Bayesian Deep Q-Learning via Continuous-Time Flows

Ruiyi Zhang
Duke University
Durham, NC 27708
`ryzhang@cs.duke.edu`

Changyou Chen
SUNY at Buffalo
Buffalo, NY 14260
`cchangyou@gmail.com`

Chunyuan Li, Lawrence Carin
Duke University
Durham, NC 27708
`{cl319, lcarin}@duke.edu`

Abstract

Efficient exploration in reinforcement learning (RL) can be achieved by incorporating uncertainty into model predictions. Bayesian deep Q-learning provides a principle way for this by modeling Q-values as probability distributions. We propose an efficient algorithm for Bayesian deep Q-learning by posterior sampling actions in the Q-function via *continuous-time flows* (CTFs), achieving efficient exploration without explicit assumptions on the forms of posterior distributions. Specifically, based on the recently proposed soft Q-learning framework, our algorithm learns an energy-based policy, which is distilled into a sampling network via CTF for efficient action generation. The distillation procedure relies on the recently developed technique of particle optimization in CTF. Experiments on the toy and real tasks demonstrate excellent exploration ability of our algorithm, obtaining improved performance, compared to related techniques such as Stein variational gradient descent in standard soft Q-learning.

1 Introduction

We consider reinforcement learning (RL) for the task of sequential decision-making under unknown environment dynamics. A typical formulation for sequential decision-making is Markov decision process (MDP), in which an agent interacts with an environment in the attempt to maximize the accumulated reward. At each interaction, an agent takes action based on the current state, a reward will be returned from the environment, and the agent will transit to a new state. Due to the unknown underlying dynamics for the states and the rewards, the agent is trained to manage trade-off between exploration and exploitation, *i.e.*, exploring some potential optimal actions in the long term, or taking current known optimal action to maximize the current reward.

As an effective and popular way to tackle exploration-exploitation trade-off, Thompson sampling (TS) maintains a posterior distribution over some quantity, which is used to generate the next reward. TS is believed better than optimization for MDPs [22] due to its feasibility to incorporate the problem structures as the priors while maintaining efficient computation. It also has been investigated in the contextual multi-arm bandits (CMABs), which can be regarded as MDPs with a single state [4, 13, 16]. Thus, TS is a preferable way to balance exploration and exploitation than other heuristic methods.

However, posterior distributions are usually intractable except for some simple toy problems. Consequently, many methods such as variational inference and MCMC can be used for posterior approximation [3, 33, 7]. In this paper, we propose to use continuous-time flows (CTFs) [5], for approximately sampling the unknown policy in the deep soft Q-learning framework [11]. Specifically, a generator (*sampling network*) is learned to transform a simple distribution (*e.g.*, $\mathcal{N}(0, I)$) to a stochastic policy. In contrast to most methods on Bayesian RL [8, 12], our method can effectively represent complex multimodal policies, while enabling efficient sampling. The technique adopted for learning the sampling network relies on the development of CTFs in [6], which provides the first method to directly solve a sampling problem via stochastic optimization. More details are described in Section 3.

2 Preliminaries

2.1 Thompson Sampling

Thompson sampling [29] is a popular method to solve sequential decision problems [17, 25, 21]. It approximates the posterior of the policy parameters in an online manner. At each step, Thompson sampling (i) first draws a parameter sample, then (ii) picks the action by maximizing the expected reward over current step, (iii) collects data samples after observing the reward, and add it into experience pool, and (iv) updates posterior of the policy.

Thompson sampling is a natural way to utilize model uncertainty in sequential decision problems. Greater uncertainty on the weights typically introduces more variability into a decision made by a policy network [15], naturally leading the policy to explore. As more data are observed, the uncertainty decreases, allowing the decisions made by a policy to become more deterministic as the environment is better understood (exploitation when the policy becomes more confident).

In the Bayesian Q-Learning setting, uncertainty in the parameters of Q-value can be translated into uncertainty in the *predicted actions*, *i.e.*, instead of learning a distribution over parameters, we can directly learn a conditional distribution characterized by a stochastic policy. This setting is adopted in this paper, where a stochastic policy is learned implicitly in the soft deep Q-learning framework by CTFs.

2.2 Reinforcement Learning

The sequential decision-making procedure is formulated as MDP, which defines a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P_s, P_r, r, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $r \in \mathbb{R}$ is the reward, P_s and P_r are the unknown environment distributions to draw the next state and reward, respectively. At each time step t, given the current state $\boldsymbol{s}_t \in \mathcal{S}$, an agent can take an action $\boldsymbol{a}_t \in \mathcal{A}$ based on a policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$, a mapping from a state to an action; the state $\boldsymbol{s}_t$ then transits to a new state $\boldsymbol{s}_{t+1} \sim P_s(\boldsymbol{s}_t, \boldsymbol{a}_t)$, and the agent receives a reward $r_t \sim P_r(\boldsymbol{s}_t, \boldsymbol{a}_t)$ from the environment. The goal of RL is to find the optimal policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ to maximize the expected reward:

$$J(\pi) = \sum_t \gamma^t \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t) \sim \rho_\pi} r_t \tag{1}$$

where $\rho_\pi(\boldsymbol{s}_t, \boldsymbol{a}_t)$ denotes the induced state and state-action marginals of the trajectory distribution, and γ is the discount factor ensuring that the sum of expected rewards is finite. To incorporate prior domain knowledge for the policy, we restrict the optimal policy to be close to a prior distribution $p(\boldsymbol{a})$. Consequently, the objective (1) is formulated as:

$$\pi^* = \arg\max_\pi \sum_t \gamma^t \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t) \sim \rho_\pi} \left[r_t - \alpha \mathsf{KL}(\pi(\cdot|\boldsymbol{s}_t) \| p(\cdot)) \right] \ , \tag{2}$$

If we use an improper (uninformative) prior p, and the KL term is simplified to the entropy $\mathcal{H}(\pi(\cdot|\boldsymbol{s}_t))$, resulting in the maximum entropy reinforcement learning framework (MERL). α is the temperature parameter determining the relative importance of the entropy and reward. Specifically, the MERL augments the reward with an entropy term, such that the optimal policy aims to maximize its entropy as well at each time. The optimal policy is solved by the following optimization problem:

$$\pi^*_{\text{ME}} = \arg\max_\pi \sum_t \gamma^t \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t) \sim \rho_\pi} \left[r_t + \alpha \mathcal{H}(\pi(\cdot|\boldsymbol{s}_t)) \right] \ . \tag{3}$$

2.3 Soft Q-Learning

The MERL framework essentially leads to the Q-learning framework with an entropy-regularized Q-value, called soft Q-learning*i.e.*, *soft* versions of the value function $V_s(\boldsymbol{s}_t)$ and Q-functions $Q(\boldsymbol{s}_t, \boldsymbol{a}_t)$ are defined. Specifically, we have:

Lemma 2.1 ([11]) *The optimal policy for the soft Q-Learning* (3) *is given by*

$$\pi^*_{ME}(\boldsymbol{a}_t|\boldsymbol{s}_t) \propto \exp\left(\frac{1}{\alpha}\left(Q^*_s(\boldsymbol{s}_t, \boldsymbol{a}_t) - V^*_s(\boldsymbol{s}_t)\right)\right) , \tag{4}$$

where the optimal soft Q-function $Q^*_s(\boldsymbol{s}_t, \boldsymbol{a}_t)$ and soft value function $V^*_s(\boldsymbol{s}_t)$ are defined as follows, respectively.

$$Q^*_s(\boldsymbol{s}_t, \boldsymbol{a}_t) = r_t + \mathbb{E}_{(\boldsymbol{s}_{t+1},\cdots)\sim\rho_\pi}\left[\sum_{l=1}^{\infty}\gamma^l(r_{t+1} + \alpha\mathcal{H}(\pi^*_{ME}(\cdot|\boldsymbol{s}_{t+1})))\right] , \tag{5}$$

$$V^*_s(\boldsymbol{s}_t) = \alpha\log\int_{\mathcal{A}}\exp\left(\frac{1}{\alpha}Q^*_s(\boldsymbol{s}_t, \boldsymbol{a}')\right)\mathrm{d}\boldsymbol{a}' . \tag{6}$$

The soft value function $V^{\boldsymbol{\theta}}_s$ in terms of an expectation can be expressed via importance sampling: $V^{\boldsymbol{\theta}}_s(\boldsymbol{s}_{t+1}) = \alpha\log\mathbb{E}_{q_{\boldsymbol{a}'}}\left[\frac{\exp(\frac{1}{\alpha}Q^{\boldsymbol{\theta}}_s(\boldsymbol{s}_{t+1},\boldsymbol{a}'))}{q_{\boldsymbol{a}'}(\boldsymbol{a}')}\right]$, where $q_{\boldsymbol{a}'}$ can be an arbitrary distribution over the action space. Lemma 2.1 suggests a natural way to parameterize the optimal soft Q-function with a deep neural network (DNN), denoted as $\hat{Q}^{\boldsymbol{\theta}}_s(\boldsymbol{s}, \boldsymbol{a}_t)$ and parameterized by $\boldsymbol{\theta}$ [11]. The soft Q-iteration is equivalently to minimize the following objective:

$$J_Q(\boldsymbol{\theta}) \triangleq \mathbb{E}_{\boldsymbol{s}_t\sim q_{\boldsymbol{s}_t},\boldsymbol{a}_t\sim q_{\boldsymbol{a}_t}}\left[\frac{1}{2}\left(\hat{Q}^{\bar{\boldsymbol{\theta}}}_s(\boldsymbol{s}_t, \boldsymbol{a}_t) - Q^{\boldsymbol{\theta}}_s(\boldsymbol{s}_t, \boldsymbol{a}_t)\right)^2\right] , \tag{7}$$

where $q_{\boldsymbol{s}_t}$ and $q_{\boldsymbol{a}_t}$ can be arbitrary distributions with supports on $\mathcal{S}$ and $\mathcal{A}$, respectively; $\hat{Q}^{\bar{\boldsymbol{\theta}}}_s(\boldsymbol{s}_t, \boldsymbol{a}_t) = r_t + \gamma\mathbb{E}_{\boldsymbol{s}_{t+1}\sim\rho_\pi}[V^{\bar{\boldsymbol{\theta}}}_s(\boldsymbol{s}_{t+1})]$ is the target Q-value.

Following [11], real samples from current policy are used to estimate $q_{\boldsymbol{s}'}$ and $q_{\boldsymbol{a}'}$. Further, it is difficult to sample from $\pi^*_{\mathrm{ME}}(\boldsymbol{a}_t|\boldsymbol{s}_t)$ directly. Consequently, [11] propose to use an implicit generator $f^\phi(\xi; \boldsymbol{s})$ that takes state $\boldsymbol{s}$ and random noise ξ as input, and outputs an action $\boldsymbol{a}$ whose distribution attempts to approach $\pi^*_{\mathrm{ME}}(\boldsymbol{a}_t|\boldsymbol{s}_t)$. In their method, they use amortized Stein variational gradient descent (SVGD) to learn the parameter ϕ of the generator $f^\phi(\xi; \boldsymbol{s})$.

In this paper, instead of adopting SVGD, we investigate an alternative method CTFs [5] to learn the sampling network $f^\phi(\xi; \boldsymbol{s})$ to approximate the optimal stochastic policy. CTFs appear to endow advantages from both theoretical and application perspectives, as will be discussed later. Importantly, instead of adopting the amortized learning idea for CTFs, we propose a novel algorithm to directly optimize $f^\phi(\xi; \boldsymbol{s})$.

2.4 Continuous-Time Flows

Assume the action space to be $\mathbb{R}^L$. We consider a flow on $\mathbb{R}^L$, defined as the mapping $\mathcal{T} : \mathbb{R}^L \times \mathbb{R} \to \mathbb{R}^L$ such that we have $\mathcal{T}(\boldsymbol{a}, 0) = \boldsymbol{a}$ and $\mathcal{T}(\mathcal{T}(\boldsymbol{a}, t_1), t_2) = \mathcal{T}(\boldsymbol{a}, t_1 + t_2)$, for all $\boldsymbol{a} \in \mathbb{R}^L$ and $t_1, t_2 \in \mathbb{R}$. A typical example of this family is defined as $\mathcal{T}(\boldsymbol{a}, t) = \boldsymbol{a}_t$, where $\boldsymbol{a}_t$ is driven by a diffusion of the form:

$$\mathrm{d}\boldsymbol{a}_t = F_{\boldsymbol{s}}(\boldsymbol{a}_t)\mathrm{d}t + \sigma_{\boldsymbol{s}}(\boldsymbol{a}_t)\mathrm{d}\mathcal{W} . \tag{8}$$

Here $F_{\boldsymbol{s}} : \mathbb{R}^L \to \mathbb{R}^L$, $\sigma_{\boldsymbol{s}} : \mathbb{R}^{L\times L} \to \mathbb{R}^L$ are called the drift term and diffusion term, respectively; they both depend on some parameter $\boldsymbol{s}$ (in our case $\boldsymbol{s}$ represents the state); $\mathcal{W}$ is the standard L-dimensional Brownian motion. In our setting, we seek to make the stationary policy distribution of $\boldsymbol{a}_t$ approach the optimal policy, $\pi^*_{\mathrm{ME}}(\boldsymbol{a}_t|\boldsymbol{s}_t)$. One solution for this is to set $F_{\boldsymbol{s}}(\boldsymbol{a}_t) = \frac{1}{2}\nabla_{\boldsymbol{a}}\log\pi^*_{\mathrm{ME}}(\boldsymbol{a}_t|\boldsymbol{s}_t)$ and $\sigma_{\boldsymbol{s}}(\boldsymbol{a}_t) = \mathbf{I}_L$ with $\mathbf{I}_L$ the $L \times L$ identity matrix. The resulting diffusion is called Langevin dynamics [31]. Denoting the policy distribution of $\boldsymbol{a}_t$ as ρ_t, it is proved [24] that ρ_t is characterized by the Fokker-Planck (FP) equation:

$$\frac{\partial\rho_t}{\partial t} = -\nabla_{\boldsymbol{z}}\cdot(\rho_t F_{\boldsymbol{s}}(\boldsymbol{a}_t)) + \nabla_{\boldsymbol{a}}\nabla_{\boldsymbol{a}}:\left(\rho_t\sigma_{\boldsymbol{s}}(\boldsymbol{a}_t)\sigma^\top_{\boldsymbol{s}}(\boldsymbol{a}_t)\right) , \tag{9}$$

where $\boldsymbol{a}\cdot\boldsymbol{b} \triangleq \boldsymbol{a}^\top\boldsymbol{b}$ for vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, $\mathbf{A}:\mathbf{B} \triangleq \mathrm{trace}(\boldsymbol{A}^\top\mathbf{B})$ for matrices $\mathbf{A}$ and $\mathbf{B}$.

Compared with SVGD SVGD has been explained as gradient flows whose gradient operator is defined on the RKHS [18]; whereas CTF are flows with the flow operator defined on the $\mathcal{L}_2$ space. Since RKHS is smaller than $\mathcal{L}_2$, CTFs can obtain better asymptotic properties than SVGD in theory [18]. Introducing CTF naturally brings some nice properties specially for soft Q-learning; however, it also brings challenges in the learning process, as detailed in section 3.

3 Soft Q-Learning via CTFs

We describe our method in the soft Q-learning setting defined above. Our goal is to learn a stochastic policy $f^{\phi}(\xi; \boldsymbol{s})$ such that samples $\{\boldsymbol{a}^{(\ell)}\}_{\ell=1}^{M}$ generated by $f^{\phi}(\xi; \boldsymbol{s})$ approach the optimal policy $\pi^{*}_{\mathrm{ME}}(\boldsymbol{a}_t|\boldsymbol{s}_t)$. We first introduce lemma 3.1 from [5], viewing CTFs from an optimization perspective.

Lemma 3.1 *Assume that $\pi^{*}_{ME}(\boldsymbol{a}_t|\boldsymbol{s}_t) \leq C_1$ is infinitely differentiable, and $\|\nabla_{\boldsymbol{a}} \log \pi^{*}_{ME}(\boldsymbol{a}_t|\boldsymbol{s}_t)\| \leq C_2 \left(1 + C_1 - \log \pi^{*}_{ME}(\boldsymbol{a}_t|\boldsymbol{s}_t)\right) (\forall \boldsymbol{a}, \boldsymbol{s})$ for some constants $\{C_1, C_2\}$. Let $T = hK$ with K being an integer (interpreted as number of iterations) and h interpreted as stepsize, $\tilde{\rho}_0$ is an arbitrary distribution with same support as $\pi^{*}_{ME}(\boldsymbol{a}_t|\boldsymbol{s}_t)$, and $\{\tilde{\rho}_k\}_{k=1}^{K}$ be the solution of the functional optimization problem:*

$$\tilde{\rho}_k = \arg\min_{\rho \in \mathcal{L}_2} \left(\rho \| \pi^{*}_{ME}(\cdot|\boldsymbol{s}_t)\right) + \frac{1}{2h} W_2^2\left(\tilde{\rho}_{k-1}, \rho\right) , \tag{10}$$

where $W_2^2\left(\mu_1, \mu_2\right) \triangleq \inf_{p \in \mathcal{P}(\mu_1, \mu_2)} \int \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \, p(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{y})$, $W_2\left(\mu_1, \mu_2\right)$ is the 2nd-order Wasserstein distance, with $\mathcal{P}(\mu_1, \mu_2)$ being the space of joint distributions of $\{\mu_1, \mu_2\}$. $\mathcal{L}_2$ is the space of probability distributions with finite 2nd-order moment. Then $\tilde{\rho}_K$ converges to ρ_T in the limit of $h \to 0$, i.e., $\lim_{h \to 0} \tilde{\rho}_K = \rho_T$, where ρ_T is the solution of the FP equation (9) at time T.

Lemma 3.1 reveals an interesting way to compute ρ_T via a sequence of functional optimization problems. By comparing it with the objective of SVGD, which minimizes the KL-divergence between ρ_k and $\pi^{*}_{\mathrm{ME}}(\boldsymbol{a}_t, \boldsymbol{s}_t)$, at each sub-optimization-problem in Lemma 3.1, it minimizes the same KL-divergence, plus a regularization term as the Wasserstein distance between $\tilde{\rho}_{k-1}$ and $\tilde{\rho}_k$. The extra Wasserstein-distance term arises naturally due to the fact that the Langevin diffusion can be explained as a gradient flow equipped with a geometric associated with the Wasserstein distance [23]. From another point of view, it is known that the Wasserstein distance is a better metric for probability distributions than the KL-divergence, especially in the case of non-overlapping domains [1, 2].

Binding the generator to the FP equation Denote the distribution induced by the generator $f^{\phi}(\xi; \boldsymbol{s})$ as $\pi^{\phi}(\cdot|\boldsymbol{s})$, where ξ are samples of some simple distribution. Instead of adopting the amortization idea to update ϕ with CTF as in [5], we directly match $\pi^{\phi}(\cdot|\boldsymbol{s}_t)$ to $\tilde{\rho}_t$ defined in Lemma 3.1, *i.e.*, we want $\pi^{\phi}(\cdot|\boldsymbol{s})$ to evolve as the FP equation (9). According to Lemma 3.1, this corresponds to an optimization problem with objective

$$J_{\pi}(\phi; \boldsymbol{s}_t) = \underset{\pi^{\phi}(\cdot|\boldsymbol{s}) \in \mathcal{K}_2}{\mathsf{KL}} \left(\pi^{\phi}(\cdot|\boldsymbol{s}) \| \pi^{*}_{\mathrm{ME}}(\cdot|\boldsymbol{s}_t)\right) + \frac{1}{2h} W_2^2\left(\pi^{\phi_{\mathrm{old}}}(\cdot|\boldsymbol{s}), \pi^{\phi}(\cdot|\boldsymbol{s})\right)$$

for iteration t, where $\mathcal{K}_2$ is the space of probability distributions with finite second moment.

3.1 Optimization

We can apply SGD to update the policy parameter. Specifically, we need to calculate:

$$\frac{\partial J_{\pi}(\phi; \boldsymbol{s}_t)}{\partial \phi} = \frac{\partial \mathsf{KL}\left(\pi^{\phi}(\cdot|\boldsymbol{s}) \| \pi^{*}_{\mathrm{ME}}(\cdot|\boldsymbol{s}_t)\right)}{\partial \phi} + \frac{1}{2h} \frac{\partial W_2^2\left(\pi^{\phi_{\mathrm{old}}}(\cdot|\boldsymbol{s}), \pi^{\phi}(\cdot|\boldsymbol{s})\right)}{\partial \phi} . \tag{11}$$

When $\pi^{\phi}(\cdot|\boldsymbol{s})$ lies on the RKHS defined in SVGD [19], the first term in the RHS of (11) is readily calculated in [11]. In our case, $\pi^{\phi}(\cdot|\boldsymbol{s})$ is restricted to a larger space of $\mathcal{L}_2$. Following [6], we approximate this by adding noise in the gradient calculated using SVGD, resulting in

$$\frac{\partial \mathsf{KL}\left(\pi^{\phi}(\cdot|\boldsymbol{s}) \| \pi^{*}_{\mathrm{ME}}(\cdot|\boldsymbol{s}_t)\right)}{\partial \phi} \approx \mathbb{E}_{\xi}\left[\left(\Delta f^{\phi}(\xi; \boldsymbol{s}_t) + \zeta\right) \frac{\partial f^{\phi}(\xi; \boldsymbol{s}_t)}{\partial \phi}\right], \; \zeta \sim \mathcal{N}\left(\mathbf{0}, \delta \mathbf{I}\right) , \tag{12}$$

where $\Delta f^{\phi}(\xi; \boldsymbol{s}_t) \triangleq \mathbb{E}_{\boldsymbol{a}_t \sim \pi^{\phi}} \left[\kappa(\boldsymbol{a}_t, f^{\phi}(\cdot; \boldsymbol{s}_t)) \nabla_{\boldsymbol{a}'} Q_s^{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}') \,|_{\boldsymbol{a}'=\boldsymbol{a}} + \alpha \nabla_{\boldsymbol{a}'} \kappa(\boldsymbol{a}', f^{\phi}(\cdot; \boldsymbol{s}_t)) \,|_{\boldsymbol{a}'=\boldsymbol{a}}\right]$, with κ being a kernel function; δ is a user-control parameter, controlling the variance of injected noises. δ is typically decreasing in the algorithm to ensure convergence. Note that this technique resembles stochastic gradient descent, where full gradients are replaced with noisy gradients, thus its convergence can still be guaranteed.

In order to calculate the second term on the RHS of (11), we adapt results from optimal transport theory [30, 10] to rewrite $W_2^2\left(\pi^{\phi_{\text{old}}}(\cdot|\boldsymbol{s}), \pi^{\phi}(\cdot|\boldsymbol{s})\right)$ as

$$\begin{aligned} & W_2^2\left(\pi^{\phi_{\text{old}}}(\cdot|\boldsymbol{s}), \pi^{\phi}(\cdot|\boldsymbol{s})\right) \\ =& \mathbb{E}_{\boldsymbol{a}' \sim \pi^{\phi_{\text{old}}}} \|\boldsymbol{a}'\|^2 + \mathbb{E}_{\boldsymbol{a} \sim \pi^{\phi}} \|\boldsymbol{a}\|^2 + 2 \sup_{\psi(\cdot) \text{ convex}} -\mathbb{E}_{\boldsymbol{a}' \sim \pi^{\phi_{\text{old}}}} [\psi(\boldsymbol{a}')] - \mathbb{E}_{\boldsymbol{a}_t \sim \pi^{\phi}} [\psi^*(\boldsymbol{a})] \ , \end{aligned} \tag{13}$$

where $\psi^*(\boldsymbol{a}) \triangleq \sup_{\boldsymbol{v}} \left(\boldsymbol{v}^T \boldsymbol{a} - \psi(\boldsymbol{v})\right)$ is the convex-conjugate of the function ψ. To optimize $W_2^2\left(\pi^{\phi_{\text{old}}}(\cdot|\boldsymbol{s}), \pi^{\phi}(\cdot|\boldsymbol{s})\right)$, we can construct a GAN-like structure, with discriminator defined as the "sup" part in (13), except that the discriminator is required to be convex. For ease of calculation, following our ongoing work [6], we restrict ψ to be a quadratic function, *e.g.*, $\psi(\boldsymbol{a}) \triangleq \frac{1}{2}\boldsymbol{a}^T \mathbf{A} \boldsymbol{a} + \boldsymbol{b}^T \boldsymbol{a}$ with parameters $\{\mathbf{A}, \boldsymbol{b}\}$. After simplification, we obtain a simple form of

$$W_2^2\left(\pi^{\phi_{\text{old}}}(\cdot|\boldsymbol{s}), \pi^{\phi}(\cdot|\boldsymbol{s})\right) = \mathbb{E}_{\boldsymbol{a}' \sim \pi^{\phi_{\text{old}}}, \boldsymbol{a} \sim \pi^{\phi}} \left(\boldsymbol{a} - \boldsymbol{a}'\right)^2 \ .$$

Then we obtain the gradient of Wasserstein-2 trust-region $\Delta W_2^2 = \frac{1}{h}\left(f^{\phi}(\xi; \boldsymbol{s}_t) - f^{\phi_{\text{old}}}(\xi; \boldsymbol{s}_t)\right)$. Since $\boldsymbol{a} \sim \pi^{\phi}$ is equivalent to $\xi \sim q_0(\xi), \boldsymbol{a} = f^{\phi}(\xi; \boldsymbol{s}_t)$ with $q_0(\cdot)$ a simple noise distribution, the gradient can be approximated as

$$\frac{\partial J_{\pi}(\phi; \boldsymbol{s}_t)}{\partial \phi} \approx \mathbb{E}_{\xi}\left[\left(\Delta f^{\phi}(\xi; \boldsymbol{s}_t) + \zeta + \Delta W_2^2\right) \frac{\partial f^{\phi}(\xi; \boldsymbol{s}_t)}{\partial \phi}\right] \tag{14}$$

This is essentially SVGD with noisy gradient and momentum algorithm, a nice connection between SVGD and CTF [6]. Since SGD with momentum has show faster convergence speed than standard SGD, CTF is also expected to be faster than SVGD. We also note that this is the first time the FP equation of an CTF can be solved efficiently via optimization, as it was believed that the optimal decreased direction of the FP equation is generally infeasible [18].

Connections with Thompson sampling Different from traditional TS, which draws samples from the posterior distribution of policy parameter, we learn to draw samples from posteriors of stochastic policies, *i.e.*, the posterior is encoded in an implicit distribution so that parameter uncertainty in traditional TS is translated into uncertainty in action samples. The proposed method has the following advantages: *i*) exploration is achieved by similar ideas from TS. *ii*) The intractable posterior sampling for complex policies in TS is overcome by introducing an implicit policy network that does not endow an explicit distribution form, forming a larger candidate distribution space. *iii*) The sampling network directly approximate the stochastic policy in action space instead of uncertainty in parameter space, achieving more efficient exploration.

Connections with Trust Region Policy Optimization (TRPO) Recent work [26] has illustrated the equivalence between soft Q-learning and policy gradient. We can also show the connection between the proposed method and TRPO [27]. In TRPO, an objective function is maximized subjected to a constraint on the size of policy update. Specifically,

$$\max_{\boldsymbol{\phi}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi^{\phi}(\cdot|\boldsymbol{s})}{\pi^{\phi_{\text{old}}}(\cdot|\boldsymbol{s})}\hat{A}_t\right] \tag{15}$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t\left[\text{KL}[\pi^{\phi}(\cdot|\boldsymbol{s}), \pi^{\phi_{\text{old}}}(\cdot|\boldsymbol{s})]\right] \leq \delta \tag{16}$$

Here, π_{ϕ} is a stochastic policy; ϕ_{old} is the vector of policy parameters before the update; $\hat{A}_t$ is an estimator of the advantage function at timestep t. The theory of TRPO suggests using a penalty instead of a constraint, *i.e.*, solving the unconstrained optimization problem,

$$\max_{\boldsymbol{\phi}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi^{\phi}(\cdot|\boldsymbol{s})}{\pi^{\phi_{\text{old}}}(\cdot|\boldsymbol{s})}\hat{A}_t - \beta \text{KL}[\pi^{\phi}(\cdot|\boldsymbol{s}), \pi^{\phi_{\text{old}}}(\cdot|\boldsymbol{s})]\right] \tag{17}$$

In our proposed framework, the Wasserstein distance between $\pi^{\phi}(\cdot|\boldsymbol{s})$ and $\pi^{\phi_{\text{old}}}(\cdot|\boldsymbol{s})$, a weaker metric than the KL divergence, constrains the update of a policy on a manifold endowed with the Wasserstein metric, and potentially leads to more robust solutions. This is evidenced by the development of Wasserstein GAN [2]. As a result, our framework can be regarded as a trust region based counterpart for solving the soft Q-learning problem [11].

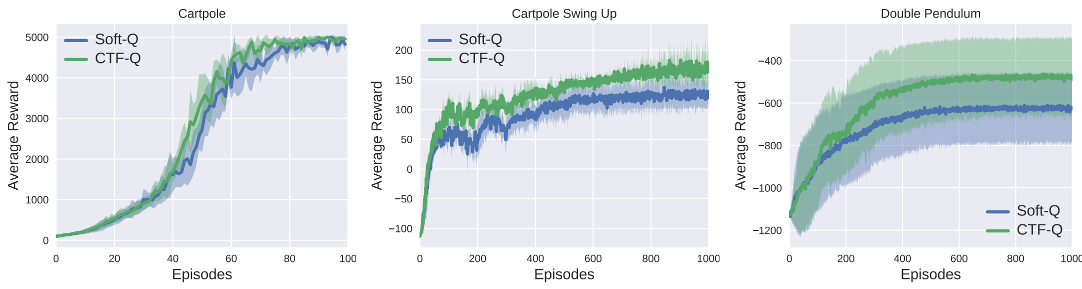
4 Experiments

To demonstrate the effectiveness of our proposed CTF soft Q-Learning (CTF-Q for short), we first conduct a toy experiments following [11]. We then test our method on classic continuous control problems: CartPole ($\mathcal{S} \subseteq \mathbb{R}^4$, $\mathcal{A} \subseteq \mathbb{R}^1$), CartPoleSwingup ($\mathcal{S} \subseteq \mathbb{R}^4$, $\mathcal{A} \subseteq \mathbb{R}^1$), DoublePendulum ($\mathcal{S} \subseteq \mathbb{R}^6$, $\mathcal{A} \subseteq \mathbb{R}^1$). The superiority of CTF-Q is further demonstrated on the MuJoCo continuous control tasks: Walker2D, Swimmer, and Hopper. All experiments are based on the OpenAI `rllab` toolkits [9].

We compare CTF-Q with the recently proposed soft Q-Learning (Soft-Q for short) algorithms, as it has shown superiority compared with other methods such as DDPG [28], A3C [20], TRPO [27], etc. The ADAM [14] optimizer is employed when needed. We use a RBF kernel $\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}') = \exp(-\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2^2/h)$ when calculating (12), with the bandwidth set to $h = \texttt{med}^2/\log M$. Here `med` is the median of the pairwise distance between particles. We set the hyperparameter, temperature $\alpha = 1$. All experiments are conducted on a single TITAN X GPU. Performance is measured through the average return (not including the intrinsic rewards) at each episode.

4.1 Classic Control Problems

For classic control problems, we adopt similar settings as [12, 32], where the Q-function is parameterized by a two-layer (25-16 hidden units) NN with `tanh` as the activation function for the complex Cartpole Swingup and Double Pendulum tasks. The agents are trained for 100 episodes. For the easier task Cartpole, a one-layer (32 hidden units) NN is used, and the agents are trained for 50 episodes. The maximal length of horizon is set to 10000. We use a batch size of 64, with $M = 16$ and $\alpha = 1$. We implement sampling networks with the same size as the Q-network. Figure 1 illustrates the learning curves obtained with CTF-Q and SVPG. Compared with SVPG, CTF-Q appears to converge faster and is more stable in the three classic continuous control tasks with lower variance.

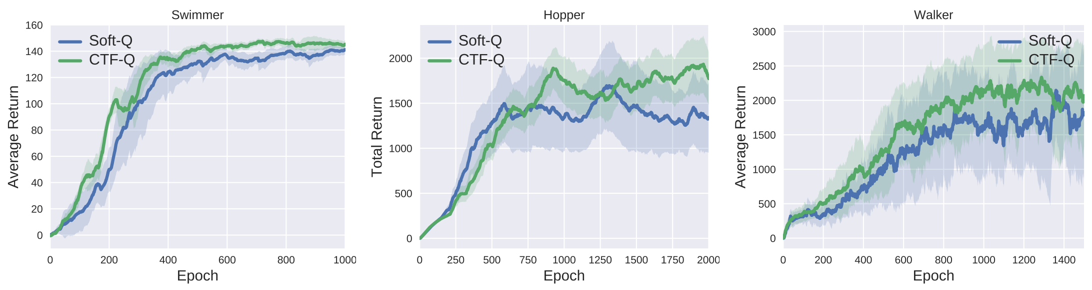


(a) Cartpole Swing Up (b) Double Pendulum (c) CartPole

Figure 1: Average return over episodes by SVPG and CTF-Q (IP-WGF) on classic continuous control tasks.

4.2 MuJoCo Environment

We further apply our proposed method on MuJuCo tasks, where the Q-function is parameterized by a two-layer (128-128 hidden units) NN with `tanh` as the activation function for all tasks. All agents are trained for 150 episodes. The maximal length of horizon is set to 1000. Again we use a batch size of 64, with $M = 16$, $\alpha = 1$. We implement sampling networks with the same size as the Q-network. Figure 2 illustrates the learning curves of all the three tasks. Similarly, faster convergence and larger rewards are obtained by our proposed CTF-Q method.



(a) Swimmer (b) Hopper (c) Walker

Figure 2: Learning curves of Soft-Q and CTF-Q on MuJoCo tasks.

5 Conclusion

We present a method for learning stochastic energy-based policies by implicit sampling networks learned via a novel technique based on CTF. Our approach is a type of soft Q-learning method, with powerful exploration ability. Sampling from the sampling network can be viewed as a type of Thompson sampling. Our experimental results show that our method can effectively explore multi-modal landscapes of the policy networks. Experiments on complex continuous control of simulated robots such as walker, swimmer and hopper show promising results compared to the recently proposed soft Q-learning. To sum up, our method demonstrates that stochastic policies with improved exploration by TS-style methods provide a a promising way to solve Bayesian deep RL problems.

References

[1] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. In *ICML*, 2017.

[3] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks, 2015.

[4] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In *NIPS*, 2011.

[5] C. Chen, C. Li, L. Chen, W. Wang, Y. Pu, and L. Carin. Continuous-time flows for deep generative models, 2017.

[6] C. Chen and R. Zhang. Particle optimization in stochastic gradient MCMC. *arXiv:1711.10927*, 2017.

[7] C. Chen, N. Ding, C. Li, Y. Zhang, and L. Carin. Stochastic gradient mcmc with stale gradients. In *NIPS*, 2016.

[8] R. Dearden, N. Friedman, and S. Russell. Bayesian q-learning. In *AAAI/IAAI*, 1998.

[9] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *ICML*, 2016.

[10] S. Feizi, C. Suh, F. Xia, and D. Tse. Understanding GANs: the LQG setting. In *arXiv:1710.10793*, 2017.

[11] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *ICML*, 2017.

[12] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. Vime: Variational information maximizing exploration. In *NIPS*, 2016.

[13] J. Kawale, H. H. Bui, B. Kveton, L. Tran-Thanh, and S. Chawla. Efficient thompson sampling for online matrix-factorization recommendation. In *NIPS*, 2015.

[14] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[15] J. Z. Kolter and A. Y. Ng. Near-bayesian exploration in polynomial time. In *ICML*, 2009.

[16] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, 2010.

[17] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*, 2011.

[18] Q. Liu. Stein variational gradient descent as gradient flow. In *NIPS*, 2017.

[19] Q. Liu and D. Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *NIPS*, 2016.

[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

[21] I. Osband, D. Russo, and B. Van Roy. (more) efficient reinforcement learning via posterior sampling. In *NIPS*, 2013.

[22] I. Osband and B. Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *ICML*, 2017.

[23] F. Otto. Dynamics of Labyrinthine pattern formation in magnetic fluids: A mean-field theory. *Arch. Rational Mech. Anal.*, 1998.

[24] H. Risken. *The Fokker-Planck equation*. Springer-Verlag, New York, 1989.

[25] D. Russo and B. Van Roy. An information-theoretic analysis of thompson sampling. *JMLR*, 2016.

[26] J. Schulman, P. Abbeel, and X. Chen. Equivalence between policy gradients and soft q-learning. *arXiv:1704.06440*, 2017.

[27] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *ICML*, 2015.

[28] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

[29] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933.

[30] C. Villani. *Optimal transport: old and new*. Springer Science & Business Media, 2008.

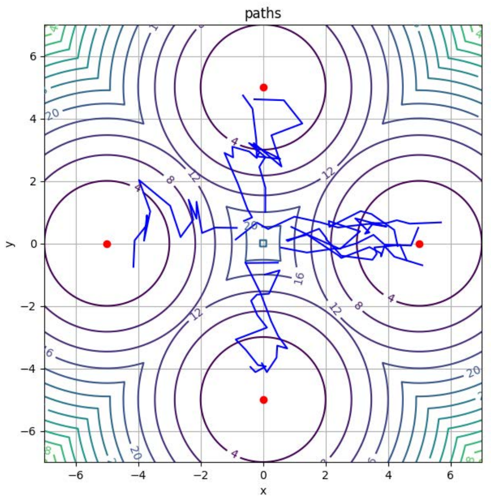
[31] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *ICML*, 2011.

[32] R. Zhang, C. Li, C. Chen, and L. Carin. Learning structural weight uncertainty for sequential decision-making, 2018.

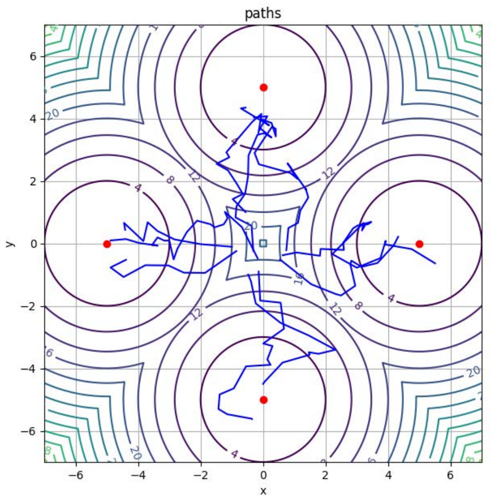
[33] Y. Zhang, X. Wang, C. Chen, R. Henao, K. Fan, and L. Carin. Towards unifying hamiltonian monte carlo and slice sampling. In *NIPS*, 2016.

A Learning Multi-Modal Policies for Exploration

Following [11], we first validate CTF can correctly draw samples from energy-based policies by learning a sampling network that represents multi-modal policies. In this environment, a 2D-point-mass agent is trying to reach one of four symmetrically placed goals. The reward is defined as a mixture of Gaussians, with means placed at the goal positions. Figure 3 illustrates the policies obtained by CTF-Q and Soft-Q for the same number of updates. It is seen that CTF-Q explores the modes better than Soft-Q. Specifically, trajectories of CTF-Q uniformly reach the four target goals, while more trajectories reach the goal on the right in Soft-Q, indicating that CTF-Q endows better exploration ability than Soft-Q.



(a) Soft Q



(b) CTF Q

Figure 3: Illustration of the 2D multi-goal environment. The x and y axes correspond to 2D positions (states). The agent is initialized at the origin. The goals are depicted as red dots, and the level curves show the reward. The left plot represents the policy learned by Soft-Q, and the right plot represents the policy learned by CTF-Q.

B Algorithm Overview

Algorithm 1 CTF Q-Learning

```
Require: $\mathcal{D} = \emptyset$; initialize $\boldsymbol{\theta}, \boldsymbol{\phi} \sim$ some (prior) distribution. Target parameters: $\overline{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$, $\overline{\boldsymbol{\phi}} \leftarrow \boldsymbol{\phi}$
 1: for each epoch do
 2:     for each t do
 3:         % Collect expereince
 4:         Sample an action for $\boldsymbol{s}_t$ using $f^{\phi}$: $\boldsymbol{a}_t \leftarrow f^{\phi}(\boldsymbol{\xi}; \boldsymbol{s}_t)$, where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
 5:         Sample next state from the environment: $\boldsymbol{s}_{t+1} \sim p_{\boldsymbol{s}}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$
 6:         Save the new experience in the replay memory: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{s}_t, \boldsymbol{a}_t, r(\boldsymbol{s}_t, \boldsymbol{a}_t), \boldsymbol{s}_{t+1}\}$
 7:         % Sample a minibatch from the replay memory
 8:         $\{(\boldsymbol{s}_t^{(i)}, \boldsymbol{a}_t^{(i)}, r_t^{(i)}, \boldsymbol{s}_{t+1}^{(i)})\}_{i=0}^{N} \sim \mathcal{D}$.
 9:         % Update Q function parameters
10:         Sample $\{\boldsymbol{a}^{(i,j)}\}_{j=0}^{M} \sim q_{\boldsymbol{a}'}$ for each $\boldsymbol{s}_{t+1}^{(i)}$.
11:         Compute empirical soft values $\hat{V}_{soft}^{\overline{\boldsymbol{\theta}}}(\boldsymbol{s}_{t+1}^{(i)})$ in (5)
12:         Compute empirical gradient $\hat{\nabla}_{\boldsymbol{\theta}} J_Q(\boldsymbol{\theta})$ of (7) and update $\boldsymbol{\theta}$ according to it using ADAM
13:         % Update policy
14:         Sample $\{\boldsymbol{\xi}^{(i,j)}\}_{j=0}^{M} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for each $\boldsymbol{s}_t^{(i)}$
15:         Compute actions $\boldsymbol{a}_t^{(i,j)} = f^{\boldsymbol{\phi}}(\boldsymbol{\xi}^{(i,j)}, \boldsymbol{s}_t^{(i)})$.
16:         Compute empirical estimate in (11), and update $\boldsymbol{\phi}$ according to it using ADAM
17:     end for
18:     if epoch mod update_interval = 0 then
19:         Update target parameters: $\overline{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$, $\overline{\boldsymbol{\phi}} \leftarrow \boldsymbol{\phi}$
20:     end if
21: end for
```