GUIDE ACTOR-CRITIC FOR CONTINUOUS CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Actor-critic methods solve reinforcement learning problems by updating a parameterized policy known as an actor in a direction that increases an estimate of the expected return known as a critic. However, existing actor-critic methods only use values or gradients of the critic to update the policy parameter. In this paper, we propose a novel actor-critic method called the guide actor-critic (GAC). GAC firstly learns a guide actor that locally maximizes the critic and then it updates the policy parameter based on the guide actor by supervised learning. Our main theoretical contributions are two folds. First, we show that GAC updates the guide actor by performing second-order optimization in the action space where the curvature matrix is based on the Hessians of the critic. Second, we show that the deterministic policy gradient method is a special case of GAC when the Hessians are ignored. Through experiments, we show that our method is a promising reinforcement learning method for continuous controls.

1 INTRODUCTION

The goal of reinforcement learning is to learn an optimal policy that lets an agent achieve the maximum cumulative rewards known as the return (Sutton & Barto, 1998). Reinforcement learning has been shown to be effective in solving challenging artificial intelligence tasks such as playing games (Mnih et al., 2015; Silver et al., 2016) and controlling robots (Deisenroth et al., 2013; Levine et al., 2016).

Reinforcement learning methods can be classified into three categories: *value-based*, *policy-based*, and *actor-critic* methods. Value-based methods learn an optimal policy by firstly learning a *value function* that estimates the expected return. Then, they infer an optimal policy by choosing an action that maximizes the learned value function. Choosing an action in this way requires solving a maximization problem which is not trivial for continuous controls. While extensions to continuous controls were considered recently, they are restrictive since specific structures of the value function are assumed (Gu et al., 2016; Amos et al., 2017).

On the other hand, policy-based methods, also called policy search methods (Deisenroth et al., 2013), learn a parameterized policy maximizing a sample approximation of the expected return without learning the value function. For instance, policy gradient methods such as REINFORCE (Williams, 1992) use gradient ascent to update the policy parameter so that the probability of observing high sample returns increases. Compared with value-based methods, policy search methods are simpler and naturally applicable to continuous problems. Moreover, the sample return is an unbiased estimator of the expected return and methods such as policy gradients are guaranteed to converge to a locally optimal policy under standard regularity conditions (Sutton et al., 1999). However, sample returns usually have high variance and this makes such policy search methods converge too slowly.

Actor-critic methods combine the advantages of value-based and policy search methods. In these methods, the parameterized policy is called an actor and the learned value-function is called a critic. The goal of these methods is to learn an actor that maximizes the critic. Since the critic is a low variance estimator of the expected return, these methods often converge much faster than policy search methods. Prominent examples of these methods are *actor-critic* (Sutton et al., 1999; Konda & Tsitsiklis, 2003), *natural actor-critic* (Peters & Schaal, 2008), *trust-region policy optimization* (Schulman et al., 2015a), and *asynchronous advantage actor-critic* (Mnih et al., 2016). While their approaches to learn the actor are different, they share a common property that they only use the *value*

of the critic, i.e., the zero-th order information, and ignore higher-order ones such as gradients and Hessians w.r.t. actions of the critic[1]. To the best of our knowledge, the only actor-critic methods that use gradients of the critic to update the actor are *deterministic policy gradients* (DPG) (Silver et al., 2014) and *stochastic value gradients* (Heess et al., 2015). However, these two methods do not utilize the second-order information of the critic.

In this paper, we argue that the second-order information of the critic is useful and should not be ignored. A motivating example can be seen by comparing gradient ascent to the Newton method: the Newton method which also uses the Hessian converges to a local optimum in a fewer iterations when compared to gradient ascent which only uses the gradient (Nocedal & Wright, 2006). This suggests that the Hessian of the critic can accelerate actor learning which leads to higher data efficiency. However, the computational complexity of second-order methods is at least quadratic in terms of the number of optimization variables. For this reason, applying second-order methods to optimize the parameterized actor directly is prohibitively expensive and impractical for deep reinforcement learning which represents the actor by deep neural networks.

Our contribution in this paper is a novel actor-critic method for continuous controls which we call *guide actor-critic* (GAC). Unlike existing methods, the actor update of GAC utilizes the second-order information of the critic in a computationally efficient manner. This is achieved by separating actor learning into two steps. In the first step, we learn a *non-parameterized Gaussian actor* that locally maximizes the critic under a Kullback-Leibler (KL) divergence constraint. Then, the Gaussian actor is used as a guide for learning a parameterized actor by supervised learning. Our analysis shows that learning the mean of the Gaussian actor is equivalent to performing a second-order update in the action space where the curvature matrix is given by Hessians of the critic and the step-size is controlled by the KL constraint. Furthermore, we establish a connection between GAC and DPG where we show that DPG is a special case of GAC when the Hessians and KL constraint are ignored.

2 Background

In this section, we firstly give a background of reinforcement learning. Then, we discuss existing second-order methods for policy learning and their issue in deep reinforcement learning.

2.1 Reinforcement Learning

We consider discrete-time Markov decision processes (MDPs) with continuous state space $\mathcal{S} \subseteq \mathbb{R}^{d_{\mathbf{s}}}$ and continuous action space $\mathcal{A} \subseteq \mathbb{R}^{d_{\mathbf{a}}}$. We denote the state and action at time step $t \in \mathbb{N}$ by $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$, respectively. The initial state $\boldsymbol{s}_1$ is determined by the initial state density $\boldsymbol{s}_1 \sim p(\boldsymbol{s})$. At time step t, the agent in state $\boldsymbol{s}_t$ takes an action $\boldsymbol{a}_t$ according to a policy $\boldsymbol{a}_t \sim \pi(\boldsymbol{a}|\boldsymbol{s}_t)$ and obtains a reward $r_t = r(\boldsymbol{s}_t, \boldsymbol{a}_t)$. Then, the next state $\boldsymbol{s}_{t+1}$ is determined by the transition function $\boldsymbol{s}_{t+1} \sim p(\boldsymbol{s}'|\boldsymbol{s}_t, \boldsymbol{a}_t)$. A trajectory $\tau = (\boldsymbol{s}_1, \boldsymbol{a}_1, r_1, \boldsymbol{s}_2, \dots)$ gives us cumulative rewards or returns defined as $\sum_{t=1}^{\infty} \gamma^{t-1} r(\boldsymbol{s}_t, \boldsymbol{a}_t)$, where the discount factor $0 < \gamma < 1$ assigns different weights to rewards given at different time steps. The expected returns of π after executing an action $\boldsymbol{a}$ in a state $\boldsymbol{s}$ can be expressed through the *action-value function* which is defined as

$$Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)_{t\geqslant 2}, p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)_{t\geqslant 1}} \left[\sum_{t=1}^{\infty} \gamma^{t-1} r(\boldsymbol{s}_t, \boldsymbol{a}_t) | \boldsymbol{s}_1 = \boldsymbol{s}, \boldsymbol{a}_1 = \boldsymbol{a} \right], \tag{1}$$

where $\mathbb{E}_p[\cdot]$ denotes the expectation over the density p and the subscript $t \geqslant 1$ indicates that the expectation is taken over the densities at time steps $t \geqslant 1$. We can define the expected returns as

$$\mathcal{J}(\pi) = \mathbb{E}_{p(\boldsymbol{s}_1), \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)_{t\geqslant 1}, p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)_{t\geqslant 1}} \left[\sum_{t=1}^{\infty} \gamma^{t-1} r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right] = \mathbb{E}_{p(\boldsymbol{s}), \pi(\boldsymbol{a}|\boldsymbol{s})} \left[Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}) \right]. \tag{2}$$

The goal of reinforcement learning is to find an optimal policy that maximizes the expected returns.

The policy search approach (Deisenroth et al., 2013) parameterizes π by a parameter $\boldsymbol{\theta} \in \mathbb{R}^{d_{\boldsymbol{\theta}}}$ and finds $\boldsymbol{\theta}^{\star}$ which maximizes the expected returns:

$$\boldsymbol{\theta}^{\star} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{s}), \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})} \left[Q^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{s}, \boldsymbol{a}) \right]. \tag{3}$$

[1]This is different from using the gradient of the critic w.r.t. critic parameters to update the critic itself.

Policy gradient methods such as REINFORCE (Williams, 1992) solve this optimization problem by gradient ascent:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \mathbb{E}_{p(\boldsymbol{s}),\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) Q^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{s},\boldsymbol{a}) \right], \tag{4}$$

where $\alpha > 0$ is a step-size. In policy search, the action-value function is commonly estimated by the *sample return*: $Q^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{s},\boldsymbol{a}) \approx \frac{1}{N}\sum_{n,t=1}^{N} \gamma^{t-1} r(\boldsymbol{s}_{t,n},\boldsymbol{a}_{t,n})$ obtained by collecting N trajectories using $\pi_{\boldsymbol{\theta}}$. The sample return is an unbiased estimator of the action-value function. However, it often has high variance which leads to slow convergence.

An alternative approach is to estimate the action-value function by a *critic* denoted by $\widehat{Q}(\boldsymbol{s},\boldsymbol{a})$ whose parameter is learned such that $\widehat{Q}(\boldsymbol{s},\boldsymbol{a}) \approx Q^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{s},\boldsymbol{a})$. By replacing the action-value function in Eq.(3) with the critic, we obtain the following optimization problem:

$$\boldsymbol{\theta}^{\star} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{s}),\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})} \left[\widehat{Q}(\boldsymbol{s},\boldsymbol{a}) \right]. \tag{5}$$

The *actor-critic* method (Sutton et al., 1999) solves this optimization problem by gradient ascent:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \mathbb{E}_{p(\boldsymbol{s}),\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) \widehat{Q}(\boldsymbol{s},\boldsymbol{a}) \right]. \tag{6}$$

The gradient in Eq.(6) often has less variance than that in Eq.(4), which leads to faster convergence[2]. A large class of actor-critic methods is based on this method (Peters & Schaal, 2008; Mnih et al., 2016). As shown in these papers, these methods only use the value of the critic to learn the actor.

The *deterministic policy gradients* (DPG) method (Silver et al., 2014) is an actor-critic method that uses the first-order information of the critic. DPG updates a deterministic actor $\pi_{\boldsymbol{\theta}}(\boldsymbol{s})$ by

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \mathbb{E}_{p(\boldsymbol{s})} \left[\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\boldsymbol{s}) \nabla_{\boldsymbol{a}} \widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\pi_{\boldsymbol{\theta}}(\boldsymbol{s})} \right]. \tag{7}$$

A method related to DPG is the *stochastic value gradients* (SVG) method (Heess et al., 2015) that is able to learn a stochastic policy but it requires learning a model of the transition function.

The actor-critic methods described above only use up to the first-order information of the critic when learning the actor and ignore higher-order ones. Below, we discuss existing approaches that utilize the second-order information by applying second-order optimization methods to solve Eq.(5).

2.2 Second-order Methods for Policy Learning

The actor-critic methods described above are *first-order methods* which update the optimization variables based on the gradient of the objective function. First-order methods are popular in deep learning thanks to their computational efficiency. However, it is known in machine learning that *second-order methods* often lead to faster learning because they use the curvature information to compute a better update direction, i.e., the steepest ascent direction along the curvature[3].

The main idea of second-order methods is to rotate the gradient by the inverse of a *curvature matrix*. For instance, second-order updates for the actor-critic method in Eq.(6) are in the following form:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{G}^{-1} \left\{ \mathbb{E}_{p(\boldsymbol{s}),\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) \widehat{Q}(\boldsymbol{s},\boldsymbol{a}) \right] \right\}, \tag{8}$$

where $\boldsymbol{G} \in \mathbb{R}^{d_{\boldsymbol{\theta}} \times d_{\boldsymbol{\theta}}}$ is a curvature matrix. The behavior of second-order methods depend on the definition of a curvature matrix. The most well-known second-order method is the *Newton method* where its curvature matrix is the Hessian of the objective function w.r.t. the optimization variables:

$$\boldsymbol{G}_{\text{Hessian}} = \mathbb{E}_{p(\boldsymbol{s}),\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})} \left[\left(\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) \right)^{\top} + \nabla^2_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) \right) \widehat{Q}(\boldsymbol{s},\boldsymbol{a}) \right]. \tag{9}$$

The *natural gradient method* is another well-known second-order method which uses the *Fisher information matrix* (FIM) as the curvature matrix (Amari, 1998):

$$\boldsymbol{G}_{\text{FIM}} = \mathbb{E}_{p(\boldsymbol{s}),\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})^{\top} \right]. \tag{10}$$

[2]This gradient is a biased estimator of the policy gradient in Eq.(4). However, it is unbiased under some regularity conditions (Sutton et al., 1999).

[3]We use the term "second-order methods" in a broad sense here, including quasi-Newton and natural gradient methods which approximate the curvature matrix by the first-order information.

Unlike the Hessian matrix, FIM provides information about changes of the policy measured by an approximated KL divergence: $\mathbb{E}_{p(\boldsymbol{s})}\left[\mathrm{KL}(\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})||\pi_{\boldsymbol{\theta}'}(\boldsymbol{a}|\boldsymbol{s}))\right] \approx (\boldsymbol{\theta}-\boldsymbol{\theta}')^\top \boldsymbol{G}_{\mathrm{FIM}}(\boldsymbol{\theta}-\boldsymbol{\theta}')$ (Kakade, 2001). We can see that $\boldsymbol{G}_{\mathrm{Hessian}}$ and $\boldsymbol{G}_{\mathrm{FIM}}$ are very similar but the former also contains the critic and the Hessian of the actor while the latter does not. This suggests that the Hessian provides more information than that in FIM. However, FIM is always positive semi-definite while the Hessian may be indefinite. See Furmston et al. (2016) for detailed comparisons between the two curvature matrices in policy search[4]. Nonetheless, actor-critic methods based on natural gradient were shown to be very efficient (Peters & Schaal, 2008; Schulman et al., 2015a).

We are not aware of existing work that considers second-order updates for DPG or SVG. However, their second-order updates can be trivially derived. For example, a Newton update for DPG is

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{D}^{-1}\left\{\mathbb{E}_{p(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\pi_{\boldsymbol{\theta}}(\boldsymbol{s})}\right]\right\}, \tag{11}$$

where the (i,j)-th entry of the Hessian matrix $\boldsymbol{D} \in \mathbb{R}^{d_{\boldsymbol{\theta}}\times d_{\boldsymbol{\theta}}}$ is

$$D_{ij} = \mathbb{E}_{p(\boldsymbol{s})}\left[\frac{\partial \pi_{\boldsymbol{\theta}}(\boldsymbol{s})}{\partial \theta_i}^\top \nabla^2_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\pi_{\boldsymbol{\theta}}(\boldsymbol{s})}\frac{\partial \pi_{\boldsymbol{\theta}}(\boldsymbol{s})}{\partial \theta_j} + \frac{\partial^2 \pi_{\boldsymbol{\theta}}(\boldsymbol{s})}{\partial \theta_i \partial \theta_j}^\top \nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\pi_{\boldsymbol{\theta}}(\boldsymbol{s})}\right]. \tag{12}$$

Note that $\partial\pi_{\boldsymbol{\theta}}(\boldsymbol{s})/\partial\theta$ and $\partial^2\pi_{\boldsymbol{\theta}}(\boldsymbol{s})/\partial\theta\partial\theta'$ are vectors since $\pi_{\boldsymbol{\theta}}(\boldsymbol{s})$ is a vector-valued function. Interestingly, the Hessian of DPG contains the Hessians of the actor and the critic. In contrast, the Hessian of the actor-critic method contains the Hessian of the actor and the value of the critic.

Second-order methods are appealing in reinforcement learning because they have high data efficiency. However, inverting the curvature matrix (or solving a linear system) requires cubic computational complexity in terms of the number of optimization variables. For this reason, the second-order updates in Eq.(8) and Eq.(11) are impractical in deep reinforcement learning due to a large number of weight parameters in deep neural networks. In such a scenario, an approximation of the curvature matrix is required to reduce the computational burden. For instance, Furmston et al. (2016) proposed to use only diagonal entries of an approximated Hessian matrix. However, this approximation clearly leads to a loss of useful curvature information since the gradient is scaled but not rotated. More recently, Wu et al. (2017) proposed a natural actor-critic method that approximates block-diagonal entries of FIM. However, this approximation corresponds to ignoring useful correlations between weight parameters in different layers of neural networks.

3 Guide Actor-Critic

In this section, we propose the guide actor-critic (GAC) method that performs second-order updates without the previously discussed computational issue. Unlike existing methods that directly learn the parameterized actor from the critic, GAC separates the problem of learning the parameterized actor into problems of 1) learning a *guide actor* that locally maximizes the critic, and 2) learning a parameterized actor based on the guide actor. This separation allows us to perform a second-order update for the guide actor where the dimensionality of the curvature matrix is independent of the parameterization of the actor.

We formulate an optimization problem for learning the guide actor in Section 3.1 and present its solution in Section 3.2. Then in Section 3.3 and Section 3.4, we show that the solution corresponds to performing second-order updates. Finally, Section 3.5 presents the learning step for the parameterized actor using supervised learning. The pseudo-code of our method is provided in Appendix B.

3.1 Optimization Problem for Guide Actor

Our first goal is to learn a guide actor that maximizes the critic. However, greedy maximization should be avoided since the critic is a noisy estimate of the expected return and a greedy actor may change too abruptly across learning iterations. Such a behavior is undesirable in real-world problems, especially in robotics (Deisenroth et al., 2013). Instead, we maximize the critic with

[4] Furmston et al. (2016) proposed an approximate Newton method for policy search. Their policy search method was shown to perform better than methods based on gradient ascent and natural gradient ascent.

additional constraints:

$$\begin{aligned} \max_{\tilde{\pi}} \quad & \mathbb{E}_{p^\beta(\boldsymbol{s}),\tilde{\pi}(\boldsymbol{a}|\boldsymbol{s})}\left[\widehat{Q}(\boldsymbol{s},\boldsymbol{a})\right], \\ \text{subject to} \quad & \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\mathrm{KL}(\tilde{\pi}(\boldsymbol{a}|\boldsymbol{s})||\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}))\right] \leqslant \epsilon, \\ & \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\mathrm{H}(\tilde{\pi}(\boldsymbol{a}|\boldsymbol{s}))\right] \geqslant \kappa, \\ & \mathbb{E}_{p^\beta(\boldsymbol{s})\tilde{\pi}(\boldsymbol{a}|\boldsymbol{s})} = 1, \end{aligned} \tag{13}$$

where $\tilde{\pi}(\boldsymbol{a}|\boldsymbol{s})$ is the guide actor to be learned, $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$ is the current parameterized actor that we want to improve upon. The state distribution $p^\beta(\boldsymbol{s})$ is induced by past trajectories. The objective function differs from the ones in Eq.(3) in two important aspects. First, we maximize for a policy function $\tilde{\pi}$ and not for the policy parameter. This is more advantageous than optimizing for a policy parameter since the policy function can be obtained in a closed form, as will be shown in the next subsection. Second, the expectation is defined over a state distribution from past trajectories and this gives us off-policy methods with higher data efficiency. The first constraint is the Kullback-Leibler (KL) divergence constraint where $\mathrm{KL}(p(\boldsymbol{x})||q(\boldsymbol{x})) = \mathbb{E}_{p(\boldsymbol{x})}\left[\log p(\boldsymbol{x}) - \log q(\boldsymbol{x})\right]$. The second constraint is the Shannon entropy constraint where $\mathrm{H}(p(\boldsymbol{x})) = -\mathbb{E}_{p(\boldsymbol{x})}\left[\log p(\boldsymbol{x})\right]$. The KL constraint is commonly used in reinforcement learning to prevent unstable behavior due to excessively greedy update (Peters & Schaal, 2008; Peters et al., 2010; Levine & Koltun, 2013; Abdolmaleki et al., 2015; Schulman et al., 2015a). The entropy constraint is crucial for maintaining stochastic behavior and preventing premature convergence (Abdolmaleki et al., 2015; Mnih et al., 2016; Haarnoja et al., 2017). The final constraint ensures that the guide actor is a proper probability distribution. The KL bound $\epsilon > 0$ and the entropy bound κ are hyper-parameters which control the exploration-exploitation trade-off of the method. In practice, we fix the value of ϵ and adaptively reduce the value of κ based on the current actor's entropy, as suggested by Abdolmaleki et al. (2015). More details of these tuning parameters are given in Appendix C.

This optimization problem can be solved by the method of Lagrange multipliers. The solution is

$$\tilde{\pi}(\boldsymbol{a}|\boldsymbol{s}) \propto \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})^{\frac{\eta^\star}{\eta^\star+\omega^\star}} \exp\left(\frac{\widehat{Q}(\boldsymbol{s},\boldsymbol{a})}{\eta^\star+\omega^\star}\right), \tag{14}$$

where $\eta^\star > 0$ and $\omega^\star > 0$ are dual variables corresponding to the KL and entropy constraints, respectively. The dual variable corresponding to the probability distribution constraint is contained in the normalization term and is determined by $\eta^\star$ and $\omega^\star$. These dual variables are obtained by minimizing the dual function:

$$g(\eta,\omega) = \eta\epsilon - \omega\kappa + (\eta+\omega)\mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\log \int \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})^{\frac{\eta}{\eta+\omega}} \exp\left(\frac{\widehat{Q}(\boldsymbol{s},\boldsymbol{a})}{\eta+\omega}\right) \mathrm{d}\boldsymbol{a}\right]. \tag{15}$$

All derivations and proofs are given in Appendix A. The solution in Eq.(14) tells us that the guide actor is obtained by weighting the current actor with $\widehat{Q}(\boldsymbol{s},\boldsymbol{a})$. If we set $\epsilon \to 0$ then we have $\tilde{\pi} \approx \pi_{\boldsymbol{\theta}}$ and the actor is not updated. On the other hand, if we set $\epsilon \to \infty$ then we have $\tilde{\pi}(\boldsymbol{a}|\boldsymbol{s}) \propto \exp(\widehat{Q}(\boldsymbol{s},\boldsymbol{a})/\omega^\star)$, which is a softmax policy where $\omega^\star$ is the temperature parameter.

3.2 Learning Guide Actor

Computing $\tilde{\pi}(\boldsymbol{a}|\boldsymbol{s})$ and evaluating $g(\eta,\omega)$ are intractable for an arbitrary $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$. We overcome this issue by imposing two assumptions. First, we assume that the actor is the Gaussian distribution:

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{a}|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{s})), \tag{16}$$

where the mean $\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{s})$ are functions parameterized by a policy parameter $\boldsymbol{\theta}$. Second, we assume that Taylor's approximation of $\widehat{Q}(\boldsymbol{s},\boldsymbol{a})$ is locally accurate up to the second-order. More concretely, the second-order Taylor's approximation using an arbitrary action $\boldsymbol{a}_0$ is given by

$$\widehat{Q}(\boldsymbol{s},\boldsymbol{a}) \approx \widehat{Q}(\boldsymbol{s},\boldsymbol{a}_0) + (\boldsymbol{a}-\boldsymbol{a}_0)^\top \boldsymbol{g}_0(\boldsymbol{s}) + \frac{1}{2}(\boldsymbol{a}-\boldsymbol{a}_0)^\top \boldsymbol{H}_0(\boldsymbol{s})(\boldsymbol{a}-\boldsymbol{a}_0) + \mathcal{O}(\|\boldsymbol{a}\|^3), \tag{17}$$

where $\boldsymbol{g}_0(\boldsymbol{s}) = \nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{a}_0}$ and $\boldsymbol{H}_0(\boldsymbol{s}) = \nabla^2_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{a}_0}$ are the gradient and Hessian of the critic w.r.t. $\boldsymbol{a}$ evaluated at $\boldsymbol{a}_0$, respectively. By assuming that the higher order term $\mathcal{O}(\|\boldsymbol{a}\|^3)$ is sufficiently small, we can rewrite Taylor's approximation at $\boldsymbol{a}_0$ as

$$\widehat{Q}_0(\boldsymbol{s},\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^\top \boldsymbol{H}_0(\boldsymbol{s})\boldsymbol{a} + \boldsymbol{a}^\top \boldsymbol{\psi}_0(\boldsymbol{s}) + \xi_0(\boldsymbol{s}), \tag{18}$$

where $\boldsymbol{\psi}_0(\boldsymbol{s}) = \boldsymbol{g}_0(\boldsymbol{s}) - \boldsymbol{H}_0(\boldsymbol{s})\boldsymbol{a}_0$ and $\xi_0(\boldsymbol{s}) = \frac{1}{2}\boldsymbol{a}_0^\top \boldsymbol{H}_0(\boldsymbol{s})\boldsymbol{a}_0 - \boldsymbol{a}_0^\top \boldsymbol{g}_0(\boldsymbol{s}) + \widehat{Q}(\boldsymbol{s},\boldsymbol{a}_0)$. Note that $\boldsymbol{H}_0(\boldsymbol{s})$, $\boldsymbol{\psi}_0(\boldsymbol{s})$, and $\xi_0(\boldsymbol{s})$ depend on the value of $\boldsymbol{a}_0$ and do not depend on the value of $\boldsymbol{a}$. This dependency is explicitly denoted by the subscript.

Substituting the Gaussian distribution and Taylor's approximation into Eq.(14) yields another Gaussian distribution $\tilde{\pi}(\boldsymbol{a}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{a}|\boldsymbol{\phi}_+(\boldsymbol{s}), \boldsymbol{\Sigma}_+(\boldsymbol{s}))$, where the mean and covariance are given by

$$\boldsymbol{\phi}_+(\boldsymbol{s}) = \boldsymbol{F}^{-1}(\boldsymbol{s})\boldsymbol{L}(\boldsymbol{s}), \quad \boldsymbol{\Sigma}_+(\boldsymbol{s}) = (\eta^\star + \omega^\star)\boldsymbol{F}^{-1}(\boldsymbol{s}). \tag{19}$$

The matrix $\boldsymbol{F}(\boldsymbol{s}) \in \mathbb{R}^{d_{\mathbf{a}}\times d_{\mathbf{a}}}$ and vector $\boldsymbol{L}(\boldsymbol{s}) \in \mathbb{R}^{d_{\mathbf{a}}}$ are defined as

$$\boldsymbol{F}(\boldsymbol{s}) = \eta^\star \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s}) - \boldsymbol{H}_0(\boldsymbol{s}), \quad \boldsymbol{L}(\boldsymbol{s}) = \eta^\star \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \boldsymbol{\psi}_0(\boldsymbol{s}). \tag{20}$$

The dual variables $\eta^\star$ and $\omega^\star$ are obtained by minimizing the following dual function:

$$\begin{aligned}\widehat{g}(\eta,\omega) = \eta\epsilon - \omega\kappa + (\eta+\omega)\mathbb{E}_{p^\beta(\boldsymbol{s})}&\left[\log\sqrt{\frac{|2\pi(\eta+\omega)\boldsymbol{F}_\eta^{-1}(\boldsymbol{s})|}{|2\pi\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{s})|^{\frac{\eta}{\eta+\omega}}}}\right] \\ &+ \frac{1}{2}\mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\boldsymbol{L}_\eta(\boldsymbol{s})^\top \boldsymbol{F}_\eta^{-1}(\boldsymbol{s})\boldsymbol{L}_\eta(\boldsymbol{s}) - \eta\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})^\top \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\right] + \text{const},\end{aligned} \tag{21}$$

where $\boldsymbol{F}_\eta(\boldsymbol{s})$ and $\boldsymbol{L}_\eta(\boldsymbol{s})$ are defined similarly to $\boldsymbol{F}(\boldsymbol{s})$ and $\boldsymbol{L}(\boldsymbol{s})$ but with η instead of $\eta^\star$. The term $\xi_0(\boldsymbol{s})$ is contained in the normalization term of the guide actor and in the constant term.

The practical advantage of using the Gaussian distribution and Taylor's approximation is that the guide actor can be obtained in a closed form and the dual function can be evaluated through matrix-vector products. The expectation over $p^\beta(\boldsymbol{s})$ can be approximated by e.g., samples drawn from a replay buffer (Mnih et al., 2015). We require inverting $\boldsymbol{F}_\eta(\boldsymbol{s})$ to evaluate the dual function. However, these matrices are computationally cheap to invert when the dimension of actions is not large.

3.3 Guide Actor Learning as Second-order Optimization

For now we assume that the critic is an accurate estimator of the true action-value function. In this case, the quality of the guide actor depends on the accuracy of sample approximation in $\widehat{g}(\eta,\omega)$ and the accuracy of Taylor's approximation. To make Taylor's approximation locally accurate, we should take the action $\boldsymbol{a}_0$ in the vicinity of $\boldsymbol{a}$. However, we did not directly use any individual $\boldsymbol{a}$ to compute the guide actor, but we weight $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$ by the critic (see Eq.(14)). Thus, to obtain an accurate Taylor's approximation of the critic, the action $\boldsymbol{a}_0$ needs to be similar to actions sampled from $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$. Based on this observation, we propose two approaches to perform Taylor's approximation.

Taylor's approximation around the mean. In this approach, we perform Taylor's approximation using the mean of $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$. More specifically, we use $\boldsymbol{a}_0 = \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})}[\boldsymbol{a}] = \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})$ for Eq.(18). In this case, we can show that the mean update in Eq.(19) corresponds to performing a second-order update in the action space to maximize $\widehat{Q}(\boldsymbol{s},\boldsymbol{a})$:

$$\boldsymbol{\phi}_+(\boldsymbol{s}) = \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \boldsymbol{F}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}^{-1}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})}, \tag{22}$$

where $\boldsymbol{F}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s}) = \eta^\star \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s}) - \boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})$ and $\boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s}) = \nabla^2_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})}$. This equivalence can be shown by substitution and the proof is given in Appendix A.2. This update equation reveals that the guide actor maximizes the critic by taking a step in the action space similarly to the Newton method. However, the main difference lies in the curvature matrix where the Newton method uses Hessians $\boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})$ but GAC uses a *damped Hessian* $\boldsymbol{F}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})$. The damping term $\eta^\star \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s})$ corresponds to the effect of the KL constraint and can be viewed as a trust-region that controls the update step-size. This damping term is particularly important since Taylor's approximation is accurate only locally and we should not take a large step in each update (Nocedal & Wright, 2006).

Expectation of Taylor's approximations. Instead of using $\boldsymbol{a}_0 = \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})$, we may perform Taylor's approximation at samples $\boldsymbol{a}_0 \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$ and compute its expectation. More concretely, we define $\widetilde{Q}(\boldsymbol{s}, \boldsymbol{a})$ to be an expectation of $\widehat{Q}_0(\boldsymbol{s}, \boldsymbol{a})$ over $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})$:

$$\widetilde{Q}(\boldsymbol{s}, \boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^\top \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\boldsymbol{H}_0(\boldsymbol{s})\right]\boldsymbol{a} + \boldsymbol{a}^\top \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\boldsymbol{\psi}_0(\boldsymbol{s})\right] + \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\xi_0(\boldsymbol{s})\right]. \tag{23}$$

Note that $\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{H}_0(\boldsymbol{s})] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\nabla^2_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{a}_0}]$ and the expectation is computed w.r.t. the distribution $\pi_{\boldsymbol{\theta}}$ of $\boldsymbol{a}_0$. We use this notation to avoid confusion even though $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})$ and $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$ are the same distribution. When Eq.(23) is used, the mean update does not directly correspond to any second-order optimization step. However, under an (unrealistic) independence assumption $\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{H}_0(\boldsymbol{s})\boldsymbol{a}_0] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{H}_0(\boldsymbol{s})]\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{a}_0]$, we can show that the mean update corresponds to the following second-order optimization step:

$$\boldsymbol{\phi}_+(\boldsymbol{s}) = \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\boldsymbol{F}_0(\boldsymbol{s})\right]^{-1}\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{a}_0}\right], \tag{24}$$

where $\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\boldsymbol{F}_0(\boldsymbol{s})\right] = \eta^\star\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s}) - \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\boldsymbol{H}_0(\boldsymbol{s})\right]$. Interestingly, the mean is updated using averaged gradients and Hessians. We believe that this can be advantageous for avoiding local optima. Note that $\widetilde{Q}(\boldsymbol{s}, \boldsymbol{a})$ can be used even when the independence assumption does not hold.

In the remainder, we use $\boldsymbol{F}(\boldsymbol{s})$ to denote both of $\boldsymbol{F}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})$ and $\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\boldsymbol{F}_0(\boldsymbol{s})\right]$, and use $\boldsymbol{H}(\boldsymbol{s})$ to denote both of $\boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})$ and $\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{H}_0(\boldsymbol{s})]$. In the experiments, we use GAC-0 to refer to GAC with Taylor's approximation by $\boldsymbol{a}_0 = \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})$, and we use GAC-1 to refer to GAC with Taylor's approximation by $\boldsymbol{a}_0 \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$, where we use a single sample for computational efficiency.

3.4 Gauss-Newton Approximation of Hessian

The covariance update in Eq.(19) indicates that $\boldsymbol{F}(\boldsymbol{s}) = \eta^\star\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s}) - \boldsymbol{H}(\boldsymbol{s})$ needs to be positive definite. The matrix $\boldsymbol{F}(\boldsymbol{s})$ is guaranteed to be positive definite if $\boldsymbol{H}(\boldsymbol{s})$ is negative semi-definite. However, this is not guaranteed in general since $\boldsymbol{H}(\boldsymbol{s})$ can be indefinite. To overcome this issue, we firstly consider the following identity:

$$\boldsymbol{H}(\boldsymbol{s}) = \nabla^2_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s}, \boldsymbol{a}) = -\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})^\top + \nabla^2_{\boldsymbol{a}}\exp(\widehat{Q}(\boldsymbol{s}, \boldsymbol{a}))\exp(-\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})). \tag{25}$$

The proof is given in Appendix A.3. The first term is always negative semi-definite while the second term is indefinite. Therefore, a negative semi-definite approximation of the Hessian can be obtained as

$$\boldsymbol{H}_0(\boldsymbol{s}) \approx -\left[\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})^\top\right]_{\boldsymbol{a}=\boldsymbol{a}_0}. \tag{26}$$

The second term in Eq.(25) is proportional to $\exp(-\widehat{Q}(\boldsymbol{s}, \boldsymbol{a}))$ and it will be small for high values of $\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})$. This implies that the approximation should gets more accurate as the policy approach a local maxima of $\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})$. We call this approximation *Gauss-Newton approximation* since it is similar to the Gauss-Newton approximation for the Newton method (Nocedal & Wright, 2006).

3.5 Learning Parameterized Actor

The second step of GAC is to learn a parameterized actor that well represents the guide actor. Below, we discuss two supervised learning approaches.

3.5.1 Fully-Parameterized Gaussian Policy

Since the guide actor is a Gaussian distribution with a state-dependent mean and covariance, a natural choice for the parameterized actor is again a parameterized Gaussian distribution with a state-dependent mean and covariance: $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{a}|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{s}))$. The parameter $\boldsymbol{\theta}$ can be learned by minimizing the expected KL divergence to the guide actor:

$$\begin{aligned}\mathcal{L}^{\mathrm{KL}}(\boldsymbol{\theta}) &= \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\mathrm{KL}\left(\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})||\tilde{\pi}(\boldsymbol{a}|\boldsymbol{s})\right)\right] \\ &= \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\frac{\mathrm{Tr}(\boldsymbol{F}(\boldsymbol{s})\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{s}))}{\eta^\star + \omega^\star} - \log|\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{s})|\right] + \frac{\mathcal{L}^{\mathrm{W}}(\boldsymbol{\theta})}{\eta^\star + \omega^\star} + \mathrm{const},\end{aligned} \tag{27}$$

where $\mathcal{L}^{\mathrm{W}}(\boldsymbol{\theta}) = \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) - \boldsymbol{\phi}_+(\boldsymbol{s})\|^2_{\boldsymbol{F}(\boldsymbol{s})}\right]$ is the weighted-mean-squared-error (WMSE) which only depends on $\boldsymbol{\theta}$ of the mean function. The const term does not depend on $\boldsymbol{\theta}$.

Minimizing the KL divergence reveals connections between GAC and deterministic policy gradients (DPG) (Silver et al., 2014). By computing the gradient of the WMSE, we can show that

$$\begin{aligned}\frac{\nabla_{\boldsymbol{\theta}}\mathcal{L}^{\mathrm{W}}(\boldsymbol{\theta})}{2} = &-\mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})}\right] + \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_+(\boldsymbol{s})}\right] \\ &+ \eta^\star\mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{\Sigma}^{-1}(\boldsymbol{s})\boldsymbol{H}^{-1}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})}\right] \\ &- \eta^\star\mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{\Sigma}^{-1}(\boldsymbol{s})\boldsymbol{H}^{-1}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_+(\boldsymbol{s})}\right]. \end{aligned} \tag{28}$$

The proof is given in Appendix A.4. The negative of the first term is precisely equivalent to DPG. Thus, updating the mean parameter by minimizing the KL loss with gradient descent can be regarded as updating the mean parameter with *biased* DPG where the bias terms depend on $\eta^\star$. We can verify that $\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_+(\boldsymbol{s})} = 0$ when $\eta^\star = 0$ and this is the case of $\epsilon \to \infty$. Thus, all bias terms vanish when the KL constraint is ignored and the mean update of GAC coincides with DPG. However, unlike DPG which learns a deterministic policy, we can learn both the mean and covariance in GAC.

3.5.2 Gaussian Policy with Parameterized Mean

While a state-dependent parameterized covariance function is flexible, we observe that learning performance is sensitive to the initial parameter of the covariance function. For practical purposes, we propose using a parametrized Gaussian distribution with state-independent covariance: $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{a}|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{\Sigma})$. This class of policies subsumes deterministic policies with additive independent Gaussian noise for exploration. To learn $\boldsymbol{\theta}$, we minimize the mean-squared-error (MSE):

$$\mathcal{L}^{\mathrm{M}}(\boldsymbol{\theta}) = \frac{1}{2}\mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) - \boldsymbol{\phi}_+(\boldsymbol{s})\|_2^2\right]. \tag{29}$$

For the covariance, we use the average of the guide covariances: $\boldsymbol{\Sigma} = (\eta^\star + \omega^\star)\mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\boldsymbol{F}^{-1}(\boldsymbol{s})\right]$. For computational efficiency, we execute a single gradient update in each learning iteration instead of optimizing this loss function until convergence.

Similarly to the above analysis, the gradient of the MSE w.r.t. $\boldsymbol{\theta}$ can be expanded and rewritten into

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}^{\mathrm{M}}(\boldsymbol{\theta}) = \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{H}^{-1}(\boldsymbol{s})\left(\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})} - \nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_+(\boldsymbol{s})}\right)\right]. \tag{30}$$

Again, the mean update of GAC coincides with DPG when we minimize the MSE and set $\eta^\star = 0$ and $\boldsymbol{H}(\boldsymbol{s}) = -\boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix. We can also substitute these values back into Eq.(22). Then, we can interpret DPG as a method that performs first-order optimization in the action space:

$$\boldsymbol{\phi}_+(\boldsymbol{s}) = \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})}, \tag{31}$$

and then uses the gradient in Eq.(30) to update the policy parameter. This interpretation shows that DPG is indeed a first-order method that only uses the first-order information of the critic for the actor learning. Therefore in principle, GAC, which uses the second-order information, should learn faster than DPG.

3.6 Policy Evaluation for Critic

Beside actor learning, the performance of actor-critic methods also depends on the accuracy of the critic. We assume that the critic $\widehat{Q}_{\boldsymbol{\nu}}(\boldsymbol{s},\boldsymbol{a})$ is represented by neural networks with a parameter $\boldsymbol{\nu}$. We adopt the approach proposed by Lillicrap et al. (2015) with some adjustment to learn $\boldsymbol{\nu}$. More concretely, we use gradient descent to minimize the squared Bellman error with a slowly moving target critic:

$$\boldsymbol{\nu} \leftarrow \boldsymbol{\nu} - \alpha\nabla_{\boldsymbol{\nu}}\mathbb{E}_{p^\beta(\boldsymbol{s}),\beta(\boldsymbol{a}|\boldsymbol{s}),p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})}\left[\left(\widehat{Q}_{\boldsymbol{\nu}}(\boldsymbol{s},\boldsymbol{a}) - y\right)^2\right], \tag{32}$$

where $\alpha > 0$ is the step-size. The target value $y = r(\boldsymbol{s},\boldsymbol{a}) + \gamma\mathbb{E}_{\pi(\boldsymbol{a}'|\boldsymbol{s}')}[\widehat{Q}_{\bar{\boldsymbol{\nu}}}(\boldsymbol{s}',\boldsymbol{a}')]$ is computed by target critic $\widehat{Q}_{\bar{\boldsymbol{\nu}}}(\boldsymbol{s}',\boldsymbol{a}')$ which has parameter $\bar{\boldsymbol{\nu}}$ updated by $\bar{\boldsymbol{\nu}} \leftarrow \tau\boldsymbol{\nu} + (1-\tau)\bar{\boldsymbol{\nu}}$ for $0 < \tau < 1$.

As suggested by Lillicrap et al. (2015), the target critic improves the learning stability and we set $\tau = 0.001$ in experiments. The expectation for the squared error is approximated using mini-batch samples $\{(\boldsymbol{s}_n, \boldsymbol{a}_n, r_n, \boldsymbol{s}'_n)\}_{n=1}^N$ drawn from a replay buffer. The expectation over $\pi(\boldsymbol{a}'|\boldsymbol{s}')$ is approximated using samples $\{\boldsymbol{a}'_{n,m}\}_{m=1}^M \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{a}'|\boldsymbol{s}'_n)$ for each $\boldsymbol{s}'_n$. We do not use a target actor to compute y since the KL upper-bound already constrains the actor update and a target actor will further slow it down. Note that we are not restricted to this evaluation method and more efficient methods such as *Retrace* (Munos et al., 2016) can also be used.

GAC requires computing $\nabla_{\boldsymbol{a}} \widehat{Q}_{\boldsymbol{\nu}}(\boldsymbol{s}, \boldsymbol{a})$ and its outer product for the Gauss-Newton approximation. The computational complexity of the outer product operation is $\mathcal{O}(d_{\mathbf{a}}^2)$ and is inexpensive when compared to the dimension of $\boldsymbol{\nu}$. For a linear-in-parameter model $\widehat{Q}_{\boldsymbol{\nu}}(\boldsymbol{s}, \boldsymbol{a}) = \boldsymbol{\nu}^\top \boldsymbol{\mu}(\boldsymbol{s}, \boldsymbol{a})$, the gradient can be efficiently computed for common choices of the basis function $\boldsymbol{\mu}$ such as the Gaussian function. For deep neural network models, the gradient can be computed by the automatic-differentiation (Goodfellow et al., 2016) where its cost depends on the network architecture.

4 Related Work

Besides the connections to DPG, GAC also relates to existing methods as follows. A similar optimization problem to Eq.(13) was considered by the *model-free trajectory optimization* (MOTO) method (Akrour et al., 2016). GAC can be viewed as an extension of MOTO with two significant novelties. First, MOTO learns a sequence of time-dependent log-linear Gaussian policies $\pi_t(\boldsymbol{a}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{a}|\boldsymbol{B}_t\boldsymbol{s} + \boldsymbol{b}_t, \boldsymbol{\Sigma}_t)$, while GAC learns a log-nonlinear Gaussian policy. Second, MOTO learns a time-dependent critic given by $\widehat{Q}_t(\boldsymbol{s}, \boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^\top \boldsymbol{C}_t \boldsymbol{a} + \boldsymbol{a}^\top \boldsymbol{D}_t \boldsymbol{s} + \boldsymbol{a}^\top \boldsymbol{c}_t + \xi_t(\boldsymbol{s})$ and performs policy update with these functions. In contrast, our method learns a more complex critic and performs Taylor's approximation in each training step.

Besides MOTO, the optimization problem also resembles that of *trust region policy optimization* (TRPO) (Schulman et al., 2015a). TRPO solves the following optimization problem:

$$\max_{\boldsymbol{\theta}'} \mathbb{E}_{p^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{s}), \pi_{\boldsymbol{\theta}'}(\boldsymbol{a}|\boldsymbol{s})} \left[\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})\right] \text{ subject to } \mathbb{E}_{p^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{s})} \left[\mathrm{KL}(\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) || \pi_{\boldsymbol{\theta}'}(\boldsymbol{a}|\boldsymbol{s}))\right] \leqslant \epsilon, \tag{33}$$

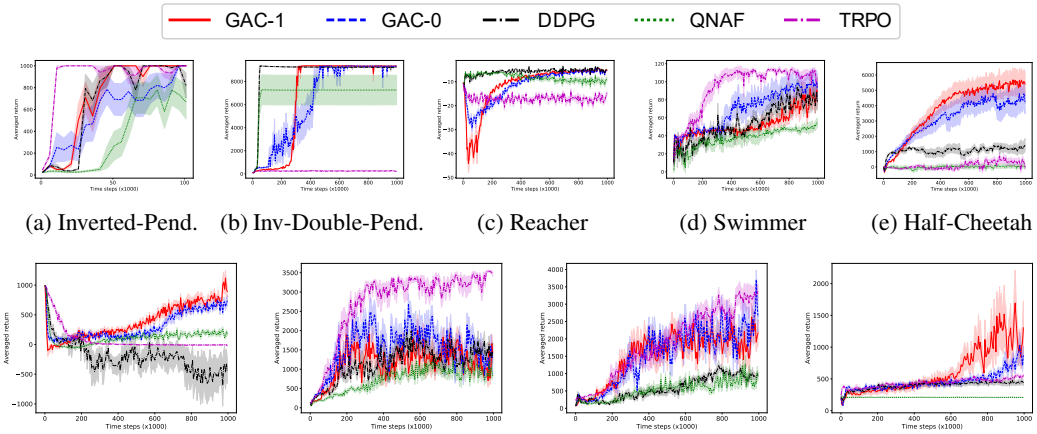
where $\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})$ may be replaced by an estimate of the advantage function (Schulman et al., 2015b). There are two major differences between the two problems. First, TRPO optimizes the policy parameter while we optimize the guide actor. Second, TRPO solves the optimization problem by conjugate gradient where the KL divergence is approximated by the Fisher information matrix, while we solve the optimization problem in a closed form with a quadratic approximation of the critic.

The idea of firstly learning a non-parameterized policy and then later learning a parameterized policy by supervised learning was considered previously in *guided policy search* (GPS) (Levine & Koltun, 2013). However, GPS learns the guide policy by trajectory optimization methods such as an iterative linear-quadratic Gaussian regulator (Li & Todorov, 2004) which require a model of the transition function. In contrast, we learn the guide policy via the critic without learning the transition function.

5 Experimental Results

We evaluate GAC on the OpenAI gym platform (Brockman et al., 2016) with the Mujoco Physics simulator (Todorov et al., 2012). Both the actor and critic use neural networks with two hidden layers of 300 and 400 units, as described in Appendix C. We use the provided state space from the platform and do not use the pixel state space. However, an extension to the pixel state space is straightforward by using convolutional neural networks (Goodfellow et al., 2016). We compare GAC-0 and GAC-1 against deep DPG (DDPG) (Lillicrap et al., 2015), Q-learning with a normalized advantage function (Q-NAF) (Gu et al., 2016), and TRPO (Schulman et al., 2015a;b). Figure 1 shows the learning performance on 9 continuous control tasks. Overall, both GAC-0 and GAC-1 perform comparably with existing methods and they clearly outperform the other methods in Half-Cheetah.

The performance of GAC-0 and GAC-1 is comparable on these tasks, except on Humanoid where GAC-1 learns faster. We expect GAC-0 to be more stable and reliable but easier to get stuck at local optima. On the other hand, the randomness introduced by GAC-1 leads to high variance approximation but this could help escape poor local optima. We conjecture GAC-S that uses $S > 1$



(a) Inverted-Pend. (b) Inv-Double-Pend. (c) Reacher (d) Swimmer (e) Half-Cheetah

(f) Ant (g) Hopper (h) Walker2D (i) Humanoid

Figure 1: Expected returns averaged over 10 trials. The x-axis indicates training time steps. The y-axis indicates averaged return and higher is better. More clear figures are provided in Appendix C.

samples for the averaged Taylor's approximation should outperform both GAC-0 and GAC-1. While this is computationally expensive, we can use parallel computation to reduce the computation time.

The expected returns of both GAC-0 and GAC-1 have high fluctuations on the Hopper and Walker2D tasks when compared to TRPO as can be seen in Figure 1g and Figure 1h. We observe that they can learn good policies for these tasks in the middle of learning. However, the policies quickly diverge to poor ones and then they are quickly improved to be good policies again. We believe that this happens because the step size $\boldsymbol{F}^{-1}(\boldsymbol{s}) = \left(\eta^\star \boldsymbol{\Sigma}^{-1} - \boldsymbol{H}(\boldsymbol{s})\right)^{-1}$ of the guide actor in Eq. (22) can be very large near local optima for Gauss-Newton approximation. That is, the gradients near local optima have small magnitude and this makes the approximation $\boldsymbol{H}(\boldsymbol{s}) = \nabla_{\boldsymbol{a}} \widehat{Q}(\boldsymbol{s}, \boldsymbol{a}) \nabla_{\boldsymbol{a}} \widehat{Q}(\boldsymbol{s}, \boldsymbol{a})^\top$ small as well. If $\eta^\star \boldsymbol{\Sigma}^{-1}$ is also relatively small then the matrix $\boldsymbol{F}^{-1}(\boldsymbol{s})$ can be very large. Thus, under these conditions, GAC may use too large step sizes to compute the guide actor and this results in high fluctuations in performance. We expect that this scenario can be avoided by adding a regularization constant to the Gauss-Newton approximation.

Table 1 in Appendix C shows the wall-clock computation time. DDPG is computationally the most efficient method on all tasks. GAC has low computation costs on tasks with low dimensional actions and its cost increases as the dimensionality of action increases. This high computation cost is due to the dual optimization for finding the step-size parameters η and ω. We believe that the computation cost of GAC can be significantly reduced by letting η and ω be external tuning parameters.

6 Conclusion and Future Work

Actor-critic methods are appealing for real-world problems due to their good data efficiency and learning speed. However, existing actor-critic methods do not use second-order information of the critic. In this paper, we established a novel framework that distinguishes itself from existing work by utilizing Hessians of the critic for actor learning. Within this framework, we proposed a practical method that uses Gauss-Newton approximation instead of the Hessians. We showed through experiments that our method is promising and thus the framework should be further investigated.

Our analysis showed that the proposed method is closely related to deterministic policy gradients (DPG). However, DPG was also shown to be a limiting case of the stochastic policy gradients when the policy variance approaches zero (Silver et al., 2014). It is currently unknown whether our framework has a connection to the stochastic policy gradients as well, and finding such a connection is our future work.

Our main goal in this paper was to provide a new actor-critic framework and we do not claim that our method achieves the state-of-the-art performance. However, its performance can still be improved in

many directions. For instance, we may impose a KL constraint for a parameterized actor to improve its stability, similarly to TRPO (Schulman et al., 2015a). We can also apply more efficient policy evaluation methods such as Retrace (Munos et al., 2016) to achieve better critic learning.

REFERENCES

Abbas Abdolmaleki, Rudolf Lioutikov, Jan Peters, Nuno Lau, Luís Paulo Reis, and Gerhard Neumann. Model-based relative entropy stochastic search. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 3537–3545, 2015.

Riad Akrour, Gerhard Neumann, Hany Abdulsamad, and Abbas Abdolmaleki. Model-free trajectory optimization for reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 2961–2970, 2016.

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 146–155, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

Thomas Furmston, Guy Lever, and David Barber. Approximate newton methods for policy search in markov decision processes. *Journal of Machine Learning Research*, 17(227):1–51, 2016. URL `http://jmlr.org/papers/v17/15-414.html`.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Shixiang Gu, Timothy P. Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 2829–2838, 2016.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1352–1361, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Nicolas Heess, Gregory Wayne, David Silver, Timothy P. Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 2944–2952, 2015.

Sham 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]*, pp. 1531–1538, 2001.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Vijay R. Konda and John N. Tsitsiklis. On actor-critic algorithms. *SIAM J. Control Optim.*, 42(4): 1143–1166, April 2003. ISSN 0363-0129.

Sergey Levine and Vladlen Koltun. Guided policy search. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pp. 1–9, 2013.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17:39:1–39:40, 2016.

Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO 2004, Proceedings of the First International Conference on Informatics in Control, Automation and Robotics, Setúbal, Portugal, August 25-28, 2004*, pp. 222–229, 2004.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1046–1054, 2016.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization, second edition*. World Scientific, 2006.

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

Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.

John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 1889–1897. JMLR.org, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438, 2015b.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 387–395, 2014.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pp. 1057–1063, 1999.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pp. 5026–5033, 2012.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992. doi: 10.1007/BF00992696.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5285–5294. Curran Associates, Inc., 2017.

A DERIVATIONS AND PROOFS

A.1 DERIVATION OF SOLUTION AND DUAL FUNCTION OF GUIDE ACTOR

The solution of the optimization problem:

$$\begin{aligned} \max_{\widetilde{\pi}} \quad & \mathbb{E}_{p^\beta(\boldsymbol{s}),\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s})}\left[\widehat{Q}(\boldsymbol{s},\boldsymbol{a})\right], \\ \text{subject to} \quad & \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\mathrm{KL}(\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s})||\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}))\right] \leqslant \epsilon, \\ & \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\mathrm{H}(\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s}))\right] \geqslant \kappa, \\ & \mathbb{E}_{p^\beta(\boldsymbol{s})\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s})} = 1, \end{aligned} \tag{34}$$

can be obtained by the method of Lagrange multipliers. The derivation here follows the derivation of similar optimization problems by Peters et al. (2010) and Abdolmaleki et al. (2015). The Lagrangian of this optimization problem is

$$\begin{aligned} \mathcal{L}(\widetilde{\pi},\eta,\omega,\nu) = \mathbb{E}_{p^\beta(\boldsymbol{s}),\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s})}\left[\widehat{Q}(\boldsymbol{s},\boldsymbol{a})\right] + \eta(\epsilon - \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\mathrm{KL}(\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s})||\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}))\right]) \\ + \omega(\mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\mathrm{H}(\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s}))\right] - \kappa) + \nu(\mathbb{E}_{p^\beta(\boldsymbol{s})\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s})} - 1), \end{aligned} \tag{35}$$

where η, ω, and ν are the dual variables. Then, by taking derivative of $\mathcal{L}$ w.r.t. $\widetilde{\pi}$ we obtain

$$\partial_{\widetilde{\pi}}\mathcal{L} = \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\int\left(\widehat{Q}(\boldsymbol{s},\boldsymbol{a}) - (\eta+\omega)\log\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s}) + \eta\log\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})\right)\mathrm{d}\boldsymbol{a}\right] - (\eta+\omega-\nu). \tag{36}$$

We set this derivation to zero in order to obtain

$$\begin{aligned} 0 &= \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\int\left(\widehat{Q}(\boldsymbol{s},\boldsymbol{a}) - (\eta+\omega)\log\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s}) + \eta\log\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})\right)\mathrm{d}\boldsymbol{a}\right] - (\eta+\omega-\nu) \\ &= \widehat{Q}(\boldsymbol{s},\boldsymbol{a}) - (\eta+\omega)\log\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s}) + \eta\log\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) - (\eta+\omega-\nu). \end{aligned} \tag{37}$$

Then the solution is given by

$$\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s}) = \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})^{\frac{\eta}{\eta+\omega}}\exp\left(\frac{\widehat{Q}(\boldsymbol{s},\boldsymbol{a})}{\eta+\omega}\right)\exp\left(-\frac{\eta+\omega-\nu}{\eta+\omega}\right) \tag{38}$$

$$\propto \quad \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})^{\frac{\eta}{\eta+\omega}}\exp\left(\frac{\widehat{Q}(\boldsymbol{s},\boldsymbol{a})}{\eta+\omega}\right). \tag{39}$$

To obtain the dual function $g(\eta,\omega)$, we substitute the solution to the constraint terms of the Lagrangian and this gives us

$$\begin{aligned} \mathcal{L}(\eta,\omega,\nu) = \mathbb{E}_{p^\beta(\boldsymbol{s}),\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s})}\left[\widehat{Q}(\boldsymbol{s},\boldsymbol{a})\right] \\ - (\eta+\omega)\mathbb{E}_{p^\beta(\boldsymbol{s}),\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s})}\left[\frac{\widehat{Q}(\boldsymbol{s},\boldsymbol{a})}{\eta+\omega} + \frac{\eta}{\eta+\omega}\log\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) - \frac{\eta+\omega-\nu}{\eta+\omega}\right] \\ + \eta\mathbb{E}_{p^\beta(\boldsymbol{s}),\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s})}\left[\log\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})\right] + \nu\left(\mathbb{E}_{p^\beta,\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s})} - 1\right) + \eta\epsilon - \omega\kappa. \end{aligned} \tag{40}$$

After some calculation, we obtain

$$\begin{aligned} \mathcal{L}(\eta,\omega,\nu) &= \eta\epsilon - \omega\kappa + \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\eta+\omega-\nu\right] \\ &= \eta\epsilon - \omega\kappa + (\eta+\omega)\mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\int\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})^{\frac{\eta}{\eta+\omega}}\exp\left(\frac{\widehat{Q}(\boldsymbol{s},\boldsymbol{a})}{\eta+\omega}\right)\mathrm{d}\boldsymbol{a}\right] \\ &= g(\eta,\omega), \end{aligned} \tag{41}$$

where in the second line we use the fact that $\exp(-\frac{\eta+\omega-\nu}{\eta+\omega})$ is the normalization term of $\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s})$.

A.2 PROOF OF SECOND-ORDER OPTIMIZATION IN ACTION SPACE

Firstly, we show that GAC performs second-order optimization in the action space when Taylor's approximation is performed with $\boldsymbol{a}_0 = \mathbb{E}_{\pi(\boldsymbol{a}|\boldsymbol{s})}[\boldsymbol{a}] = \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})$. Recall that Taylor's approximation with $\boldsymbol{\phi}_{\boldsymbol{\theta}}$ is given by

$$\widehat{Q}(\boldsymbol{s},\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^\top \boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})\boldsymbol{a} + \boldsymbol{a}^\top \boldsymbol{\psi}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s}) + \xi_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s}), \tag{42}$$

where $\boldsymbol{\psi}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s}) = \nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})} - \boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})$. By substituting $\boldsymbol{\psi}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})$ into $\boldsymbol{L}(\boldsymbol{s}) = \eta^\star \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) - \boldsymbol{\psi}_{\boldsymbol{\theta}}(\boldsymbol{s})$, we obtain

$$\begin{aligned}
\boldsymbol{L}(\boldsymbol{s}) &= \eta^\star \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})} - \boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) \\
&= (\eta^\star \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s}) - \boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s}))\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})} \\
&= \boldsymbol{F}(\boldsymbol{s})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})}.
\end{aligned} \tag{43}$$

Therefore, the mean update is equivalent to

$$\begin{aligned}
\boldsymbol{\phi}_+(\boldsymbol{s}) &= \boldsymbol{F}^{-1}(\boldsymbol{s})\boldsymbol{L}(\boldsymbol{s}) \\
&= \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \boldsymbol{F}^{-1}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})},
\end{aligned} \tag{44}$$

which is a second-order optimization step with a curvature matrix $\boldsymbol{F}(\boldsymbol{s}) = \eta^\star \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s}) - \boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})$.

For the case when $\{\boldsymbol{a}_0\} \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{a}_0|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{\Sigma}(\boldsymbol{s}))$ are used for averaged Taylor's approximation, we obtain

$$\boldsymbol{L}(\boldsymbol{s}) = \eta^\star \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{a}_0}\right] - \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\boldsymbol{H}_0(\boldsymbol{s})\boldsymbol{a}_0(\boldsymbol{s})\right]. \tag{45}$$

Then, by assuming that $\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{H}_0(\boldsymbol{s})\boldsymbol{a}_0(\boldsymbol{s})] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{H}_0]\,\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{a}_0]$, we obtain

$$\begin{aligned}
\boldsymbol{L}(\boldsymbol{s}) &= \eta^\star \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{a}_0}\right] - \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{H}_0]\,\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{a}_0] \\
&= \eta^\star \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{a}_0}\right] - \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{H}_0]\,\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) \\
&= (\eta^\star \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s}) - \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{H}_0(\boldsymbol{s})])\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{a}_0}\right] \\
&= \boldsymbol{F}(\boldsymbol{s})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{a}_0}\right].
\end{aligned} \tag{46}$$

Therefore, we have a second-order optimization step

$$\boldsymbol{\phi}_+(\boldsymbol{s}) = \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) + \boldsymbol{F}^{-1}(\boldsymbol{s})\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}\left[\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{a}_0}\right], \tag{47}$$

where $\boldsymbol{F}^{-1}(\boldsymbol{s}) = \eta^\star \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s}) - \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{H}_0(\boldsymbol{s})]$ is a curvature matrix. As described in the main paper, this interpretation is only valid when the equality $\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{H}_0(\boldsymbol{s})\boldsymbol{a}_0(\boldsymbol{s})] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{H}_0]\,\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0|\boldsymbol{s})}[\boldsymbol{a}_0]$ holds. While this equality does not hold in general, it holds for GAC-1. Nonetheless, we can still use the expectation of Taylor's approximation to perform policy update regardless of this assumption.

A.3 PROOF OF GAUSS-NEWTON APPROXIMATION

Let $f(\boldsymbol{s},\boldsymbol{a}) = \exp(\widehat{Q}(\boldsymbol{s},\boldsymbol{a}))$, then the Hessian $\boldsymbol{H}(\boldsymbol{s}) = \nabla_{\boldsymbol{a}}^2\widehat{Q}(\boldsymbol{s},\boldsymbol{a})$ can be expressed as

$$\begin{aligned}
\boldsymbol{H}(\boldsymbol{s}) &= \nabla_{\boldsymbol{a}}\left[\nabla_{\boldsymbol{a}}\log f(\boldsymbol{s},\boldsymbol{a})\right] \\
&= \nabla_{\boldsymbol{a}}\left[\nabla_{\boldsymbol{a}} f(\boldsymbol{s},\boldsymbol{a}) f(\boldsymbol{s},\boldsymbol{a})^{-1}\right] \\
&= \nabla_{\boldsymbol{a}} f(\boldsymbol{s},\boldsymbol{a})\left(\nabla_{\boldsymbol{a}} f(\boldsymbol{s},\boldsymbol{a})^{-1}\right)^\top + \nabla_{\boldsymbol{a}}^2 f(\boldsymbol{s},\boldsymbol{a}) f(\boldsymbol{s},\boldsymbol{a})^{-1} \\
&= \nabla_{\boldsymbol{a}} f(\boldsymbol{s},\boldsymbol{a})\nabla_f f(\boldsymbol{s},\boldsymbol{a})^{-1}\left(\nabla_{\boldsymbol{a}} f(\boldsymbol{s},\boldsymbol{a})\right)^\top + \nabla_{\boldsymbol{a}}^2 f(\boldsymbol{s},\boldsymbol{a}) f(\boldsymbol{s},\boldsymbol{a})^{-1} \\
&= -\nabla_{\boldsymbol{a}} f(\boldsymbol{s},\boldsymbol{a}) f(\boldsymbol{s},\boldsymbol{a})^{-2}\left(\nabla_{\boldsymbol{a}} f(\boldsymbol{s},\boldsymbol{a})\right)^\top + \nabla_{\boldsymbol{a}}^2 f(\boldsymbol{s},\boldsymbol{a}) f(\boldsymbol{s},\boldsymbol{a})^{-1} \\
&= -\left(\nabla_{\boldsymbol{a}} f(\boldsymbol{s},\boldsymbol{a}) f(\boldsymbol{s},\boldsymbol{a})^{-1}\right)\left(\nabla_{\boldsymbol{a}} f(\boldsymbol{s},\boldsymbol{a}) f(\boldsymbol{s},\boldsymbol{a})^{-1}\right)^\top + \nabla_{\boldsymbol{a}}^2 f(\boldsymbol{s},\boldsymbol{a}) f(\boldsymbol{s},\boldsymbol{a})^{-1} \\
&= -\nabla_{\boldsymbol{a}}\log f(\boldsymbol{s},\boldsymbol{a})\nabla_{\boldsymbol{a}}\log f(\boldsymbol{s},\boldsymbol{a})^\top + \nabla_{\boldsymbol{a}}^2 f(\boldsymbol{s},\boldsymbol{a}) f(\boldsymbol{s},\boldsymbol{a})^{-1} \\
&= -\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})^\top + \nabla_{\boldsymbol{a}}^2 \exp(\widehat{Q}(\boldsymbol{s},\boldsymbol{a}))\exp(-\widehat{Q}(\boldsymbol{s},\boldsymbol{a})),
\end{aligned} \tag{48}$$

which concludes the proof.

Beside Gauss-Newton approximation, an alternative approach is to impose a special structure on $\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})$ so that Hessians are always negative semi-definite. In literature, there exists two special structures that satisfies this requirement.

Normalized advantage function (NAF) (Gu et al., 2016): NAF represents the critic by a quadratic function with a negative curvature:

$$\widehat{Q}^{\mathrm{NAF}}(\boldsymbol{s}, \boldsymbol{a}) = \frac{1}{2}(\boldsymbol{a} - \boldsymbol{b}(\boldsymbol{s}))^\top \boldsymbol{W}(\boldsymbol{s})(\boldsymbol{a} - \boldsymbol{b}(\boldsymbol{s})) + V(\boldsymbol{s}), \tag{49}$$

where a negative-definite matrix-valued function $\boldsymbol{W}(\boldsymbol{s})$, a vector-valued function $\boldsymbol{b}(\boldsymbol{s})$ and a scalar-valued function $V(\boldsymbol{s})$ are parameterized functions whose their parameters are learned by policy evaluation methods such as Q-learning (Sutton & Barto, 1998). With NAF, negative definite Hessians can be simply obtained as $\boldsymbol{H}(\boldsymbol{s}) = \boldsymbol{W}(\boldsymbol{s})$. However, a significant disadvantage of NAF is that it assumes the action-value function is quadratic regardless of states and this is generally not true for most reward functions. Moreover, the Hessians become action-independent even though the critic is a function of actions.

Input convex neural networks (ICNNs) (Amos et al., 2017): ICNNs are neural networks with special structures which make them convex w.r.t. their inputs. Since Hessians of concave functions are always negative semi-definite, we may use ICNNs to represent a negative critic and directly use its Hessians. However, similarly to NAF, ICNNs implicitly assume that the action-value function is concave w.r.t. actions regardless of states and this is generally not true for most reward functions.

A.4 Gradient of the Loss Functions for Learning Parameterized Actor

We first consider the weight mean-squared-error loss function where the guide actor is $\mathcal{N}(\boldsymbol{a}|\boldsymbol{\phi}_+(\boldsymbol{s}), \boldsymbol{\Sigma}_+(\boldsymbol{s}))$ and the current actor is $\mathcal{N}(\boldsymbol{a}|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{s}))$. Taylor's approximation of $\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})$ at $\boldsymbol{a}_0 = \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})$ is

$$\widehat{Q}(\boldsymbol{s}, \boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^\top \boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})\boldsymbol{a} + \boldsymbol{a}^\top \boldsymbol{\psi}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s}) + \xi_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s}). \tag{50}$$

By assuming that $\boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})$ is strictly negative definite[5], we can take a derivative of this approximation w.r.t. $\boldsymbol{a}$ and set it to zero to obtain $\boldsymbol{a} = \boldsymbol{H}^{-1}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s}, \boldsymbol{a}) - \boldsymbol{H}^{-1}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})\boldsymbol{\psi}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})$. Replacing $\boldsymbol{a}$ by $\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})$ and $\boldsymbol{\phi}_+(\boldsymbol{s})$ yields

$$\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) = \boldsymbol{H}^{-1}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})} - \boldsymbol{H}^{-1}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})\boldsymbol{\psi}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s}), \tag{51}$$

$$\boldsymbol{\phi}_+(\boldsymbol{s}) = \boldsymbol{H}^{-1}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s}, \boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_+(\boldsymbol{s})} - \boldsymbol{H}^{-1}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})\boldsymbol{\psi}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s}). \tag{52}$$

Let the weight mean-squared-error be defined as

$$\mathcal{L}^{\mathrm{W}}(\boldsymbol{\theta}) = \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) - \boldsymbol{\phi}_+(\boldsymbol{s})\|^2_{\boldsymbol{F}(\boldsymbol{s})}\right]. \tag{53}$$

[5]This can be done by subtracting a small positive value to the diagonal entries of Gauss-Newton approximation.

We consider its gradient w.r.t. $\boldsymbol{\theta}$ as follows:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathcal{L}^{\mathrm{W}}(\boldsymbol{\theta}) &= 2\mathbb{E}_{p^{\beta}(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{F}(\boldsymbol{s})\left(\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})-\boldsymbol{\phi}_{+}(\boldsymbol{s})\right)\right] \\
&= 2\mathbb{E}_{p^{\beta}(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})(\eta^{\star}\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s})-\boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s}))\left(\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})-\boldsymbol{\phi}_{+}(\boldsymbol{s})\right)\right] \\
&= 2\mathbb{E}_{p^{\beta}(\boldsymbol{s})}\Big[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})(\eta^{\star}\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s})-\boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})) \\
&\qquad\qquad \times\left(\boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}^{-1}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})}-\boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}^{-1}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{+}(\boldsymbol{s})}\right)\Big] \\
&= 2\eta^{\star}\mathbb{E}_{p^{\beta}(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s})\boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}^{-1}(\boldsymbol{s})\left(\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})}-\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{+}(\boldsymbol{s})}\right)\right] \\
&\quad + 2\mathbb{E}_{p^{\beta}(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\left(\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{+}(\boldsymbol{s})}-\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})}\right)\right] \\
&= -2\mathbb{E}_{p^{\beta}(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})}\right]+2\mathbb{E}_{p^{\beta}(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{+}(\boldsymbol{s})}\right] \\
&\quad + 2\eta^{\star}\mathbb{E}_{p^{\beta}(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{\Sigma}^{-1}(\boldsymbol{s})\boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}^{-1}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})}\right] \\
&\quad - 2\eta^{\star}\mathbb{E}_{p^{\beta}(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{\Sigma}^{-1}(\boldsymbol{s})\boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}^{-1}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{+}(\boldsymbol{s})}\right]. \qquad (54)
\end{aligned}$$

This concludes the proof for the gradient in Eq.(28). Note that we should not directly replace the mean functions in the weight mean-square-error by Eq.(51) and Eq.(52) before expanding the gradient. This is because the analysis would require computing the gradient w.r.t. $\boldsymbol{\theta}$ inside the Hessians and this is not trivial. Moreover, when we perform gradient descent in practice, the mean $\boldsymbol{\phi}_{+}(\boldsymbol{s})$ is considered as a constant w.r.t. $\boldsymbol{\theta}$ similarly to an output function in supervised learning. That is, there are no computation graph connection between $\boldsymbol{\phi}_{+}(\boldsymbol{s})$ and $\boldsymbol{\theta}$.

The gradient for the mean-squared-error can be derived similarly. Let the mean-squared-error be defined as

$$\mathcal{L}^{\mathrm{M}}(\boldsymbol{\theta}) = \frac{1}{2}\mathbb{E}_{p^{\beta}(\boldsymbol{s})}\left[\|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})-\boldsymbol{\phi}_{+}(\boldsymbol{s})\|_2^2\right]. \qquad (55)$$

Its gradient w.r.t. $\boldsymbol{\theta}$ is given by

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathcal{L}^{\mathrm{M}}(\boldsymbol{\theta}) &= \mathbb{E}_{p^{\beta}(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\left(\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})-\boldsymbol{\phi}_{+}(\boldsymbol{s})\right)\right] \\
&= \mathbb{E}_{p^{\beta}(\boldsymbol{s})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{H}^{-1}(\boldsymbol{s})\left(\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})}-\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{+}(\boldsymbol{s})}\right)\right], \qquad (56)
\end{aligned}$$

which concludes the proof for the gradient in Eq.(30).

To show that $\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{+}(\boldsymbol{s})}=0$ when $\eta^{\star}=0$, we directly substitute $\eta^{\star}=0$ into $\boldsymbol{\phi}_{+}(\boldsymbol{s}) = \left(\eta\boldsymbol{\Sigma}^{-1}(\boldsymbol{s})-\boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})\right)^{-1}\left(\eta^{\star}\boldsymbol{\Sigma}^{-1}(\boldsymbol{s})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s})+\boldsymbol{\psi}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})\right)$ and this yield

$$\boldsymbol{\phi}_{+}(\boldsymbol{s}) = -\boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}^{-1}(\boldsymbol{s})\boldsymbol{\psi}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s}). \qquad (57)$$

Since $\boldsymbol{\phi}_{+}(\boldsymbol{s}) = \boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}^{-1}(\boldsymbol{s})\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{+}(\boldsymbol{s})} - \boldsymbol{H}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}^{-1}(\boldsymbol{s})\boldsymbol{\psi}_{\boldsymbol{\phi}_{\boldsymbol{\theta}}}(\boldsymbol{s})$ from Eq.(52) and the Hessians are non-zero, it has to be that $\nabla_{\boldsymbol{a}}\widehat{Q}(\boldsymbol{s},\boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\phi}_{+}(\boldsymbol{s})}=0$. This is intuitive since without the KL constraint, the mean of the guide actor always be at the optima of the second-order Taylor's approximation and the gradients are zero at the optima.

A.5 Relation to Q-learning with Normalized Advantage Function

The *normalized advantage function* (NAF) (Gu et al., 2016) is defined as

$$\widehat{Q}^{\mathrm{NAF}}(\boldsymbol{s},\boldsymbol{a}) = \frac{1}{2}(\boldsymbol{a}-\boldsymbol{b}(\boldsymbol{s}))^{\top}\boldsymbol{W}(\boldsymbol{s})(\boldsymbol{a}-\boldsymbol{b}(\boldsymbol{s}))+V(\boldsymbol{s}), \qquad (58)$$

where $\boldsymbol{W}(\boldsymbol{s})$ is a negative definite matrix. Gu et al. (2016) proposed to perform Q-learning with NAF by using the fact that $\mathrm{argmax}_{\boldsymbol{a}}\,\widehat{Q}^{\mathrm{NAF}}(\boldsymbol{s},\boldsymbol{a})=\boldsymbol{b}(\boldsymbol{s})$ and $\max_{\boldsymbol{a}}\widehat{Q}^{\mathrm{NAF}}(\boldsymbol{s},\boldsymbol{a})=V(\boldsymbol{s})$.

Here, we show that GAC includes the Q-learning method by Gu et al. (2016) as its special case. This can be shown by using NAF as a critic instead of performing Taylor's approximation of the critic.

Firstly, we expand the quadratic term of NAF as follows:

$$\begin{aligned}\widehat{Q}^{\mathrm{NAF}}(\boldsymbol{s},\boldsymbol{a}) &= \frac{1}{2}(\boldsymbol{a}-\boldsymbol{b}(\boldsymbol{s}))^\top \boldsymbol{W}(\boldsymbol{s})(\boldsymbol{a}-\boldsymbol{b}(\boldsymbol{s})) + V(\boldsymbol{s}) \\ &= \frac{1}{2}\boldsymbol{a}^\top \boldsymbol{W}(\boldsymbol{s})\boldsymbol{a} - \boldsymbol{a}^\top \boldsymbol{W}(\boldsymbol{s})\boldsymbol{b}(\boldsymbol{s}) + \frac{1}{2}\boldsymbol{b}(\boldsymbol{s})^\top \boldsymbol{W}(\boldsymbol{s})\boldsymbol{b}(\boldsymbol{s}) + V(\boldsymbol{s}) \\ &= \frac{1}{2}\boldsymbol{a}^\top \boldsymbol{W}(\boldsymbol{s})\boldsymbol{a} + \boldsymbol{a}^\top \boldsymbol{\psi}(\boldsymbol{s}) + \xi(\boldsymbol{s}), \end{aligned} \tag{59}$$

where $\boldsymbol{\psi}(\boldsymbol{s}) = -\boldsymbol{W}(\boldsymbol{s})\boldsymbol{b}(\boldsymbol{s})$ and $\xi(\boldsymbol{s}) = \frac{1}{2}\boldsymbol{b}(\boldsymbol{s})^\top \boldsymbol{W}(\boldsymbol{s})\boldsymbol{b}(\boldsymbol{s}) + V(\boldsymbol{s})$. By substituting the quadratic model obtained by NAF into the GAC framework, the guide actor is now given by $\tilde{\pi}(\boldsymbol{a}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{a}|\boldsymbol{\phi}_+(\boldsymbol{s}), \boldsymbol{\Sigma}_+(\boldsymbol{s})))$ with

$$\boldsymbol{\phi}_+(\boldsymbol{s}) = (\eta^\star \boldsymbol{\Sigma}^{-1}(\boldsymbol{s})) - \boldsymbol{W}(\boldsymbol{s}))^{-1}(\eta^\star \boldsymbol{\Sigma}^{-1}(\boldsymbol{s})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}) - \boldsymbol{W}(\boldsymbol{s})\boldsymbol{b}(\boldsymbol{s})) \tag{60}$$

$$\boldsymbol{\Sigma}_+(\boldsymbol{s}) = (\eta^\star + \omega^\star)(\eta^\star \boldsymbol{\Sigma}^{-1}(\boldsymbol{s})) - \boldsymbol{W}(\boldsymbol{s})). \tag{61}$$

To obtain Q-learning with NAF, we set $\eta^\star = 0$, i.e., we perform a greedy maximization where with the KL upper-bound approaches infinity, and this yields

$$\begin{aligned}\boldsymbol{\phi}_+(\boldsymbol{s}) &= -\boldsymbol{W}(\boldsymbol{s})^{-1}(-\boldsymbol{W}(\boldsymbol{s})\boldsymbol{b}(\boldsymbol{s})) \\ &= \boldsymbol{b}(\boldsymbol{s}), \end{aligned} \tag{62}$$

which is the policy obtained by performing Q-learning with NAF. Thus, NAF with Q-learning is a special case of GAC when the NAF model is used and the KL upper-bound approaches infinity.

A.6 Relation to Maximum Entropy Reinforcement Learning

The entropy constraint in the optimization problem in Eq.(13) suggests that GAC is closely related to the *maximum entropy reinforcement learning* (MaxEnt RL) framework. The MaxEnt RL aims to maximize an expected returns with an additional entropy bonus: $\sum_{t=1}^{\infty} \mathbb{E}_{p^\pi(\boldsymbol{s})}\left[r(\boldsymbol{s}_t,\boldsymbol{a}_t) + \alpha \mathrm{H}(\pi(\boldsymbol{a}_t|\boldsymbol{s}_t))\right]$ where $\alpha > 0$ is a trade-off parameter. The optimal policy is given by (Haarnoja et al., 2017)

$$\pi^\star_{\mathrm{MaxEnt}}(\boldsymbol{a}|\boldsymbol{s}) = \exp\left(\frac{Q^\star_{\mathrm{soft}}(\boldsymbol{s},\boldsymbol{a}) - V^\star_{\mathrm{soft}}(\boldsymbol{s})}{\alpha}\right) \propto \exp\left(\frac{Q^\star_{\mathrm{soft}}(\boldsymbol{s},\boldsymbol{a})}{\alpha}\right), \tag{63}$$

where the $V^\star_{\mathrm{soft}}(\boldsymbol{s})$ and $Q^\star_{\mathrm{soft}}(\boldsymbol{s},\boldsymbol{a})$ are the optimal soft state-value and action-value functions, respectively. The soft value functions of a policy π are defined as

$$V^\pi_{\mathrm{soft}}(\boldsymbol{s}) = \alpha \log \int \exp\left(\frac{Q^\pi_{\mathrm{soft}}(\boldsymbol{s},\boldsymbol{a})}{\alpha}\right) \mathrm{d}\boldsymbol{a}, \tag{64}$$

$$Q^\pi_{\mathrm{soft}}(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s},\boldsymbol{a}) + \mathbb{E}_{\pi(\boldsymbol{a}_t|\boldsymbol{s}_t), p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t)}\left[\sum_{t=1}^{\infty} \gamma^t (r(\boldsymbol{s}_t,\boldsymbol{a}_t) + \alpha \mathrm{H}(\pi(\cdot|\boldsymbol{s}_t)))\right] \tag{65}$$

The optimal policy and the soft state-value function closely resemble the guide actor in Eq.(14) and the log-integral term in Eq.(15), respectively, except for the definition of action-value functions and the weight $\pi(\boldsymbol{a}|\boldsymbol{s})^{\frac{\eta}{\eta+\omega}}$. To learn the optimal policy of MaxEnt-RL, Haarnoja et al. (2017) proposed *soft Q-learning* which uses importance sampling to compute the soft value functions and approximates the intractable policy using a separate policy function. The proposed method largely differs from soft Q-learning since we use Taylor's approximation to convert the intractable integral into more convenient matrix-vector products.

B Pseudo-code of GAC

The pseudo-code of GAC is given in Algorithm 1.

C Experiment details

C.1 Implementation

We try to follow the network architecture proposed by the authors of each baseline method as close as possible. For GAC and DDPG, we use neural networks with two hidden layers for the actor network

Algorithm 1 Guide actor critic

Input: Initial actor $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{a}|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{\Sigma})$, critic $\widehat{Q}_{\boldsymbol{\nu}}(\boldsymbol{s}, \boldsymbol{a})$, target critic network $\widehat{Q}_{\bar{\boldsymbol{\nu}}}(\boldsymbol{s}, \boldsymbol{a})$, KL bound ϵ, entropy bound κ, learning rates α_1, α_2, and data buffer $\mathcal{D} = \varnothing$.
for $t = 1, \ldots, T_{\max}$ **do**
  procedure COLLECT TRANSITION SAMPLE
    Observe state $\boldsymbol{s}_t$ and sample action $\boldsymbol{a}_t \sim \mathcal{N}(\boldsymbol{a}|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}_t), \boldsymbol{\Sigma})$.
    Execute $\boldsymbol{a}_t$, receive reward r_t and next state $\boldsymbol{s}'_t$.
    Add transition $\{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}'_t)\}$ to $\mathcal{D}$.
  end procedure
  procedure LEARN
    Sample N mini-batch samples $\{(\boldsymbol{s}_n, \boldsymbol{a}_n, r_n, \boldsymbol{s}'_n)\}_{n=1}^N$ uniformly from $\mathcal{D}$.
    procedure UPDATE CRITIC
      Sample actions $\{\boldsymbol{a}'_{n,m}\}_{m=1}^M \sim \mathcal{N}(\boldsymbol{a}|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}_n), \boldsymbol{\Sigma})$ for each $\boldsymbol{s}_n$.
      Compute y_n, update $\boldsymbol{\nu}$ by, e.g., Adam, and update $\bar{\boldsymbol{\nu}}$ by moving average:

$$y_n = r_n + \gamma \frac{1}{M} \sum_{m=1}^{M} \widehat{Q}_{\bar{\boldsymbol{\nu}}}(\boldsymbol{s}'_n, \boldsymbol{a}'_{n,m}), \tag{66}$$

$$\boldsymbol{\nu} \leftarrow \boldsymbol{\nu} - \alpha_1 \frac{1}{N} \sum_{n=1}^{N} \nabla_{\boldsymbol{\nu}} \left(\widehat{Q}_{\boldsymbol{\nu}}(\boldsymbol{s}_n, \boldsymbol{a}_n) - y_n \right)^2, \tag{67}$$

$$\bar{\boldsymbol{\nu}} \leftarrow \tau \boldsymbol{\nu} + (1 - \tau) \bar{\boldsymbol{\nu}}. \tag{68}$$

    end procedure
    procedure LEARN GUIDE ACTOR
      Compute $\boldsymbol{a}_{n,0}$ for each $\boldsymbol{s}_n$ by $\boldsymbol{a}_{n,0} = \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}_n)$ or $\boldsymbol{a}_{n,0} \sim \mathcal{N}(\boldsymbol{a}|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}_n), \boldsymbol{\Sigma})$.
      Compute $\boldsymbol{g}_0(\boldsymbol{s}) = \nabla_{\boldsymbol{a}} \widehat{Q}(\boldsymbol{s}_n, \boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{a}_{n,0}}$ and $\boldsymbol{H}_0(\boldsymbol{s}_s) = -\boldsymbol{g}_0(\boldsymbol{s}_n)\boldsymbol{g}_0(\boldsymbol{s}_n)^\top$.
      Solve for $(\eta^\star, \omega^\star) = \operatorname{argmin}_{\eta>0, \omega>0} g(\eta, \omega)$ by a non-linear optimization method.
      Compute the guide actor $\widetilde{\pi}(\boldsymbol{a}|\boldsymbol{s}_n) = \mathcal{N}(\boldsymbol{a}|\boldsymbol{\phi}_+(\boldsymbol{s}_n), \boldsymbol{\Sigma}_+(\boldsymbol{s}_n))$ for each $\boldsymbol{s}_n$.
    end procedure
    procedure UPDATE PARAMETERIZED ACTOR
      Update policy parameter by, e.g., Adam, to minimize the MSE:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_2 \frac{1}{N} \sum_{n=1}^{N} \nabla_{\boldsymbol{\theta}} \|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{s}_n) - \boldsymbol{\phi}_+(\boldsymbol{s}_n)\|_2^2. \tag{69}$$

      Update policy covariance by averaging the guide covariances:

$$\boldsymbol{\Sigma} \leftarrow \frac{1}{N} \boldsymbol{\Sigma}_+(\boldsymbol{s}_n). \tag{70}$$

    end procedure
  end procedure
end for
Output: Learned actor $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$.

and the critic network. For both networks the first layer has 400 hidden units and the second layer has 300 units. For NAF, we use neural networks with two hidden layers to represent each of the functions $\boldsymbol{b}(\boldsymbol{s}), \boldsymbol{W}(\boldsymbol{s})$ and $V(\boldsymbol{s})$ where each layer has 200 hidden units. All hidden units use the relu activation function except for the output of the actor network where we use the $\tanh$ activation function to bound actions. We use the Adam optimizer (Kingma & Ba, 2014) with learning rate 0.001 and 0.0001 for the critic network and the actor network, respectively. The moving average step for target networks is set to $\tau = 0.001$. The maximum size of the replay buffer is set to 1000000. The mini-batches size is set to $N = 256$. The weights of the actor and critic networks are initialized as described by Glorot & Bengio (2010), except for the output layers where the initial weights are drawn uniformly from $\mathrm{U}(-0.003, 0.003)$, as described by Lillicrap et al. (2015). The initial covariance $\boldsymbol{\Sigma}$ in GAC is set to be an identity matrix. DDPG and QNAF use the OU-process with noise parameters $\theta = 0.15$ and $\sigma = 0.2$ for exploration .

Table 1: The total computation time for training the policy for 1 million steps (0.1 million steps for the Invert-Pendulum task). The mean and standard error are computed over 10 trials with the unit in hours. TRPO is not included since it performs a lesser amount of update using batch data samples.

Task	GAC-1	GAC-0	DDPG	QNAF
Inv-Pend.	1.13(0.09)	0.80(0.04)	0.45(0.02)	0.40(0.03)
Inv-Double-Pend.	15.56(0.93)	15.67(0.77)	9.04(0.29)	7.47(0.22)
Reacher	17.23(0.87)	12.64(0.38)	10.67(0.37)	30.91(2.09)
Swimmer	12.43(0.74)	11.94(0.74)	9.61(0.52)	32.44(2.21)
Half-Cheetah	29.82(1.76)	32.64(1.41)	10.13(0.41)	27.94(2.37)
Ant	37.84(1.80)	40.57(3.06)	9.75(0.37)	27.09(0.94)
Hopper	18.99(1.06)	14.22(0.56)	8.74(0.45)	26.48(1.42)
Walker2D	33.42(0.96)	31.71(1.84)	7.92(0.11)	26.94(1.99)
Humanoid	111.07(2.99)	106.80(6.80)	13.90(1.60)	30.43(2.21)

For TRPO, we use the implementation publicly available at `https://github.com/openai/baselines`. We also use the provided network architecture and hyper-parameters except the batch size where we use 1000 instead of 1024 since this is more suitable in our test setup.

For GAC, the KL upper-bound is fixed to $\epsilon = 0.0001$. The entropy lower-bound κ is adjusted heuristically by

$$\kappa = \max(0.99(E - E_0) + E_0, E_0), \tag{71}$$

where $E \approx \mathbb{E}_{p^\beta(\boldsymbol{s})}\left[\mathrm{H}(\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}))\right]$ denotes the expected entropy of the current policy and E_0 denotes the entropy of a base policy $\mathcal{N}(\boldsymbol{a}|\boldsymbol{0}, 0.01\boldsymbol{I})$. This heuristic ensures that the lower-bound gradually decreases but the lower-bound cannot be too small. We apply this heuristic update once every 5000 training steps. The dual function is minimize by the sequential least-squares quadratic programming (SLSQP) method with an initial values $\eta = 0.05$ and $\omega = 0.05$. The number of samples for computing the target critic value is $M = 10$.

C.2 Environments and Results

We perform experiments on the OpenAI gym platform (Brockman et al., 2016) with Mujoco Physics simulator (Todorov et al., 2012) where all environments are v1. We use the state space, action space and the reward function as provided and did not perform any normalization or gradient clipping. The maximum time horizon in each episode is set to 1000. The discount factor $\gamma = 0.99$ is only used for learning and the test returns are computed without it.

Experiments are repeated for 10 times with different random seeds. The total computation time are reported in Table 1. The figures below show the results averaged over 10 trials. The y-axis indicates the averaged test returns where the test returns in each trial are computed once every 5000 training time steps by executing 10 test episodes without exploration. The error bar indicates standard error. The performance of each method is denoted by the following colors, red: GAC-1, blue: GAC-0, black: DDPG, green: QNAF, and purple: TRPO.

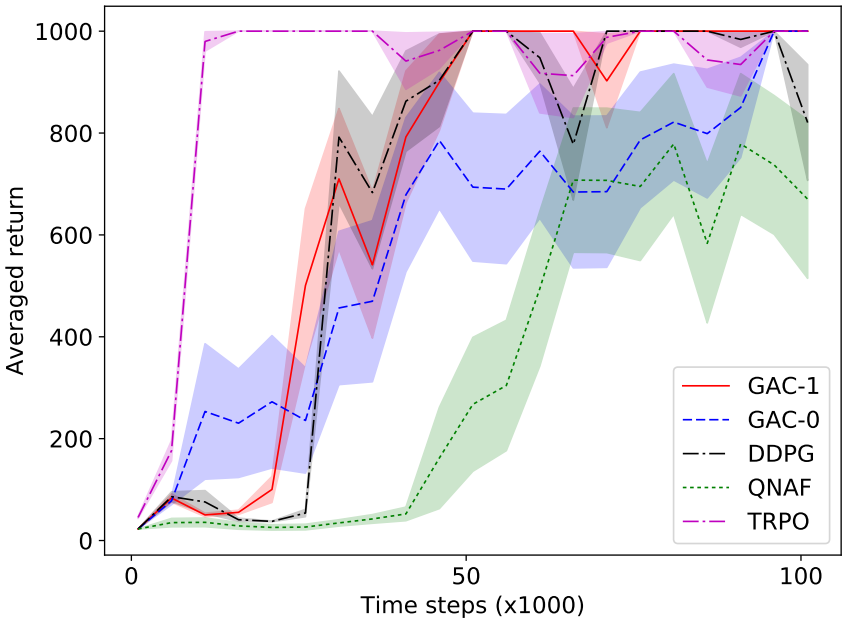


Figure 2: Performance averaged over 10 trials on the Inverted Pendulum task.

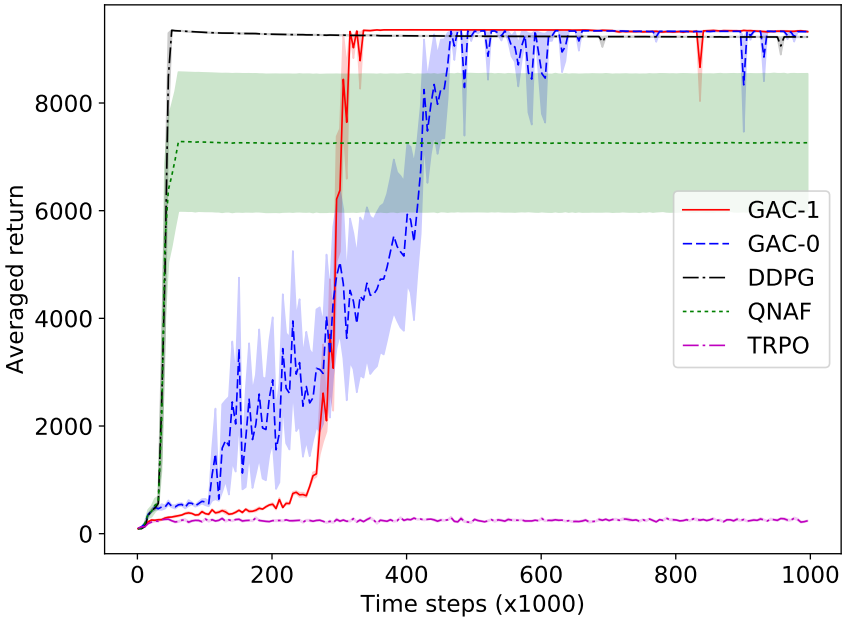


Figure 3: Performance averaged over 10 trials on the Inverted Double Pendulum task.

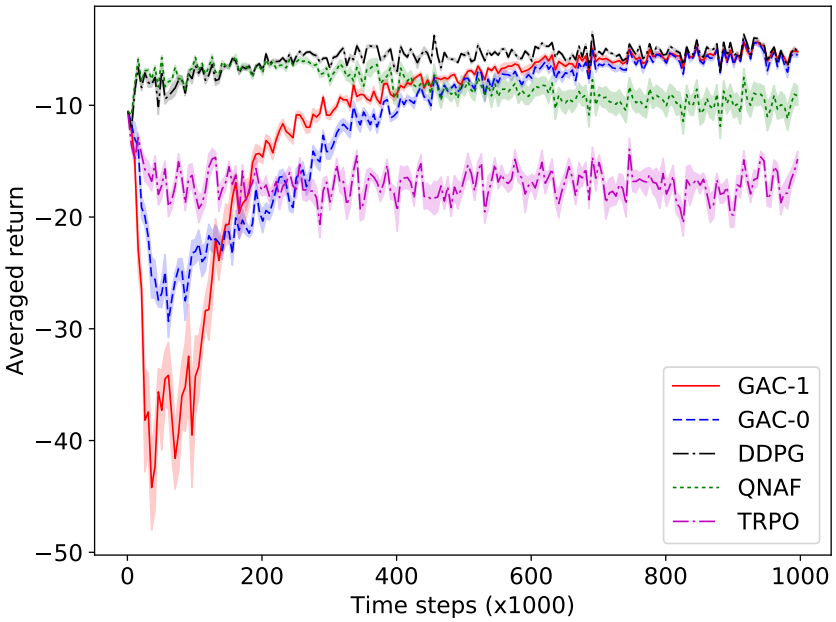


Figure 4: Performance averaged over 10 trials on the Reacher task.

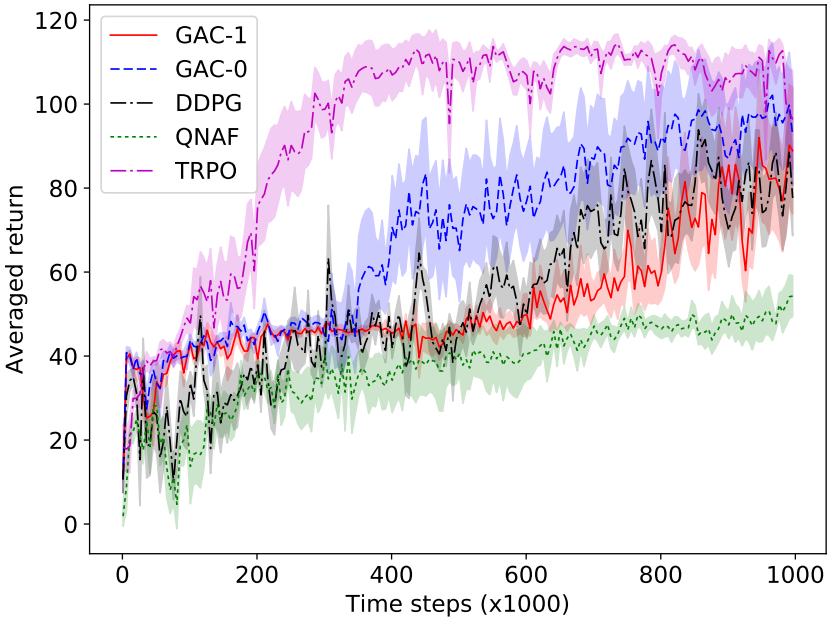


Figure 5: Performance averaged over 10 trials on the Swimmer task.

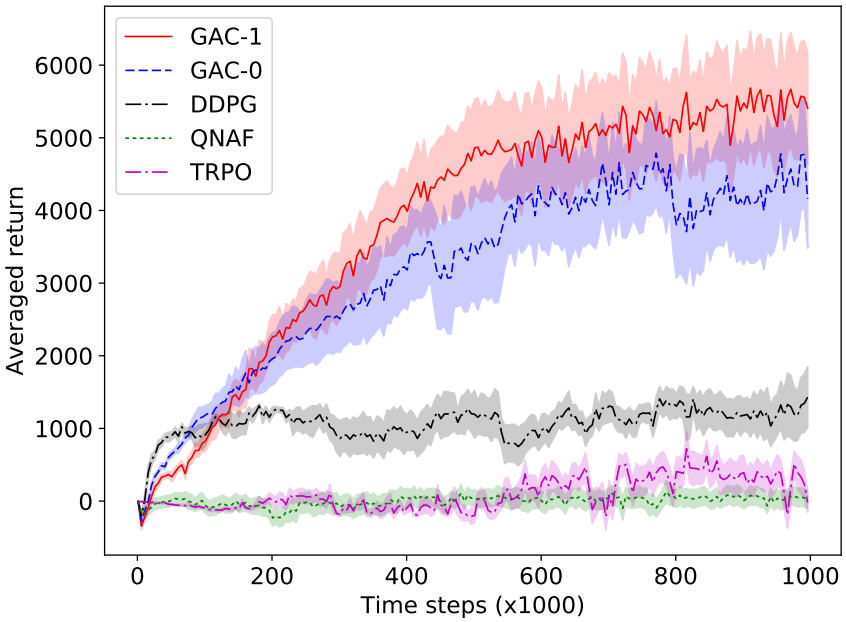


Figure 6: Performance averaged over 10 trials on the Half-Cheetah task.

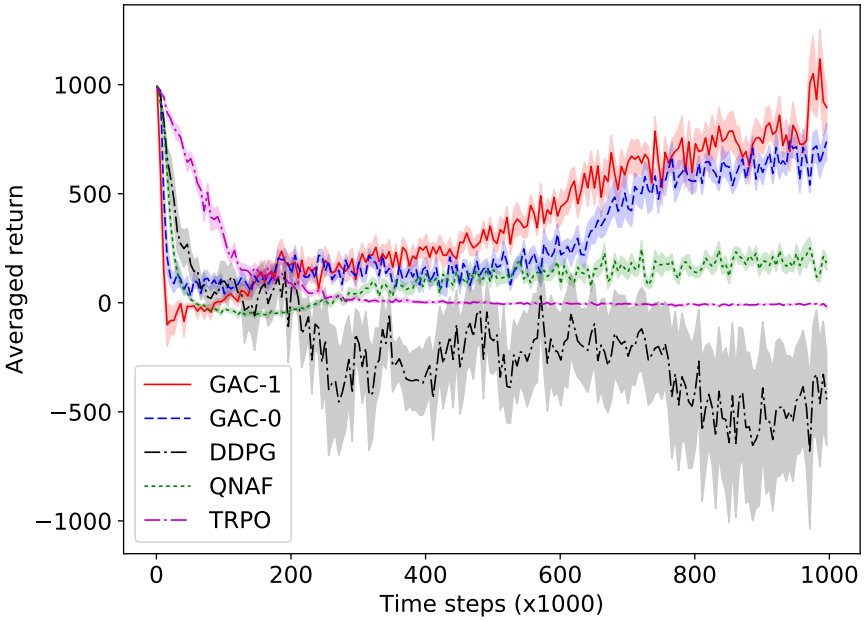


Figure 7: Performance averaged over 10 trials on the Ant task.

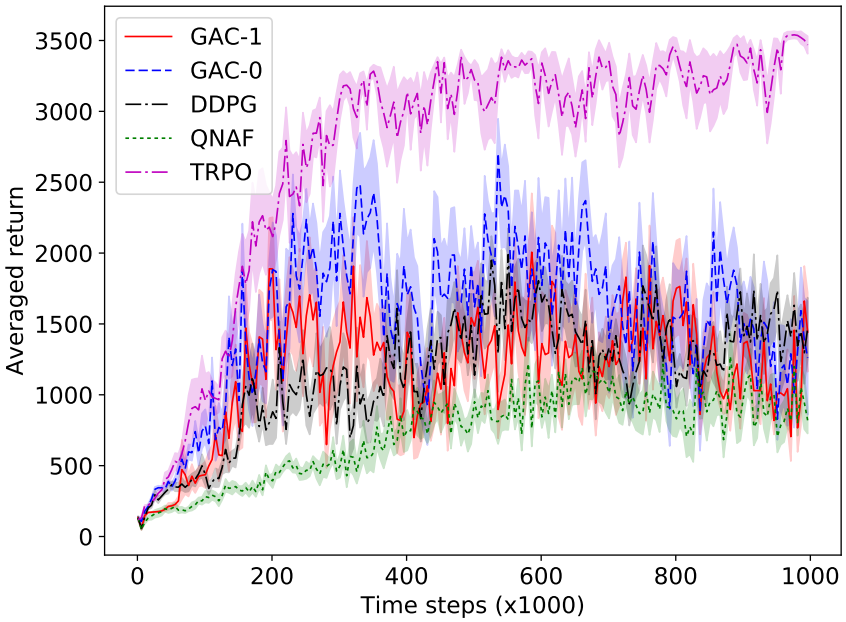


Figure 8: Performance averaged over 10 trials on the Hopper task.

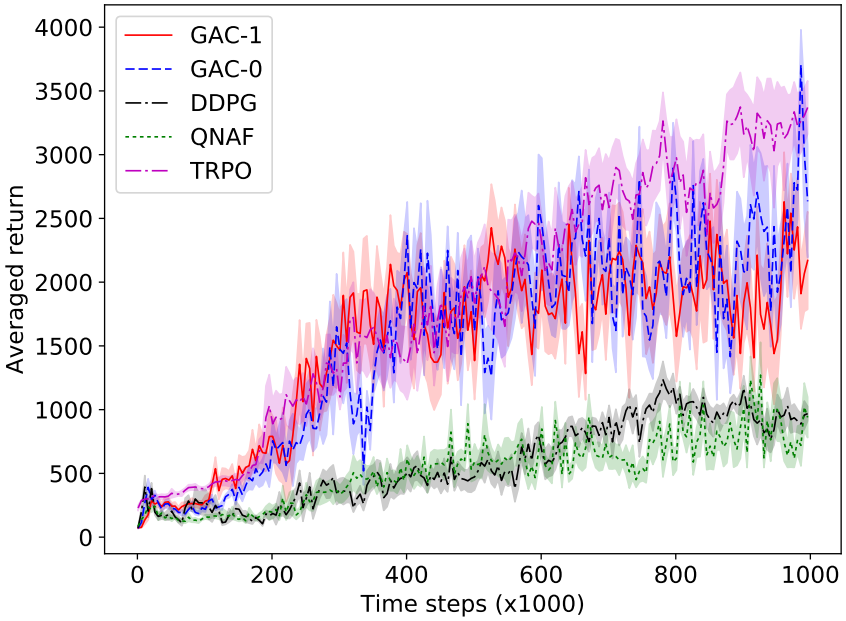


Figure 9: Performance averaged over 10 trials on the Walker2D task.

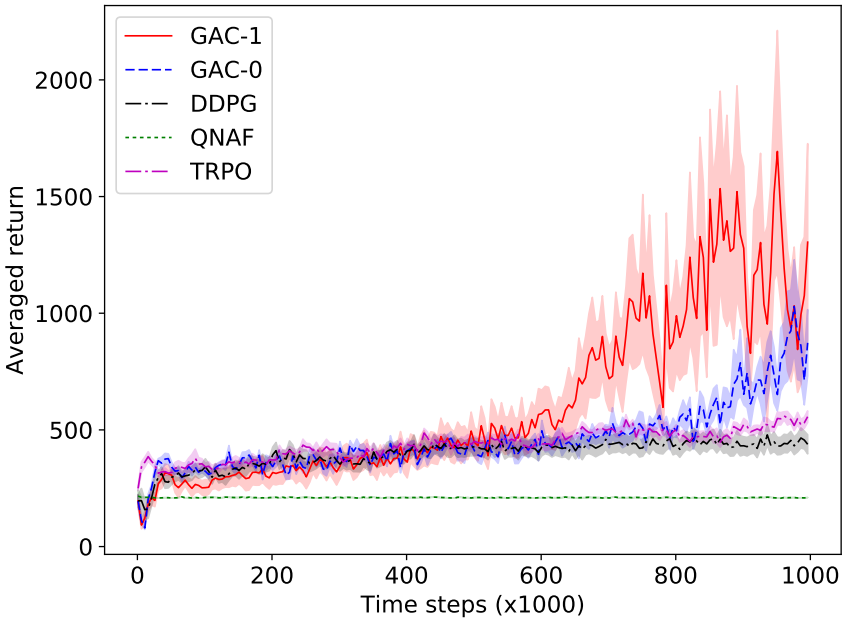


Figure 10: Performance averaged over 10 trials on the Humanoid task.