Markov Chain Monte Carlo Policy Optimization

David S. Johnson*
Department of Computer Science
Cranberry-Lemon University
Pittsburgh, PA 15213
johnson@cs.cranberry-lemon.edu

Abstract

Discovering approximately optimal policies in domains is crucial to applying reinforcement learning (RL) in many real-world scenarios, which is termed as policy optimization. By viewing the policy optimization from the perspective of variational inference, the representation power of policy network allows us to obtain the approximate posterior of actions conditioned on the states, with the entropy or KL regularization. However, in practice the policy optimization may lead to suboptimal policy estimates due to amortization gap. Inspired by the Markov Chain Monte Carlo (MCMC) techniques, instead of optimizing policy parameters or policy distributions directly, we propose a new policy optimization method, incorporating gradient-based feedback in various ways. The empirical evaluation verifies the performance improvement of the proposed method in many continuous control benchmarks.

1 Introduction

Reinforcement learning (RL) algorithms involve policy evaluation and policy optimization [32]. Given a policy, one can estimate the value for each state or state-action pair following that policy, and given a value estimate, one can improve the policy to maximize the value. This latter procedure, policy optimization, can be challenging in continuous control due to instability and poor asymptotic performance. In deep RL, where policies over continuous actions are often parameterized by deep networks, such issues are typically tackled using regularization from previous policies [29, 30] or by maximizing policy entropy [22, 9]. These techniques can be interpreted as variational inference [19], using optimization to infer a policy that yields high expected return while satisfying prior policy constraints.

However, one subtlety arises: when used with entropy or KL regularization, policy networks perform amortized optimization [11]. That is, many deep RL algorithms, such as soft actor-critic (SAC) [14], optimize a network to directly output the parameters of policy distribution, approximating the posterior of policy distribution given the collected trajectories. Specifically, it is realized by a direct mapping from states to policy distribution parameters. While direct amortization schemes have improved the efficiency of variational inference as encoder networks [18, 25, 21], the learned policy distribution can be sub-optimal [6, 17]. This suboptimality is typically defined as the amortization gap [6], resulting into a gap in policy optimization objective.

In order to fill in this amortization gap, recent work has sought to augment the direct amortization approach by transforming policy distribution through mappings with additional trainable parameters to achieve richer posterior approximations [2, 24], such as Normalizing Flow (NF) policy [33, 36]. Although it demonstrates success in various RL domains, the NF policy does not explicitly use

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

information about the target posterior of policy distribution. Hence, it is unclear whether the improved performance is resulted from better variational inference or simply the overparametrization of the sequence of transformations.

In this work, inspired by the techniques and progress in probabilistic inference [38], we propose to investigate the variational inference (VI) policy optimization augmented Markov Chain Monte Carlo (MCMC) iterations. VI chooses a family of tractable distributions, and tries to find the member of that family with the lowest KL divergence to the posterior, whereas MCMC simulates a Markov chain whose stationary distribution is the posterior. Generally speaking, VI is often much faster in finding a high posterior density region while MCMC can explore much more parameter space by random jumps conditioned on local information [26]. By merging the advantages of MCMC and VI to overcome each other's limitations, we start a Markov chain with initial values drawn from an optimized variational policy distribution, i.e., the base policy network, producing the output action or its distribution.

In this, we use MCMC iterations to transform the output of policy network to approximate the target posterior of actions. For deterministic policy, we adopt K steps of Langevin-dynamics (LD) MCMC [27]. But the density of the marginal distribution is implicitly defined in LD MCMC. When the output of policy network is a distribution and the action probability is explicitly evaluated in the policy optimization objective, Hamiltonian-dynamics MCMC (HMC) [23] is adopted in MCMC transformations. Both LD and HMC can be thought of as a Normalizing Flow scheme in which the flow depends explicitly on the target distribution. Building upon the framework of Maximum a posterior policy optimization (MPO) [1], we demonstrate performance improvements of MCMC-augmented policies over other methods in various aspects, such as action exploration and policy improvement.

In MPO, and many other algorithms, the KL divergence was added as a constraint to stabilize the learning process [1, 29]. But their solutions were built upon computing the inverse of Fisher information matrix, which has expensive computational overhead for high-dimensional parameters. In order to reduce computation complexity, we propose to use PID control to maintain the KL divergence distance from the previous to updated policies around a threshold value. Here we find that tuning the KL multiplier by PID control can stabilize the policy optimization significantly, improving the performance significantly.

2 Background

2.1 Preliminaries

We investigate Markov decision processes (MDP), where $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$ are the state and action at time step t, with the corresponding reward $r_t = r(s_t, a_t)$. The state transition of the environment is governed by $s_{t+1} \sim p_{\text{env}}(s_{t+1}|s_t, a_t)$, and the action is produced by the policy $\pi_\theta(a_t|s_t)$, parameterized by θ. The discounted sum of rewards is denoted as $\mathcal{R}(\tau) = \sum_t \gamma^t r_t$, where $\gamma \in (0, 1]$ is the discounted factor, and $\tau = (s_1, a_1, \ldots)$ is a trajectory. Thus, the distribution over the trajectory is

$$p(\tau) = \rho(s_1) \prod_{t=1}^{T} p_{\text{env}}(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$$

where the initial state is drawn from the distribution $\rho(s_1)$. The objective of RL is to maximize the expected discounted return $\mathbb{E}_{p(\tau)}[\mathcal{R}(\tau)]$.

At a given time step t, one can optimize this objective by estimating the accumulated future returns in the summation using an action-value network [21, 14], denoted as $Q_\pi(s, a)$ in terms of a policy π.

2.2 Reinforcement Learning via Probabilistic Inference

Recently a surge of works have formulated reinforcement learning and control as probabilistic inference [7, 35, 34, 3, 19]. In these approaches, the agent-environment interaction process is formulated as a probabilistic graphical model, then reward maximization is converted into maximum marginal likelihood estimation, where the policy resulting maximal reward is learned via probabilistic inference. This conversion is accomplished by introducing one or more binary observed variables $\mathcal{O}$,

whose probability conditioned on the trajectory can be expressed as

$$p(\mathcal{O}=1|\tau) \propto \exp(\mathcal{R}(\tau)/\alpha) \tag{1}$$

where α is the temperature hyper-parameter. By referring variables $\mathcal{O}$ as optimality [19], our target is to learn the policy π_θ which can produce actions maximizing the likelihood of optimality. However evaluating this likelihood, i.e., $p(\mathcal{O}=1) = \int p(\mathcal{O}=1|\tau)p(\tau)d\tau$, needs the averaging over all the trajectories, which is computationally intractable, especially in high-dimensional problems. Hence variational inference is adopted to lower bound the objective, where a variational distribution $q(\tau|\mathcal{O})$ is learned to approximate the posterior of trajectory given the optimality, i.e.,

$$q(\tau|\mathcal{O}) = \prod_{t=1}^{T} p_{\text{env}}(s_{t+1}|s_t, a_t)q(a_t|s_t, \mathcal{O}) \tag{2}$$

And we learn $q(a_t|s_t, \mathcal{O})$ to maximize the evidence lower bound (ELBO) of the objective, i.e., $\log p(\mathcal{O}=1)$, as below,

$$\begin{aligned} \log p(\mathcal{O}=1) &\geq \int q(\tau|\mathcal{O})\left[\log p(\mathcal{O}=1|\tau) + \log \frac{p(\tau)}{q(\tau|\mathcal{O})}\right] d\tau \\ &= \mathbb{E}_q[\mathcal{R}(\tau)/\alpha] - D_{\text{KL}}(q(\tau|\mathcal{O})\|p(\tau))) \end{aligned} \tag{3}$$

Simplifying the above equation further, we can get the objective of policy optimization as below

$$\mathcal{J}(q,\theta) = \mathbb{E}_{(s_t,r_t)\in\tau, a_t\sim q}\left[\sum_{t=1}^{T} \gamma^t r_t - \alpha \log \frac{q(a_t|s_t,\mathcal{O})}{\pi_\theta(a_t|s_t)}\right] \tag{4}$$

Specifically, at time t, this objective can be written as

$$\mathcal{J}(q,\theta) = \mathbb{E}_q[Q_q(s_t,a_t)] - \alpha D_{\text{KL}}(q(a_t|s_t,\mathcal{O})\|\pi_\theta(a_t|s_t)) \tag{5}$$

2.3 Hamiltonian Dynamics

Hamiltonian Monte Carlo (HMC) [23] is a classical MCMC method for sampling sequence of samples which converge to being distributed according to the target distribution. Based on the empirical success of Hamiltonian Monte Carlo [23, 15], many algorithms exploiting Hamiltonian dynamics have been developed to obtain unbiased estimates of the ELBO and its gradients [28, 37, 4]. In this work, we use multiple steps of Hamiltonian dynamics to improve action distribution and decrease the amortization gap in variational inference. Here we focus on the time-inhomogeneous Hamiltonian dynamics [23, 4]. This method uses reverse kernels which are optimal for reducing variance of the likelihood estimators and allows for simple calculation of the approximate posteriors of the latent variables. Additionally, we can easily use the reparameterization trick to calculate unbiased gradients of the ELBO with respect to the parameters of interest.

2.4 Optimal Control

Here we regard the policy optimization iterations as system dynamics subject to an external influence, or control. A standard formulation for discrete-time systems with feedback control is:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= F(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ \boldsymbol{y}_k &= Z(\boldsymbol{x}_k) \\ \boldsymbol{u}_k &= h(\boldsymbol{y}_0, \ldots, \boldsymbol{y}_k) \end{aligned}$$

with state vector $\boldsymbol{x}$, dynamics function F, measurement outputs $\boldsymbol{y}$, applied control $\boldsymbol{u}$, and the subscript denoting the time step. The feedback rule h has access to past and present measurements. The goal in optimal control is to design a control rule, h, that results in a sequence $\boldsymbol{y}_{0:T} = \{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_T\}$, scoring highly according to some metric C.

It is always easier to analyze and control systems with simpler dependence on the input, even though the dependence on the state is complex. Control-affine systems are a broad class of dynamical systems which are especially amenable to analysis [16]. Generally the dynamics there take the form

$$F(\boldsymbol{x}_k, \boldsymbol{u}_k) = f(\boldsymbol{x}_k) + g(\boldsymbol{x}_k)\boldsymbol{u}_k$$

where f and g can be non-linear and unknown a prior.

3 MCMC Policy Optimization

Policy-based approaches in RL can be viewed as amortized variational inference, where the policy network produces action or its distribution to approximate the posterior given the optimality $\mathcal{O}$ [19]. Hence, the policy network plays the same role of encoder in variational autoencoder (VAE) [18]. However, in VAE, both empirical observations and theoretical study show that the encoder network can only produce sub-optimal approximate estimates of the target posterior, and the resulting gap in the variational bound is termed as *amortization gap* [6]. In this work, in order to close this gap, we propose to mix the policy network with MCMC iterations. Here MCMC transitions are initialized by the output of policy network, and the optimization of hyper-parameters in MCMC transitions is alternating with the optimization of policy network parameters.

3.1 Deterministic Policy

For deterministic policy, the policy network is a mapping from current state to action directly, where the variational inference cannot be applied. In this case, in order to substitute the variational framework, we formulate the policy optimization as a deterministic autoencoder (AE), i.e., $\mathcal{L}_{\text{AE}} = \mathcal{L}_{\text{REC}} + \alpha\mathcal{L}_Z^{\text{RAE}}$ [12]. By formulating our problem as AE, the reconstruction loss $\mathcal{L}_{\text{REC}}$ is just negative of Q value from $Q(s, a)$. And latent loss $\mathcal{L}_Z^{\text{RAE}}$ can be set to $\|a - \pi_{\theta_t}(s)\|$, since the latent loss is essentially simplified KL loss [12] and the prior of action is assumed to be the last updated policy. Thus the policy optimization objective to be maximized is formulated as

$$U_{\theta_t}(s, a) = Q_{\pi_{\theta_t}}(s, a)/\alpha - \|a - \pi_{\theta_t}(s)\| \tag{6}$$

In order to better estimate the maximum a posterior (MAP) of actions, we adopt Langevin-dynamic (LD) [27] to transform the output of policy network. LD is used here because exploration is more important for deterministic policy, and random noise injected in every iteration of LD Markov chain can augment exploration a lot. Assume there are K iterations of LD Markov chain. For k-th LD iteration, given current state s, the action is updated as below,

$$a_{k+1} = \frac{\epsilon_k}{2}\left(\nabla_a U(s, a)_{a=a_k}\right) + \xi_k \tag{7}$$

where ϵ_k is the step size and $\xi_k \sim \mathcal{N}(0, \lambda^2 I)$. We define a vector $h_{\text{LD}} = [\epsilon_1, \epsilon_2, \ldots, \epsilon_K, \lambda]$, containing all the hyper-parameters in LD Markov chains. Denote the K steps of LD transitions as a function $f_{\text{LD}}^K(a, h_{\text{LD}})$ with initial action a, whose output is the action a_K at K-th step of LD transitions.

However, the action updates in (7) are not deterministic, which cannot be applied to algorithms based on deterministic policy gradient theorem [31], such as DDPG [31] and TD3 [10]. Besides, scalar hyper-parameters h_{LD}, controlling stepsize and covariance of noise, are not flexible enough to adapt to complex critic function. Here we formulate the transformations of actions via Langevin dynamics as a normalizing flow [24, 33, 36, 20]. Instead of simply applying Langevin dynamics in the classical formulation, in order to adapt the local geometry of the critic function, inspired by Langevin flows [13], we add extra neural networks to augment the flexibility of the action transformations. All extra neural networks are simple and do not increase the model complexity significantly. And we do not need noise in any coupling layer, making transformations deterministic.

Specifically, we define the action transformations as a sequence of coupling layers:

$$f_{\text{LD}}^K(a, \{\phi_1, \ldots, \phi_K\}) = g_{\phi_K} \circ \cdots \circ g_{\phi_1}(a) \tag{8}$$

where the k-th coupling layer g_{ϕ_k} with mask m_l transforms action x to y. Building upon [8], the coupling layer incorporates the Langevin dynamics as below,

$$\begin{aligned} y \;=\; & m \odot x + \\ & \left(\overline{m} \odot x - \overline{m} \odot \frac{\epsilon^2}{2}\nabla U(x) + \epsilon \exp(\sigma(m \odot x))\right) \odot \exp(S(m \odot x)) \\ & + T(m \odot x) \end{aligned} \tag{9}$$

where the subscript dependent on k is omitted. The masks m in every coupling layer are binary vectors, with half of elements to be 1. We define $\sigma(\cdot), S(\cdot)$ and $T(\cdot)$ as neural networks σ, S, T :

$\mathbb{R}^{d_a/2} \longrightarrow \mathbb{R}^{d_a/2}$, whose parameters at the k-th layer are denoted as ϕ_k. The neural networks σ, S, T in all K layers are trained together in an end-to-end manner.

Combining all the discussions above, the policy optimization objective is as below, in iteration t,

$$\mathcal{J}_{\text{LD}}(\theta, h) = \mathbb{E}_{s\sim d^{\mathcal{D}}(s)}\Big[Q_{\pi_{\theta_t}}\big(s, a_K\big) - \beta_t \|a_K - \pi_{\theta_t}(s)\|\Big] \tag{10}$$

where a_K is the output of $f_{\text{LD}}^K(\pi_\theta(s), h)$, and $d^{\mathcal{D}}(s)$ is the distribution of states stored in replay buffer $\mathcal{D}$. Here the second term in (10) is essentially same as the KL divergence loss in the variational framework (5). The parameters, $h := \{\phi_k\}_{k=1}^K$, consist of the parameters of neural networks in every coupling layer. We replace α by β_t dependent on step t, since it is tuned adaptively by PID control by the end of every policy optimization iteration.

In implementation, we update the policy parameters θ and LD transition hyperparameters h in an alternating way. That is

$$\theta_{t+1} = \arg\max_\theta \mathcal{J}(\theta, h_{\text{LD}}^t) \quad \text{and} \quad h_{\text{LD}}^{t+1} = \arg\max_h \mathcal{J}(\theta_{t+1}, h) \tag{11}$$

When optimizing the objective (10), we use stochastic gradient descent (SGD) for multiple steps, where in each step a minibatch of states are uniformly sampled from the replay buffer $\mathcal{D}$ and the expectation becomes empirical averaging over the minibatch.

3.2 Stochastic Policy

For stochastic policy, the output of policy network is a distribution of action. Since the density of actions is needed to be evaluated in the stochastic policy optimization algorithms, we adopt Hamiltonian Monte Carlo (HMC) [23, 4] to improve the action distribution, generated by policy network, to better approximate the posterior of action given optimality. The marginal likelihood after HMC can be evaluated, since the composition of equations in Hamiltonian dynamics has unit Jacobian [23].

Specifically, in HMC, we introduce momentum variables ρ to pair with the action a, and make Markov chain to work in an extended space $(a, \rho) \in \mathbb{R}^{d_a} \times \mathbb{R}^{d_a}$. Assume the momentum variable ρ has Gaussian prior $\mathcal{N}(\rho|0, I)$, and the prior of action a is the output distribution of the last updated policy $\pi_{\theta_t}(a|s)$ for iteration t of policy optimization. Therefore, the target distribution for variational policy optimization, i.e., unnormalized posterior of action and momentum, becomes

$$\bar{p}_\alpha(s, a, \rho) \propto \exp\big(Q_{\pi_{\theta_t}}(s, a)/\alpha\big)\pi_{\theta_t}(a|s)\mathcal{N}(\rho|0, I) \tag{12}$$

Since the stochastic policy is optimized in the formulation of variational inference, the variational distribution of action and momentum variable can be the policy distribution being updated and prior of momentum, i.e., $\pi_\theta(a|s)\mathcal{N}(\rho|0, I)$. The core idea is to improve variational distributions by applying multiple steps of deterministic Hamiltonian transitions, to better approximate the posterior and decrease the amortization gap [6]. Denote one step of Hamiltonian transition as $\Phi_{\theta_t, h_{\text{HMC}}}^k$, where θ_t is the parameters contained in the potential function of Hamiltonian dynamics and the vector h_{HMC} contains the hyper-parameters used in Hamiltonian transitions. Hence, $\{\Phi_{\theta, h_{\text{HMC}}}^k\}_{k=1}^K$ denotes time-discretized and inhomogeneous Hamiltonian dynamics, which discretizes the Hamiltonian dynamics into K steps of transitions [23, 4]. Different from normalizing flow policy [33, 36], in addition to the acceleration from gradient information, the log density is easy to compute here, without any complex inference. Each transformation $\Phi_{\theta_t, h}^k$ represents a leapfrog step, transforming (a, ρ) to (a', ρ'), which can be described as:

$$\begin{aligned} \tilde{\rho} &= \rho - \frac{\epsilon}{2} \odot \nabla U_{\theta_t}(s, a) \\ a' &= a + \epsilon \odot \tilde{\rho} \\ \rho' &= \tilde{\rho} - \frac{\epsilon}{2} \odot \nabla U_{\theta_t}(s, a') \end{aligned} \tag{13}$$

where U_{θ_t} is the potential function containing parameters θ_t, ∇ is differentiation taken with respect to a, and step size $\epsilon \in \mathbb{R}^{d_a}$. By ignoring the normalizer of target distribution (12), the potential function used in leapfrog (13) can be described as $U_{\theta_t}(s, a) = -(Q_{\pi_{\theta_t}}(s, a)/\alpha + \log \pi_{\theta_t}(a|s))$. Specifically, the hyperparameter h_{HMC} in HMC only contains step sizes $\epsilon \in \mathbb{R}^{d_a}$. Thus we have

the deterministic transition as $(a_k, \rho_k) = \Phi^k_{\theta_t, h_{\text{HMC}}}(a_{k-1}, \rho_{k-1})$. Denote the distribution of action and momentum variable at k-th step of transitions as $q^k_{\theta_t, h_{\text{HMC}}}(a_k, \rho_k)$, and the initial action and momentum are sampled from variational distributions, i.e., $q^0_{\theta_t, h_{\text{HMC}}}(a_0, \rho_0) := \pi_\theta(a|s)\mathcal{N}(\rho|0, I)$. Thus the process of transforming (a_0, ρ_0) to (a_K, ρ_K) can be defined as

$$(a_K, \rho_K) = f^K_{\text{HMC}}(a_0, \rho_0, \theta_t, h_{\text{HMC}}) := \big(\Phi^K_{\theta_t, h_{\text{HMC}}} \circ \cdots \circ \Phi^1_{\theta_t, h_{\text{HMC}}}\big)(a_0, \rho_0) \tag{14}$$

where $(a_0, \rho_0) \sim \pi_\theta(a_0|s)\mathcal{N}(0, I)$.

Since the policy optimization for stochastic policies is in the formulation of variational inference, the objective should be the ELBO, for both updating policy $\pi_\theta(a|s)$ and optimizing HMC hyper-parameters h_{HMC}. Based on the foundations of variational inference, the ELBO here can be written as the difference of the log target distribution and log variational distribution, i.e.,

$$\mathcal{L}_{\text{ELBO}}(s; \theta, h) = \mathbb{E}_{(a_0, \rho_0) \sim \pi_\theta(\cdot|s)\mathcal{N}(\cdot|0, I)}[\log \bar{p}(s, a_K, \rho_K) - \log q^K_{\theta_t, h}(a_K, \rho_K)] \tag{15}$$

where $(a_K, \rho_K) = f^K_{\text{HMC}}(a_0, \rho_0, \theta_t, h)$ defined in (14), θ_t is the parameters of last updated policy network regarded as the action prior, and parameters to be optimized consist of policy network parameters θ and hyper-parameters in Hamiltonian transitions h.

According to [4], based on the change of variable formula in probability density, the distribution of action and momentum variables at K-th step of leapfrog $\Phi^K_{\theta_t, h_{\text{HMC}}}$ can be expressed as

$$\begin{aligned} q^K_{\theta_t, h_{\text{HMC}}}(a_K, \rho_K) &= q^0_{\theta_t, h_{\text{HMC}}}(a_0, \rho_0) \prod_{k=0}^{K-1} \left| \det \nabla \Phi^{k+1}_{\theta_t, h_{\text{HMC}}}(a_k, \rho_k) \right|^{-1} \\ &= \pi_\theta(a_0|s)\mathcal{N}(\rho_0|0, I) \end{aligned} \tag{16}$$

where the second equality is due to the unit Jacobian of leapfrog step (13). It also helps us to evaluate the density of transformed actions in a convenient approach. Therefore, considering (3)(12)(16) together, the EBLO averaged over states in replay buffer can then be expressed as

$$\mathcal{J}(\theta, h) = \mathbb{E}_{s \sim d^{\mathcal{D}}(s)}\left[Q_{\pi_{\theta_t}}(s, a_K) - \beta_t \log \frac{\pi_\theta(a_0|s)}{\pi_{\theta_t}(a_K|s)} - \frac{\beta_t}{2}\rho_K^T \rho_K\right] \tag{17}$$

where $(a_0, \rho_0) \sim \pi_\theta(\cdot|s)\mathcal{N}(\cdot|0, I)$ and $(a_K, \rho_K) = f^K_{\text{HMC}}(a_0, \rho_0, \theta_t, h)$. Here we replace the temperature parameter α in (12) by β_t, which is tuned automatically by PID control, to keep the KL distance between previous and updated policies around a pre-defined value. The parameters of policy network θ and hyper-parameters of HMC are optimized in an alternative way same as (11).

3.3 Theoretical Analysis

By regarding the policy distribution $\pi_\theta(a|s)$ as the variational distribution, we essentially improve the variational distribution by running HMC method initialized at $\pi_\theta(a|s)$ and whose stationary distribution is the posterior of the interst $\bar{p}(a|s)$. Assume we run the HMC iteration for k steps, by marginalizing the momentum ρ, the distribution of action is denoted as $q^k_{\theta, h}(a)$. It is provable that the distribution $q^k_{\theta, h}(a)$ is an improvement of the variational distribution and it is closer to the target posterior $\bar{p}(a|s)$ in terms of KL divergence [5], reducing the *amortization gap*,

$$\text{KL}(q_\theta(a)|\bar{p}(a|s)) \geq \text{KL}(q^k_\theta(a)|\bar{p}(a|s))$$

According to [23], as the step $k \to \infty$, the distribution $q^k_{\theta, h}(a)$ will tend to $\bar{p}(a|s)$ in terms of KL divergence. So the improvement of our method can be proved. And empirically it is enough to have around $K = 5$ steps of HMC iterations.

4 Implementation

4.1 Tuning Multiplier β_t by PID Control

In implementation, the policy distance, the second term in (10)(17), is calculated by the empirical average over the minibatch of states randomly sampled from the replay buffer. Hence, $\hat{D}_{\text{LD}} =$

$\hat{\mathbb{E}}_{s\sim\mathcal{B}}[\|a_K - \pi_{\theta_t}(s)\|]$ and $\hat{D}_{\text{HMC}} = \hat{\mathbb{E}}_{s\sim\mathcal{B}}\big[\log \frac{\pi_\theta(a_0|s)\mathcal{N}(\rho_0|0,I)}{\pi_{\theta_t}(a_K|s)\mathcal{N}(\rho_K|0,I)}\big]$, for deterministic and stochastic policy respectively.

By representing $\hat{D}_{\text{LD}}$ and $\hat{D}_{\text{HMC}}$ as $\hat{D}$, the policy optimization can be formulated as a first-order system, i.e., $\theta_{t+1} = F(\theta_t, h, \beta_t), y_t = \hat{D}, \beta_t = h(y_0, \ldots, y_t, \epsilon)$, where $F(\theta_t, h, \beta_t) = \theta_t + \gamma \cdot \nabla\hat{\mathbb{E}}[Q(s, a_K)] + \beta \cdot \gamma \cdot \nabla\hat{D}$ with learning rate γ, function h is the PID control rule, and ϵ is the pre-defined threshold for the distance. The control rules h for updating β_t can be described as below,

$$\begin{aligned} \Delta &\longleftarrow \hat{D} - \epsilon, \quad \text{and} \quad \partial \longleftarrow \hat{D} - \hat{D}_{\text{prev}} \\ I &\longleftarrow (I + \Delta)_+, \quad \text{and} \quad \beta \longleftarrow (K_P\Delta + K_I I + K_D\partial)_+ \end{aligned}$$

where $\hat{D}_{\text{prev}}$ is the empirical distance in the last iteration. And here K_P, K_I, K_D are tuned empirically.

4.2 Explicit and Implicit Gradient

In the practical policy optimization, the action value given by the Q network may not be accurate, and directly incorporating gradient information like (13) may lead the policy distribution to local optima. To tackle this problem, we propose to incorporate gradient information in both explicit and implicit approaches, controlled by a trade-off parameter σ. The explicit approach is to directly use the gradient of Q network with respect to action like (13), while the implicit approach is to use the gradient processed by a function T. Specifically, σ_h and T_h, parameterized by h, are realized by simple neural networks with sigmoid and tanh functions as activations respectively. Their inputs are gradient, action and state, where the state is optional and can be ignored in some environments. The output of σ_h is a scalar, controlling the trade-off between explicit and implicit gradients. The output of T_h has the same dimension as action. Therefore, the leapfrog operation can be rewritten as

$$\tilde{\rho} = \rho - \frac{\epsilon}{2} \odot (\sigma_h(s, a, g) \cdot g + (1 - \sigma_h(s, a, g)) \cdot T_h(s, a, g)) \tag{18}$$

$$\rho' = \tilde{\rho} - \frac{\epsilon}{2} \odot (\sigma_h(s, a, g') \cdot g' + (1 - \sigma_h(s, a, g')) \cdot T_h(s, a, g')) \tag{19}$$

where $g := \nabla U_{\theta_t}(s, a), g' := \nabla U_{\theta_t}(s, a')$ and $a' = a + \epsilon \odot \tilde{\rho}$. In order to stabilize the learning process, the gated mechanism is used for the gradient trade-off. It is straightforward that the modified leapfrog (19) still has unit Jacobian, and it leaves the canonical joint distribution of action and momentum (a, ρ) invariant.

Empirically, we find that it is enough to choose simple architecture for the neural networks σ_h and T_h, i.e., two-layer with 8 or 16 hidden neurons. Otherwise, the complex architecture can make learning unstable, and the learned policy may be far from optimality. During training, neural network parameters h are included into the parameters of HMC iteration, optimized alternatively with the parameters of the policy network θ.

The proposed method is shown in Algorithm 1.

Algorithm 1: Hamiltonian Policy Optimization

Data: Denote a_t, s_t as the action and state at time$t$; Denote the replay buffer as $\mathcal{B}$;
Initialize θ, h;
for $t = 1, 2, \ldots$ **do**
  Sample $a_t \sim \pi_{\theta_t}(\cdot|s_t)$;
  Obtain $a_t^K, \rho_t^K = \text{HMC}(s_t, a_t; K, h_t)$;
  Apply a_t^K into the environment, and obtain next state s_{t+1};
  Store the experience tuple (s_t, a_t^K, s_{t+1}) into $\mathcal{B}$;
  Sample a minibatch of experience tuples $\mathcal{D}_t$ from $\mathcal{B}$;
  Update the Q network by $\mathcal{D}_t$;
  Obtain the transformed action and momentum $a^K, \rho^K = \text{HMC}(s, a; K, h_t), \forall(s, a) \in \mathcal{D}_t$;
  Update the policy and HMC parameters by optimizing $\mathcal{J}(\theta, h)$ defined in (17);
end

Algorithm 2: HMC$(s, a; K, h), \epsilon \in h$

Sample $\rho^0 \sim \mathcal{N}(0, I)$;
for $k = 1, \ldots, K$ **do**
    Obtain $\tilde{\rho}^k$ by (18);
    Update $a^k = a^{k-1} + \epsilon \odot \tilde{\rho}^k$;
    Obtain ρ^k by (19);
end
Output: a^K, ρ^K

5 Experiments

The second set of experiments studies the performance improvement of the multi-actor method. The environments are Hopper, HalfCheetah, and Ant. The performance metric is cumulative reward at the step of $5e5$, obtained from 5 random seeds. The proposed method uses PID control to update the KL multiplier β_t. The baseline is SAC [14], where the entropy coefficient α is set to 0.2 in Humanoid-v2 and 0.05 in other environments. The policy networks in both proposed method and SAC are the same, having two-layer with 256 hidden neurons in each layer. The neural networks T_h and σ_h in Hamiltonian dynamics are very small, only having two layers with 16 neurons in each layer.

Table 1: Summary of quantitative results. All results correspond to the original exact reward defined in OpenAI Gym

	Hopper	HalfCheetah	Ant
SAC	3281.1 ± 168.1	2613.2 ± 102.8	3492.1 ± 126.9
Ours	$\mathbf{4021.3 \pm 172.2}$	$\mathbf{2925.7 \pm 133.1}$	$\mathbf{4037.1 \pm 158.1}$

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

[2] Shubhankar Agarwal, Harshit Sikchi, Cole Gulino, and Eric Wilkinson. Imitative planning using conditional normalizing flow. *arXiv preprint arXiv:2007.16162*, 2020.

[3] Matthew Botvinick and Marc Toussaint. Planning as inference. *Trends in cognitive sciences*, 16(10):485–488, 2012.

[4] Anthony L Caterini, Arnaud Doucet, and Dino Sejdinovic. Hamiltonian variational auto-encoder. In *Advances in Neural Information Processing Systems*, pages 8167–8177, 2018.

[5] T Cover and J Thomas. Elements of information theory, 2ndwiley. *Hobo-ken, NJ*, 2006.

[6] Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086, 2018.

[7] Peter Dayan and Geoffrey E Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997.

[8] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[9] Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 202–211, 2016.

[10] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[11] Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the annual meeting of the cognitive science society*, volume 36, 2014.

[12] Partha Ghosh, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436*, 2019.

[13] Minghao Gu, Shiliang Sun, and Yan Liu. Dynamical sampling with langevin normalization flows. *Entropy*, 21(11):1096, 2019.

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870, 2018.

[15] Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

[16] Alberto Isidori. *Nonlinear control systems*. Springer Science & Business Media, 2013.

[17] Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pages 2678–2687, 2018.

[18] Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *Second International Conference on Learning Representations, ICLR*, volume 19, 2014.

[19] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[20] Bogdan Mazoure, Thang Doan, Audrey Durand, Joelle Pineau, and R Devon Hjelm. Leveraging exploration in off-policy algorithms via normalizing flows. In *Conference on Robot Learning*, pages 430–444. PMLR, 2020.

[21] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, pages 1791–1799, 2014.

[22] 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 *International conference on machine learning*, pages 1928–1937, 2016.

[23] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[24] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[25] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

[26] Christian P Robert, Víctor Elvira, Nick Tawn, and Changye Wu. Accelerating mcmc algorithms. *Wiley Interdisciplinary Reviews: Computational Statistics*, 10(5):e1435, 2018.

[27] Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.

[28] Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226, 2015.

[29] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[31] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. 2014.

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

[33] Yunhao Tang and Shipra Agrawal. Boosting trust region policy optimization by normalizing flows policy. *arXiv preprint arXiv:1809.10326*, 2018.

[34] Emanuel Todorov. General duality between optimal control and estimation. In *2008 47th IEEE Conference on Decision and Control*, pages 4286–4292. IEEE, 2008.

[35] Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *Proceedings of the 23rd international conference on Machine learning*, pages 945–952, 2006.

[36] Patrick Nadeem Ward, Ariella Smofsky, and Avishek Joey Bose. Improving exploration in soft-actor-critic with normalizing flows policies. *arXiv preprint arXiv:1906.02771*, 2019.

[37] Christopher Wolf, Maximilian Karl, and Patrick van der Smagt. Variational inference with hamiltonian monte carlo. *arXiv preprint arXiv:1609.08203*, 2016.

[38] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.