\,\partial_\psi J_w && (35)\end{aligned}$$

To understand the correction term $\nabla_\theta^\psi J_\delta$ intuitively, note that the residual reward $\boldsymbol{\delta}'_{\theta,\phi}$ is signed. For an (s,a) with $\delta'_{\theta,\phi}(s,a) > 0$, we have $q_\phi(s,a) < r(s,a) + \gamma\mathbb{E}[q_\phi(\tilde{s},\tilde{a})]$, and the agent is incentivized to visit the underestimated region. On the other hand, if $\delta'_{\theta,\phi}(s,a) < 0$, the agent is discouraged to visit the overestimated location.

Res-AC is a generic framework that can be combined with different AC-based algorithms. In Appendix C and in the experiments, we show that Soft Actor-Critic (SAC) (Haarnoja et al., 2018a) can be enhanced with a res-critic and the resultant Res-SAC method improves over the original SAC.

5. Stackelberg Actor-Critic as a Special Case

Before proceeding to an experimental evaluation of the new Res-AC framework, we first present another, somewhat surprising finding that the results above are also consistent with the characterization of AC as a Stackelberg game. That is, previously we focused on correcting the actor's gradient in both Actor$_\text{o}$-Critic and Actor$_\text{g}$-Critic to obtain the true policy gradient, whereas now we consider the interplay between actor and critic from a game-theoretic perspective. Here we will be able to show that when treating AC as a Stackelberg game (Sinha et al., 2017), the *Stackelberg policy gradient* is in fact the true policy gradient under certain conditions. This will follow as a special case of the analysis in Section 4.1. Moreover, we show that even when the critic update is based on semi-gradient (i.e., with a fixed target), the Stackelberg policy gradient remains unbiased.

For this section we restrict the AC formulation by adding the following assumption.

Assumption 1. *The critic is directly parametrized* $\mathbf{q}_\phi = \boldsymbol{\phi} = \mathbf{q}$. *The critic loss is weighted by the on-policy distribution* $\mathbf{d} = \mathbf{d}_\theta$ *in Eq.(11).* μ_0 *and* π_θ *have full support.*

5.1. Actor-Critic as a Stackelberg Game

Actor-critic methods can be considered as a two-player general-sum game, where the actor and critic are the players and the objectives (Eqs.(10) and (11)) are their respective utility/cost functions. More specifically, one can treat actor-critic as a *Stackelberg game* in which there is a *leader* who moves first and a *follower* who moves subsequently (Sinha et al., 2017). By treating the actor as the leader, Stackelberg Actor-Critic (Stack-AC) solves the following

$$\max_{\theta} \left\{ J_\pi(\theta, \mathbf{q}) \,\middle|\, \mathbf{q} \in \operatorname*{argmin}_{\tilde{\mathbf{q}}} \; J_q(\theta, \tilde{\mathbf{q}}) \right\} \tag{36}$$

$$\min_{\mathbf{q}} J_q(\theta, \mathbf{q}) \tag{37}$$

A key distinction from the original AC is that the actor is now aware of the critic's goal. Given that in the ideal case $\mathbf{q}$ is implicitly a function of θ (i.e., policy evaluation), one may differentiate through $\mathbf{q}$ to obtain the following *Stackelberg gradient*[6] (Fiez et al., 2020)

$$\mathbf{g}_{S,\theta} := \partial_\theta J_\pi - (\partial_\theta \partial_q J_q)^\top (\partial_q^2 J_q)^{-1} (\partial_q J_\pi) \tag{38}$$

The second order derivative $\partial_q^2 J_q$ can be computed, based on the critic objective Eq.(11), as

$$\partial_q^2 J_q = \Psi_\theta^\top D_\theta \Psi_\theta \tag{39}$$

which is invertible under Assumption 1, hence the Stackelberg gradient is well-defined. However, it is unclear what this gradient achieves in the AC setting. We show, in the following theorem, that the Stackelberg gradient is in fact the policy gradient of the cumulative reward objective (6).

[6]It is the total derivative from the implicit function theorem.

Theorem 3. *Under Assumption 1,*

$$\mathbf{g}_{S,\theta} = \partial_\theta J_\pi + \partial_\theta(\mathbf{d}_\theta^\top \boldsymbol{\delta}_{\theta,\phi}) = \nabla_\theta J.$$

This indicates that, under some conditions, one can compute the gradient of the objective correction in Eq.(19) using $(\partial_\theta \partial_q J_q)^\top (\partial_q^2 J_q)^{-1} (\partial_q J_\pi)$.

5.2. Semi-Gradient Extension

As discussed earlier in Section 3.3, it is common to use a semi-gradient update for the critic to address the double sampling issue. This section shows that, surprisingly, the Stackelberg gradient remains the true policy gradient even when using semi-gradient for the critic.

From the semi-gradient (18), the *semi-Hessian* is

$$(\partial_q^{\text{semi}})^2 J_q = \partial_q (D_\theta(\mathbf{q} - \mathbf{q}')) = D_\theta. \tag{40}$$

Compared to Eq.(39), the derivative does not go through the next-state values so Ψ_θ is now replaced by the identity matrix. Additionally, the actor objective can be reformulated using Eq.(3) as

$$J_\pi(\theta, \mathbf{q}) = \mathbf{d}_\theta^\top \Psi_\theta \mathbf{q} = \mathbf{d}_\theta^\top (\mathbf{q} - \mathbf{q}') \tag{41}$$

Thus the semi-derivative is

$$\partial_q^{\text{semi}} J_\pi = \mathbf{d}_\theta \tag{42}$$

Then the Stackelberg gradient based on semi-critic-gradient is given by

$$\mathbf{g}_{S,\theta}^{\text{semi}} := \partial_\theta J_\pi - (\partial_\theta \partial_q^{\text{semi}} J_q)^\top ((\partial_q^{\text{semi}})^2 J_q)^{-1} (\partial_q^{\text{semi}} J_\pi) \tag{43}$$

$$= \partial_\theta J_\pi - (\partial_\theta(-D_\theta \boldsymbol{\delta}_{\theta,\phi}))^\top D_\theta^{-1} \mathbf{d}_\theta \tag{44}$$

$$= \partial_\theta J_\pi - (\partial_\theta(-D_\theta \boldsymbol{\delta}_{\theta,\phi}))^\top \mathbf{1}_{|\mathcal{S}||\mathcal{A}|} \tag{45}$$

$$= \partial_\theta J_\pi + \partial_\theta(\mathbf{d}_\theta^\top \boldsymbol{\delta}_{\theta,\phi}) = \mathbf{g}_{S,\theta} = \nabla_\theta J. \tag{46}$$

which again corresponds to the true policy gradient.

5.3. Stack-AC Update Rules

The update rules for Stack-AC are summarized as follows

$$\text{Actor:} \qquad \theta \leftarrow \theta + \alpha\, \mathbf{g}_{S,\theta}^{\text{semi}} \tag{47}$$

$$\text{Critic:} \qquad \mathbf{q} \leftarrow \mathbf{q} - \alpha\, \partial_q^{\text{semi}} J_q \tag{48}$$

where the Stackelberg gradient $\mathbf{g}_{S,\theta}^{\text{semi}}$ can be approximated using sample-based estimate for each term in Eq.(43). Although Eq.(43) requires an inverse-Hessian-vector product and a Jacobian-vector product, they can be efficiently carried out or approximated using standard libraries (Fiez et al., 2020). Following Fiez et al. (2020), we use a regularized version where $(\partial_q^{\text{semi}})^2 J_q$ is replaced by $(\partial_q^{\text{semi}})^2 J_q + \eta I$ with

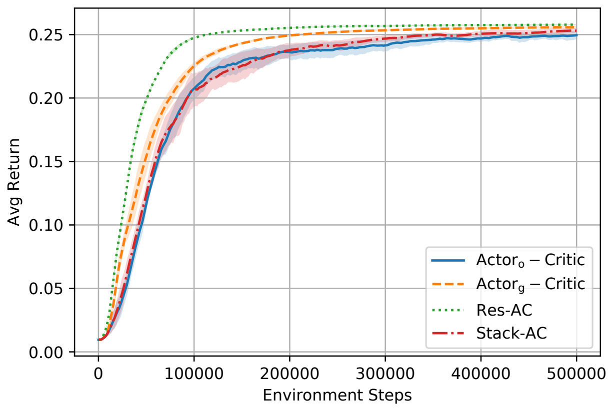


Figure 1: Comparison of Actor$_o$-Critic, Actor$_g$-Critic, Stack-AC, and Res-AC on the FourRoom domain, plotting mean with one standard deviation as shaded region over 3 runs. Res-AC outperforms the other methods in both sample efficiency and final performance.

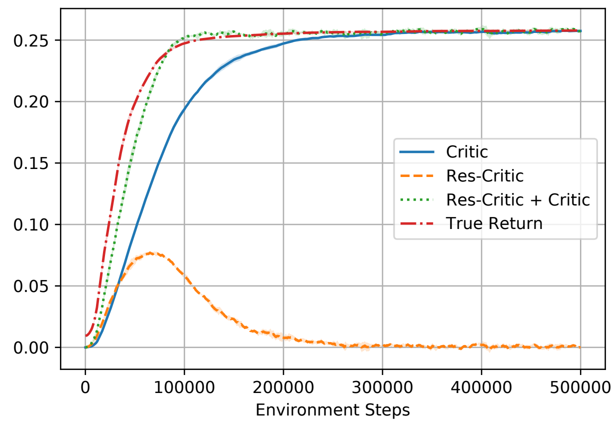


Figure 2: The predicted returns of critic and residual critic for the FourRoom domain. As training proceeds, the sum of the critic and res-critic closely approximates the true return.

$\eta = 0.5$ in our experiments (Section 6). This can ensure invertibility and stabilize learning.

There is one caveat when estimating $\partial_\theta \partial_q^{\text{semi}} J_q$ in $\mathbf{g}_{S,\theta}^{\text{semi}}$ from samples. The critic objective can be written as $J_q = \frac{1}{2}\mathbf{d}_\theta^\top \boldsymbol{\delta}_{\theta,\phi}^2$. $\partial_q^{\text{semi}} J_q$ can be estimated using a batch $\mathcal{B}$ of samples drawn from $\mathbf{d}_\theta$. However, $\partial_\theta \partial_q^{\text{semi}} J_q = \partial_\theta(-D_\theta \boldsymbol{\delta}_{\theta,\phi})$ is now difficult to estimate because we cannot take derivative of θ through the batch $\mathcal{B}$, which represents the derivative through D_θ. As a result, a sample-based estimate of $\partial_\theta \partial_q^{\text{semi}} J_q$ is only estimating $\partial_\theta(-D'_\theta \boldsymbol{\delta}_{\theta,\phi})$, where D'_θ is consider a fixed distribution unrelated to θ.

To summarize, the Stackelberg policy gradient can close the gap between AC and PG under certain conditions, even with semi-gradient updates. Despite being biased when approximated using samples, we will show in our experiments that Stack-AC can work reasonably well in practice.

6. Experiments

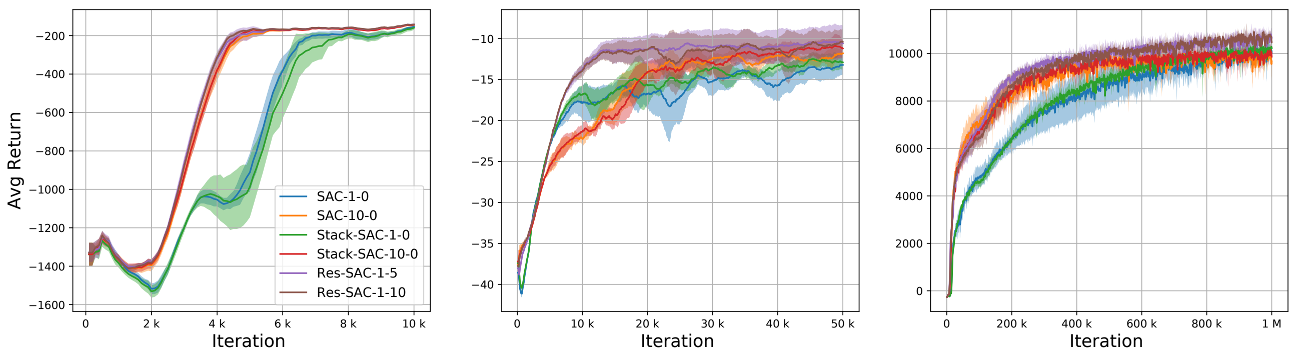
The goal of our experiments is to test whether closing the gap between the actor's update in actor-critic (Eq. (12)) and the policy gradient (Eq. (8)) leads to improved sample efficiency and performance over actor-critic methods. To this end, we conduct experiments within both the FourRoom domain, an illustrative discrete action space environment (see Appendix D), and three continuous control environments: Pendulum-v0, Reacher-v2, and HalfCheetah-v2. The environments Reacher-v2 and HalfCheetah-v2 use the MuJoCo physics engine (Todorov et al., 2012).

On the FourRoom domain, we compare Actor$_o$-Critic and Actor$_g$-Critic to Res-AC and Stack-AC. For our continuous control experiments, we modify the actor update of Soft Actor-Critic (SAC) (Haarnoja et al., 2018a), a popular maximum entropy reinforcement learning method, using the updates given by Res-AC and Stack-AC. We refer to the resulting methods as Res-SAC and Stack-SAC, respectively, and we compare SAC to Res-SAC and Stack-SAC on the continuous control tasks. A complete derivation of the modified updates of Res-SAC and Stack-SAC can be found in Appendix C. Additional training details, including hyper-parameter settings and pseudocode, and additional experimental results are in Appendix D.

6.1. Tabular Experiment

In the FourRoom domain, we use a tabular parameterization $\mathbf{q}$ for the critic and a softmax tabular actor $\boldsymbol{\pi}$. All our AC methods collect data from the environment and compute updates for the actor and critic (and res-critic for Res-AC) using this data. We train each algorithm with three different random seeds, and we plot the return of the current policy after each episode, where every episode has a fixed length of 300 environment steps (Fig. 1). The curves correspond to the mean return and the shaded region to one standard deviation over the three trials.

Res-AC enjoys improved sample-efficiency as well as final performance when compared to all other methods. Stack-AC and Actor$_o$-Critic achieve a lower final performance than Res-SAC, and they perform comparably to each other. Actor$_g$-Critic achieves a similar final return as Res-AC, but requires over 150,000 additional environment steps. The relatively poor performance of Stack-AC is not surprising since the sample-based estimate of $\partial_\theta \partial_q^{\text{semi}} J_q$ is inaccurate, as discussed in Section 5.3. In contrast, even though Res-AC introduces an additional problem of using a res-critic to learn the on-policy return with $\boldsymbol{\delta}'_{\theta,\phi}$ as reward, the res-critic significantly accelerates the improvement of the actor.



(a) Pendulum-v0 (b) Reacher-v2 (c) HalfCheetah-v2

Figure 3: Training curves on continuous control tasks. Res-AC consistently outperforms SAC and Stack-SAC on all three environments. Reporting mean with one standard deviation as shaded region over 5 runs. Each iteration on the x-axis corresponds to 10 environment steps.

To gain further understanding of the residual critic, Fig. 2 plots the predictions of the critic and res-critic for Res-AC on the FourRoom domain. The sum of the critic and res-critic approximates the true return better than the critic alone throughout training. This shows that the res-critic can correct the bias introduced by the critic in policy gradient empirically.

6.2. Continuous Control Experiments

We compare SAC to Res-SAC and Stack-AC on Pendulum-v0, Reacher-v2, and HalfCheetah-v2. We additionally introduce an update schedule for an algorithm $\mathtt{Algo}$ labelled as "$\mathtt{Algo}$-x-y", where x and y are the number of gradient updates applied to the critic and res-critic, respectively, for each actor gradient update. For example, SAC-10-0 refers to SAC with 10 critic gradient updates per actor update. (Note that the number of res-critic updates here is 0 since SAC does not use a res-critic.) In the original SAC algorithm, only one critic update is performed per actor update. Our decision to optionally perform multiple gradient updates for the critic / res-critic is guided by the intuition that a more accurate critic / res-critic would benefit the gradient updates to the policy.

For SAC, Stack-SAC, and Res-SAC, we use the hyper-parameters used in Haarnoja et al. (2018b) on all environments. Stack-SAC includes an additional regularization parameter η, introduced in Section 5.3, which we set to 0.5 for all experiments. Even though the theoretical properties of Stackelberg gradient may not hold for continuous environments (Theorem 3), we can still implement it for our experiments and investigate its performance. The performance of Res-SAC is dependent on clipping the residual that is used as the reward for the res-critic. Concretely, for a clip value $c > 0$, the reward for updating the res-critic is computed as $\text{clip}(\delta_{\theta,\phi}, -c, c)$. Without clipping, large values of the critic's residual can make training the res-critic unstable. Additional discussion on the choice of clipping value can be found in Appendix D.2.2.

The results in Fig.3 show that Res-SAC is consistently more sample-efficient than the other methods and even achieves higher asymptotic performance on Reacher-v2. Applying multiple critic updates per actor-update significantly improves the performance of SAC, as SAC-10-0 consistently outperforms SAC-1-0. The better performance of Res-SAC over SAC-10-0 suggests that bringing the actor update closer to the true policy gradient in theory translates to empirical benefits. In contrast, Stack-SAC performs comparably to SAC – there is not a clear benefit to using the Stack-SAC actor update rule over the standard SAC actor update. This is understandable as Assumption 1 does not hold when the critic uses non-linear function approximation.

7. Related Work

Actor-critic and policy gradient. We review prior works which analyze the relationship between policy gradient and actor-critic and use policy iteration to derive AC methods. Actor-critic is typically derived following our Actor$_g$-Critic derivation (Mnih et al., 2016; Lillicrap et al., 2016; Liu et al., 2020; Degris et al., 2012; Peters and Schaal, 2008), and replacing the on-policy values in the policy gradient with a critic can retain the policy gradient under certain assumptions (Konda and Tsitsiklis, 2000; Sutton et al., 2000). In this paper, we show that replacing the on-policy values with a critic corresponds to a partial policy gradient (Theorem 1). We also introduce an alternative derivation of actor-critic from an objective perspective, which has been less explored in the literature, and existing algorithms can be better understood using this perspective. As an example, while the motivation and derivation for SAC (Haarnoja et al., 2018a) is based on policy iteration, its policy improvement step of minimizing a KL divergence objective per state, can be explained by the actor objective in our Actor$_o$-Critic framework (see Appendix C for more details).

Several recent works (Ghosh et al., 2020; O'Donoghue et al., 2017; Schulman et al., 2018; Nachum et al., 2017) demonstrate connections between policy and value based methods, which also apply to actor-critic methods. Other advancements such as TRPO (Schulman et al., 2015b), GAE (Schulman et al., 2016), and TD3 (Fujimoto et al., 2018) can be considered as orthogonal to our analysis, and can be integrated into our Res-AC and Stack-AC updates.

Learning with Bellman residual. Our Res-AC approach uses the the *residual of the critic* to facilitate learning. It is slightly different from the *advantage function* (Schulman et al., 2016): the former is an approximation error of the critic (which would be zero for the perfect critic) while the latter represents relative gain of an action (which would not be zero unless all actions in the same state have the same value). GTD2 and TDC (Sutton et al., 2009; Maei et al., 2009) learn a form of the residual and use it to update the policy in the linear function approximation regime. In contrast, Res-AC learns a res-critic which is a value function of the residual. Sun et al. (2011) showed that a value function of the residual can be used as an ideal feature/basis to assist learning the reward value function when using linear function approximation. Our analysis is more general, as we show that learning the critic residual value function can be used to reconstruct the true policy gradient for arbitrary policy parametrization.

The residual (or TD error) is used in some other contexts to facilitate training RL agents such as prioritizing which data to sample from a replay buffer to perform updates (Schaul et al., 2016; Van Seijen and Sutton, 2013), or estimating the variance of the return (Sherstan et al., 2018). Res-AC is a more direct approach of leveraging the residual to improve the policy, since the res-critic is used in the actor update. Additionally, Dabney et al. (2020) showed that dopamine neurons can respond to the prediction error differently, suggesting that the residual of the value function plays an important role biologically.

Game-theoretic perspective. The concept of a differential Stackelberg equilibrium was proposed in Fiez et al. (2020) with a focus on the convergence dynamics for learning Stackelberg games. A game theoretic framework for model based RL was proposed in Rajeswaran et al. (2020), where the authors consider a Stackelberg game between an actor maximizing rewards and an agent learning an explicit model of the environment. It is computationally expensive as it requires solving actor/model to the optimum in every iteration. Our Stack-AC, on the other hand, models the actor and critic as the players and shows connections between Stackelberg gradient and true policy gradient, even when using semi-gradient for the critic. Sinha et al. (2017) provides a more comprehensive review on general bi-level optimization.

8. Conclusion

In this work, we characterize the gap between actor-critic and policy gradient methods. By defining the objective functions for the actor and the critic, we elucidate the connections between several classic RL algorithms. Our theoretical results identify the gap between AC and PG from both objective and gradient perspectives, and we propose Res-AC, which closes this gap. Additionally, by viewing AC as a Stackelberg game, we show that the Stackelberg policy gradient is the true policy gradient under certain conditions. An empirical study on tabular and continuous environments illustrates that applying Res-AC modifications to update rules of actor-critic methods improves sample efficiency and performance. Investigating the convergence guarantees of Res-AC and developing Stack-AC methods where the critic is the leader are exciting directions for future work.

9. Acknowledgements

We would like to thank Robert Dadashi, Yundi Qian and anonymous reviewers for constructive feedback. This work is partially supported by NSERC, Amii, a Canada CIFAR AI Chair, an NSF Graduate Research Fellowship, and the Stanford Knight Hennessy Fellowship.

References

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

Chou, P.-W., Maturana, D., and Scherer, S. (2017). Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *International conference on machine learning*, pages 834–843. PMLR.

Dabney, W., Kurth-Nelson, Z., Uchida, N., Starkweather, C. K., Hassabis, D., Munos, R., and Botvinick, M. (2020). A distributional code for value in dopamine-based reinforcement learning. *Nature*, 577(7792):671–675.

Degris, T., White, M., and Sutton, R. S. (2012). Off-policy actor-critic. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 179–186.

Fiez, T., Chasnov, B., and Ratliff, L. (2020). Implicit learning dynamics in stackelberg games: Equilibria characterization, convergence analysis, and empirical study. In *International Conference on Machine Learning (ICML)*.

Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR.

Ghosh, D., Machado, M. C., and Roux, N. L. (2020). An operator view of policy gradient methods. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018a). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. (2018b). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

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

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *ICLR*.

Liu, Y., Swaminathan, A., Agarwal, A., and Brunskill, E. (2020). Off-policy policy gradient with stationary distribution correction. In *Uncertainty in Artificial Intelligence*, pages 1180–1190. PMLR.

Maei, H. R., Szepesvari, C., Bhatnagar, S., Precup, D., Silver, D., and Sutton, R. S. (2009). Convergent temporal-difference learning with arbitrary smooth function approximation. In *NIPS*, pages 1204–1212.

Marbach, P. and Tsitsiklis, J. N. (2001). Simulation-based optimization of markov reward processes. *IEEE Transactions on Automatic Control*, 46(2):191–209.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

Morimura, T., Uchibe, E., Yoshimoto, J., Peters, J., and Doya, K. (2010). Derivatives of logarithmic stationary distributions for policy gradient reinforcement learning. *Neural computation*, 22(2):342–376.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. (2017). Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*.

O'Donoghue, B., Munos, R., Kavukcuoglu, K., and Mnih, V. (2017). Combining policy gradient and q-learning. In *ICLR*.

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

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Rajeswaran, A., Mordatch, I., and Kumar, V. (2020). A game theoretic framework for model-based reinforcement learning. In *International Conference on Machine Learning*.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. In *ICLR*.

Scherrer, B. (2010). Should one compute the temporal difference fix point or minimize the bellman residual? the unified oblique projection view. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 959–966.

Schulman, J., Chen, X., and Abbeel, P. (2018). Equivalence between policy gradients and soft q-learning. *ArXiv:1704.06440*. URL.

Schulman, J., Heess, N., Weber, T., and Abbeel, P. (2015a). Gradient estimation using stochastic computation graphs. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 3528–3536.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015b). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. In *ICLR*.

Sherstan, C., Ashley, D. R., Bennett, B., Young, K., White, A., White, M., and Sutton, R. S. (2018). Comparing direct and indirect temporal-difference methods for estimating the variance of the return. In *UAI*, pages 63–72.

Sinha, A., Malo, P., and Deb, K. (2017). A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295.

Sun, Y., Gomez, F., Ring, M., and Schmidhuber, J. (2011). Incremental basis construction from temporal difference error. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 481–488.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. (2009). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 993–1000.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE.

Van Seijen, H. and Sutton, R. (2013). Planning by prioritized sweeping with small backups. In *International Conference on Machine Learning*, pages 361–369. PMLR.

Wang, T., Bowling, M., and Schuurmans, D. (2007). Dual representations for dynamic programming and reinforcement learning. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 44–51. IEEE.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, pages 229–256.