Policy Optimization as Wasserstein Gradient Flows

Ruiyi Zhang [1] **Changyou Chen** [2] **Chunyuan Li** [1] **Lawrence Carin** [1]

Abstract

Policy optimization is a core component of reinforcement learning (RL), and most existing RL methods directly optimize parameters of a policy based on maximizing the expected total reward, or its surrogate. Though often achieving encouraging empirical success, its underlying mathematical principle on *policy-distribution* optimization is unclear. We place policy optimization into the space of *probability measures*, and interpret it as Wasserstein gradient flows. On the probability-measure space, under specified circumstances, policy optimization becomes a convex problem in terms of distribution optimization. To make optimization feasible, we develop efficient algorithms by numerically solving the corresponding discrete gradient flows. Our technique is applicable to several RL settings, and is related to many state-of-the-art policy-optimization algorithms. Empirical results verify the effectiveness of our framework, often obtaining better performance compared to related algorithms.

1. Introduction

There is recent renewed interest in reinforcement learning (Sutton & Barto, 1998; Kaelbling et al., 1996), largely as a consequence of the success of deep reinforcement learning (deep RL) (Mnih et al., 2015; Li, 2017), which is applicable to complex environments and has obtained state-of-the-art performance on several challenging problems. In reinforcement learning an agent interacts with the environment, seeking to learn an optimal policy that yields the maximum expected reward during the interaction. Generally speaking, a policy defines a distribution over actions conditioned on the states. Learning an optimal policy corresponds to searching for an element (a conditional action distribution) on the space of distributions that yields the best expected feedback (reward) to the agent as it interacts sequentially with the environment.

A standard technique for policy learning is the policy-gradient (PG) method (Sutton et al., 2000). In PG, a policy is represented in terms of parameters, typically optimized by stochastic gradient descent (SGD) to maximize the expected total reward. A similar idea has been applied for learning deterministic policies, termed deterministic policy gradient (DPG) (Silver et al., 2014). Significant progress has been made on advancing policy learning since introduction of deep learning techniques. As examples, the deep deterministic policy gradient (DDPG) method combines DPG and Q-learning (Watkins & Dayan, 1992) to jointly learn a policy and a Q-function for continuous control problems (Lillicrap et al., 2016). Trust region policy optimization (TRPO) improves PG by preserving the monotonic-policy-improvement property (Schulman et al., 2015), implemented by imposing a trust-region constraint, defined as the Kullback-Leibler (KL) divergence between consecutive policies. Later, (Schulman et al., 2017b) proposed proximal policy optimization (PPO) to improve TRPO by optimizing a "surrogate" objective with an adaptive KL penalty and reward-clipping mechanism. Though obtaining encouraging empirical success, many of the aforementioned algorithms optimize parameters directly, and appear to lack a rigorous interpretation in terms of distribution optimization, *e.g.*, it is not mathematically clear how sequentially optimizing policy parameters based on an expected-total-reward objective corresponds to optimizing the *distribution* of policy itself.

In this paper we introduce gradient flows in the space of probability distributions, called *Wasserstein gradient flows* (WGF), and formulate policy optimization in RL as a WGF problem. Essentially, WGF induces a geometry structure (manifold) in the distribution space characterized by an *energy functional*. The length between elements on the manifold is defined by the second-order Wasserstein distance. Thus, searching for an optimal distribution corresponds to traveling along a gradient flow on the space until convergence. In the context of deep RL, the energy functional is characterized by the expected reward. Gradient flow corresponds to a sequence of policy distributions converging to an optimal policy during an iterative optimization procedure. From this perspective, convergence behavior of the optimization can be better understood.

[1]Duke University [2]SUNY at Buffalo. Correspondence to: Ruiyi Zhang <zhangry868@gmail.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

Traditional stochastic policies are limited by their simple representation ability, such as using multinomial (Mnih et al., 2015) or Gaussian policy distributions (Schulman et al., 2015). To overcome this issue, the proposed WGF-based stochastic policies employ general energy-based representations. To optimize the stochastic policy, we define WGFs for RL in two settings: *i*) indirect-policy learning, defined on a distribution space for *parameters*; *ii*) direct-policy learning, defined on a distribution space for *policy distributions*. These correspond to two variants of our algorithms. The original form of the WGF problem is hard to deal with, as it is generally infeasible to directly optimize over a distribution (an infinite-dimensional object). To overcome this issue, based on the Jordan-Kinderlehrer-Otto (JKO) method (Jordan et al., 1998), we propose a particle-based algorithm to approximate a continuous density function with particles, and derive the corresponding gradient formulas for particle updates. Our method is conceptually simple and practically efficient, which also provides a theoretically sound way to use trust-region algorithms for RL. Empirical experiments show improved performance over related reinforcement learning algorithms.

2. Preliminaries

We review concepts and numerical algorithms for gradient flows. We start from gradient flows on the Euclidean space, and then extend them on the space of probability measures.

2.1. Gradient flows on the Euclidean space

For a smooth function* $F : \mathbb{R}^d \to \mathbb{R}$, and a starting point $\mathbf{x}_0 \in \mathbb{R}^d$, the gradient flow of $F(\mathbf{x})$ is defined as the solution of the differential equation: $\frac{\mathrm{d}\,\mathbf{x}}{\mathrm{d}\tau} = -\nabla F(\mathbf{x}(\tau))$, for time $\tau > 0$ and initial condition $\mathbf{x}(0) = \mathbf{x}_0$. This is a standard Cauchy problem (Rulla, 1996), endowed with a unique solution if ∇F is Lipschitz continuous. When F is non-differentiable, the gradient is replaced with its subgradient, which gives a similar definition, omitted for simplicity.

Numerical solution An exact solution to the above gradient-flow problem is typically intractable. A standard numerical method, called the *Minimizing Movement Scheme* (MMS) (Gobbino, 1999), evolves $\mathbf{x}$ iteratively for small steps along the gradient of F at the current point. Denoting the current point as $\mathbf{x}_k$, the next point is $\mathbf{x}_{k+1} = \mathbf{x}_k - \nabla F(\mathbf{x}_{k+1})h$, with stepsize h. Note $\mathbf{x}_{k+1}$ is equivalent to solving optimization problem $\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} F(\mathbf{x}) + \frac{\|\,\mathbf{x}-\mathbf{x}_k\,\|_2^2}{2h}$, where $\|\cdot\|_2$ denotes the vector 2-norm. Convergence of the $\{\mathbf{x}_k\}$ sequence to the exact solution has been established (Ambrosio et al., 2005). Refer to Section A.1 of the Supplementary Material (SM) for details.

*We will focus on the convex case, since this is the case for many gradient flows on the space of probability measures, as detailed subsequently.

2.2. Gradient flows on the probability-measure space

By placing the optimization onto the space of probability measures, denoted $\mathcal{P}(\Omega)$ with $\Omega \subset \mathbb{R}^d$, we arrive at Wasserstein gradient flows. For a formal definition, we first endow a Riemannian geometry on $\mathcal{P}(\Omega)$. The geometry is characterized by the length between two elements (two distributions), defined by the 2nd-order Wasserstein distance:

$$W_2^2(\mu,\nu) \triangleq \inf_{\gamma}\left\{\int_{\Omega\times\Omega} \|\,\mathbf{x}-\mathbf{y}\,\|_2^2 \mathrm{d}\gamma(\mathbf{x},\mathbf{y}) : \gamma\in\Gamma(\mu,\nu)\right\},$$

where $\Gamma(\mu,\nu)$ is the set of joint distributions over $(\mathbf{x},\mathbf{y})$ such that the two marginals equal μ and ν, respectively. This is an optimal-transport problem, where one wants to transform μ to ν with minimum cost (Villani, 2008). Thus the term $\|\,\mathbf{x}-\mathbf{y}\,\|_2^2$ represents the cost to transport $\mathbf{x}$ in μ to $\mathbf{y}$ in ν, and can be replaced by a general metric $c(\mathbf{x},\mathbf{y})$ in a metric space. If μ is absolutely continuous w.r.t. the Lebesgue measure, there is a unique optimal transport plan from μ to ν, *i.e.*, a mapping $T : \mathbb{R}^d \to \mathbb{R}^d$ pushing μ onto ν satisfying $T_{\#}\mu = \nu$. Here $T_{\#}\mu$ denotes the pushforward measure of μ. The Wasserstein distance is equivalently reformulated as

$$W_2^2(\mu,\nu) \triangleq \inf_{T}\left\{\int_{\Omega} c(\mathbf{x},T(\mathbf{x}))\mathrm{d}\mu(\mathbf{x})\right\}.$$

Let $\{\mu_\tau\}_{\tau\in[0,1]}$ be an absolutely continuous curve in $\mathcal{P}(\Omega)$ with finite second-order moments. Consider $W_2^2(\mu_\tau,\mu_{\tau+h})$. Motivated by the Euclidean-space case, if we define $\mathbf{v}_\tau(\mathbf{x}) \triangleq \lim_{h\to 0}\frac{T(\mathbf{x}_\tau)-\mathbf{x}_\tau}{h}$ as the *velocity of the particle*, a gradient flow can be defined on $\mathcal{P}(\Omega)$ correspondingly (Ambrosio et al., 2005).

Lemma 1 *Let $\{\mu_\tau\}_{\tau\in[0,1]}$ be an absolutely-continuous curve in $\mathcal{P}(\Omega)$ with finite second-order moments. Then for a.e. $\tau\in[0,1]$, the above vector field $\mathbf{v}_\tau$ defines a gradient flow on $\mathcal{P}(\Omega)$ as $\partial_\tau\mu_\tau + \nabla\cdot(\mathbf{v}_\tau\,\mu_\tau) = 0$, where $\nabla\cdot\mathbf{a} \triangleq \nabla_{\mathbf{x}}^\top\,\mathbf{a}$ for a vector $\mathbf{a}$.*

Function F in Section 2.1 is lifted to be a functional in the space of probability measures, mapping a probability measure μ to a real value, *i.e.*, $F : \mathcal{P}(\Omega) \to \mathbb{R}$. F is the energy functional of a gradient flow on $\mathcal{P}(\Omega)$. Consequently, it can be shown that $\mathbf{v}_\tau$ in Lemma 1 has the form $\mathbf{v}_\tau = -\nabla\frac{\delta F}{\delta\mu_\tau}(\mu_\tau)$ (Ambrosio et al., 2005), where $\frac{\delta F}{\delta\mu_\tau}$ is called the *first variation* of F at μ_τ. Based on this, gradient flows on $\mathcal{P}(\Omega)$ can be written

$$\partial_\tau\mu_\tau = -\nabla\cdot(\mathbf{v}_\tau\,\mu_\tau) = \nabla\cdot\left(\mu_\tau\nabla(\frac{\delta F}{\delta\mu_\tau}(\mu_\tau))\right). \quad (1)$$

Remark 2 *Intuitively, an energy functional F characterizes the landscape structure (appearance) of the corresponding manifold, and the gradient flow (1) defines a solution path on this manifold. Usually, by choosing appropriate F, the landscape is convex, e.g., the Itó-diffusion case defined below. This provides a theoretical guarantee of optimal convergence of a gradient flow.*

Itó diffusions as WGFs Itó diffusion defines a stochastic mapping $\mathcal{T} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ such that we have $\mathcal{T}(\mathbf{x}, 0) = \mathbf{x}$ and $\mathcal{T}(\mathcal{T}(\mathbf{x}, \tau), s) = \mathcal{T}(\mathbf{x}, s + \tau)$, for all $\mathbf{x} \in \mathbb{R}^d$ and $s, \tau \in \mathbb{R}$. A typical example of this family is defined as $\mathcal{T}(\mathbf{x}, \tau) = \mathbf{x}_\tau$, where $\mathbf{x}_\tau$ is driven by a diffusion of the form:

$$\mathrm{d}\,\mathbf{x}_\tau = \nabla U(\mathbf{x}_\tau)\mathrm{d}\tau + \sigma(\mathbf{x}_\tau)\mathrm{d}\mathcal{W}\,. \tag{2}$$

Here $U : \mathbb{R}^d \to \mathbb{R}^d$, $\sigma : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ are called the drift and diffusion terms, respectively; $\mathcal{W}$ is the standard d-dimensional Brownian motion. In Bayesian inference, we seek to make the stationary distribution of $\mathbf{x}_\tau$ approach a particular distribution $p(\mathbf{x})$, *e.g.*, a posterior distribution. One solution for this is to set $U(\mathbf{x}_\tau) = \frac{1}{2}\log p(\mathbf{x})$ and $\sigma(\mathbf{x}_\tau)$ as the $d \times d$ identity matrix. The resulting diffusion is called Langevin dynamics (Welling & Teh, 2011). Denoting the distribution of $\mathbf{x}_\tau$ as μ_τ, it is well known (Risken, 1989) that μ_τ is characterized by the Fokker-Planck (FP) equation:

$$\partial_\tau \mu_\tau = \nabla \cdot \left(-\mu_\tau \nabla U + \nabla \cdot \left(\mu_\tau \sigma\sigma^\top\right)\right)\,. \tag{3}$$

Note (3) is in the gradient-flow form of (1), where the energy functional F is defined as[†]:

$$F(\mu) \triangleq \underbrace{-\int U(\mathbf{x})\mu(\mathbf{x})\mathrm{d}\,\mathbf{x}}_{F_1} + \underbrace{\int \mu(\mathbf{x})\log\mu(\mathbf{x})\mathrm{d}\,\mathbf{x}}_{F_2} \tag{4}$$

Note F_2 is the energy functional of a pure Brownian motion (*e.g.*, $U(\mathbf{x}) = 0$ in (2)). To verify the FP equation with (1), the first variation of F_1 and F_2 is calculated as

$$\frac{\delta F_1}{\delta \mu} = -U, \quad \frac{\delta F_2}{\delta \mu} = \log\mu + 1\,. \tag{5}$$

Substituting (5) into (1) recovers the FP equation (3).

Numerical methods Inspired by the Euclidean-space case, gradient flow (1) can be approximately solved by discretizing time, leading to an iterative optimization problem, where for iteration k: $\mu_{k+1}^{(h)} \in \arg\min_\mu F(\mu) + \frac{W_2^2(\mu, \mu_k^{(h)})}{2h}$. Specifically, for Itó-diffusion with F defined in (4), the optimization problem becomes:

$$\mu_{k+1}^{(h)} = \arg\min_\mu \mathsf{KL}\left(\mu \| p(\mathbf{x})\right) + \frac{W_2^2(\mu, \mu_k^{(h)})}{2h}\,, \tag{6}$$

where $p(\mathbf{x}) \triangleq \frac{1}{Z}e^{U(\mathbf{x})}$ is the target distribution. This procedure is called the Jordan-Kinderlehrer-Otto (JKO) scheme. Remarkably, the convergence of (6) can be guaranteed (Jordan et al., 1998), as stated in Lemma 3.

[†]We assume σ to be independent of $\mathbf{x}$, which is the case in Langevin dynamics whose stationary distribution is set to be proportional to $e^{-U(\mathbf{x})}$. As a result, we drop σ in the following.

Lemma 3 *Assume that* $\log p(\mathbf{x}) \le C_1$ *is infinitely differentiable, and* $\|\nabla \log p(\mathbf{x})\| \le C_2\,(1 + C_1 - \log p(\mathbf{x}))\ (\forall\,\mathbf{x})$ *for some constants* $\{C_1, C_2\}$. *Let* $T = hK$, $\mu_0 \triangleq q_0(\mathbf{x})$, *and* $\{\mu_k^{(h)}\}_{k=1}^K$ *be the solution of the functional optimization problem* (6), *which are restricted to the space with finite second-order moments. Then* i) *the problem* (6) *is convex; and* ii) $\mu_K^{(h)}$ *converges to* μ_T *in the limit of* $h \to 0$, *i.e.,* $\lim_{h\to 0}\mu_K^{(h)} = \mu_T$, *where* μ_T *is the solution of* (3) *at* T.

Remark 4 *Since the stationary distribution of the FP equation* (3) *is proportional to* $e^{U(\mathbf{x})}$, *Lemma 3 suggests that* $\lim_{k\to\infty, h\to 0}\mu_k^{(h)} = \frac{1}{Z}e^U$, *a useful property to guide design of energy functionals for RL, as discussed in Section 4.*

3. Particle Approximation for WGFs

We focus on solving Itó diffusions with scheme (6). Directly reformulating gradient flows as a sequential optimization problem in (6) is infeasible, because $\{\mu_k^{(h)}\}$ are infinite-dimensional objects. We propose to use particle approximation to solve (6), where particles continuously evolve over time. There exist particle-based algorithms for gradient-flow approximations, for example, the stochastic and deterministic particle methods in (Cottet & Koumoutsakos, 2000; Russo, 1990; Carrillo et al., 2017). However, they did not target the JKO scheme, and thus are not applicable to our setting. Another advantage of the JKO scheme is that it allows direct application of gradient-based algorithms, once we get gradients of the particles; thus, it is particularly useful in deep-learning-based methods where parameters are updated by backpropagating gradients through a network.

Following similar idea as in (Chen et al., 2018), in the k-th iteration of our algorithm, M particles $\{\mathbf{x}_k^{(i)}\}_{i=1}^M$ are used to approximate $\mu_k^{(h)}$: $\mu_k^{(h)} \approx \frac{1}{M}\sum_{i=1}^M \delta_{\mathbf{x}_k^{(i)}}$. Our goal is to evolve $\{\mathbf{x}_k^{(i)}\}$ such that the corresponding empirical measure, $\mu^{(h)} \triangleq \frac{1}{M}\sum_{i=1}^M \delta_{\mathbf{x}^{(i)}}$, minimizes (6). A standard method is to use gradient descent to update the particles, where gradients $\{\frac{\partial \mathsf{KL}(\mu, \mu_k^{(h)})}{\partial\,\mathbf{x}^{(i)}}, \frac{\partial W_2^2(\mu, \mu_k^{(h)})}{\partial\,\mathbf{x}^{(i)}}\}$ are required according to (6). By assuming $\mathbf{x}_k^{(i)}$ to evolve in the form of $\mathbf{x}_{k+1}^{(i)} = \mathbf{x}_k^{(i)} + h\phi(\mathbf{x}_k^{(i)})$, with function ϕ restricted to an RKHS with kernel $K(\cdot,\cdot)$, the gradient of the KL term is calculated as (Liu & Wang, 2016):

$$\frac{\partial \mathsf{KL}(\mu^{(h)}, \mu_k^{(h)})}{\partial\,\mathbf{x}^{(i)}} \propto \frac{1}{M}\sum_{j=1}^M \Big[K(\mathbf{x}^{(j)}, \mathbf{x}^{(i)})\nabla_{\mathbf{x}^{(j)}}\log p(\mathbf{x}^{(j)}) + \nabla_{\mathbf{x}^{(j)}} K(\mathbf{x}^{(j)}, \mathbf{x}^{(i)})\Big]\,. \tag{7}$$

The gradient for $W_2^2(\mu^{(h)}, \mu_k^{(h)})$ is more involved, as the distance does not have a closed form. The Wasserstein term arises due to the Brownian motion in the diffusion process (2), and the non-differentiability of a sample path from a

Brownian motion is translated into the Wasserstein distance. We develop a simple yet effective method to overcome this issue below.

First, using a particle approximation, $W_2^2(\mu^{(h)}, \mu_k^{(h)})$ is simplified as

$$W_2^2(\mu, \mu_k^{(h)}) = \inf_{p_{i,j}} \sum_{i,j} p_{ij} c(\mathbf{x}^{(i)}, \mathbf{x}_k^{(j)}) \quad (8)$$
$$s.t. \ \sum_j p_{ij} = \frac{1}{M}, \ \sum_i p_{ij} = \frac{1}{M} ,$$

where $c(\mathbf{x}_1, \mathbf{x}_2) \triangleq \| \mathbf{x}_1 - \mathbf{x}_2 \|_2^2$. Our goal turns to solving for the optimal $\{p_{ij}\}$. Since W_2 comes from the Brownian motion, the energy functional in its gradient flow is defined as F_2 in (4). Solving the gradient flow with the JKO scheme, at each iteration we minimize $\lambda F_2 + W_2^2(\mu, \mu_k^{(h)})$ with λ a regularization parameter. Substituting W_2^2 with (8), introducing Lagrangian multipliers $\{\alpha_i, \beta_i\}$ to deal with the constraints, and letting $c_{ij} \triangleq c(\mathbf{x}^{(i)}, \mathbf{x}_k^{(j)})$, the dual problem is,

$$\mathcal{L}(\{p_{ij}\}, \{\alpha_i\}, \{\beta_i\}) = \lambda \sum_{i,j} p_{ij} \log p_{ij} + p_{ij} c_{ij}$$
$$+ \sum_i \alpha_i (\sum_j p_{ij} - \frac{1}{M}) + \sum_j \beta_j (\sum_i p_{ij} - \frac{1}{M})$$

The optimal p_{ij} have forms of $p_{ij}^* = u_i e^{-c_{ij}/\lambda} v_j$, where $u_i \triangleq e^{-\frac{1}{2} - \frac{\alpha_i}{\lambda}}$, $v_j = e^{-\frac{1}{2} - \frac{\beta_j}{\lambda}}$. Assuming $\{u_i\}$ and $\{v_j\}$ are independent of $\{\mathbf{x}^{(i)}\}$ and $\{\mathbf{x}_k^{(j)}\}$,

$$\frac{\partial W_2^2(\mu, \mu_k^{(h)})}{\partial \mathbf{x}^{(i)}} \propto \frac{\sum_j c_{ij} e^{-c_{ij}/\lambda}}{\partial \mathbf{x}^{(i)}}$$
$$= \sum_j 2 \left(1 - \frac{c_{ij}}{\lambda}\right) e^{-c_{ij}/\lambda} (\mathbf{x}^{(i)} - \mathbf{x}_k^{(j)}) . \quad (9)$$

The gradients of particles can be obtained by combining (7) and (9), which are then optimized using SGD. Intuitively, from (9), the Wasserstein term contributes in two ways: *i*) When $\frac{c_{ij}}{\lambda} > 1$, $\mathbf{x}^{(i)}$ is pulled close to previous particles $\{\mathbf{x}_k^{(j)}\}$, with force proportional to $(\frac{c_{ij}}{\lambda} - 1) e^{-c_{ij}/\lambda}$; *ii*) when $\mathbf{x}^{(i)}$ is close enough to a previous particle $\mathbf{x}_k^{(j)}$, *i.e.*, $\frac{c_{ij}}{\lambda} < 1$, $\mathbf{x}^{(i)}$ is pushed away, preventing it from collapsing to $\mathbf{x}_k^{(j)}$.

4. Policy Optimization as WGFs

Reinforcement learning is the problem of finding an optimal policy for an agent interacting with an unknown environment, collecting a reward per action. A policy is defined as a conditional distribution, $\pi(\mathbf{a} \,|\, \mathbf{s})$, defining the probability over an action $\mathbf{a} \in \mathcal{A}$ conditioned on a state variable $\mathbf{s} \in \mathcal{S}$. Formally, the problem can be described as a Markov decision process (MDP), $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P_s, r, \gamma \rangle$, where $P_s(\mathbf{s}' \,|\, \mathbf{s}, \mathbf{a})$ is the transition probability from state $\mathbf{s}$ to $\mathbf{s}'$ given action $\mathbf{a}$; $r(\mathbf{s}, \mathbf{a})$ is an unknown reward function immediately following the action $\mathbf{a}$ performed at state $\mathbf{s}$; $\gamma \in [0, 1]$ is a discount factor regularizing future rewards. We denote these variables with a subscript t to indicate their time dependency. At each time step t, conditioned on the current state $\mathbf{s}_t$, the agent chooses an action $\mathbf{a}_t \sim \pi(\cdot \,|\, \mathbf{s}_t)$ and receives a reward signal[‡] $r(\mathbf{a}, \mathbf{s})$. The environment as seen by the agent then updates its state as $\mathbf{s}_{t+1} \sim P_s(\cdot \,|\, \mathbf{s}_t, \mathbf{a}_t)$. The goal is to learn an optimal policy such that one obtains the maximum expected total reward, *e.g.*, by maximizing

$$J(\pi) = \sum_{t=1}^{\infty} \mathbb{E}_{P_s, \pi} \left[\gamma^t r(\mathbf{a}, \mathbf{s})\right] = \mathbb{E}_{\mathbf{s} \sim \rho_\pi, \mathbf{a} \sim \pi} \left[r(\mathbf{s}, \mathbf{a})\right] \quad (10)$$

where $\rho_\pi \triangleq \sum_{t=1}^{\infty} \gamma^{t-1} P_r(\mathbf{s}_t = \mathbf{s})$, and $P_r(\mathbf{s})$ denotes the state marginal distribution induced by π. Optimizing the objective in (10) with a maximum entropy constraint provides us with a framework for training stochastic policies, where specific forms of these policy distribution are required, restricting the representation power. To define a more general class of distributions that can represent more complex and multimodal distributions, we adopt the general energy-based policies (Haarnoja et al., 2017), and transform it into the WGF framework.

Specifically, in the WGF framework, policies form a Riemannian manifold on the space of probability measures. The manifold structure is determined by the expected total reward (10), and the geodesic length between two elements (policy distributions) is defined as the standard second-order Wasserstein distance. With convex energy functionals (defined below), searching for an optimal policy reduces to running SGD on the manifold of probability measures.

In the following, we define gradient flows on both parameter-distribution space and policy-distribution space, leading to indirect-policy learning and direct-policy learning, respectively. In indirect-policy learning, a WGF is defined over policy parameters; whereas in direct-policy learning, a WGF is defined over actions. For both settings, different energy functionals are defined based on the expected total reward, as detailed below. We note that most existing deep RL algorithms cannot be included into the two settings, without the concept of WGF. However, their specific techniques could be applied as intermediate ingredients in our framework.

4.1. Indirect-policy learning

With indirect-policy learning, we do not optimize the stochastic policy π directly. Instead, we aim to describe uncertainty of a policy with parameter distributions (weight uncertainty). Thus we define a gradient flow on the parameters. Let a policy be parameterized by $\boldsymbol{\theta}$, denoted as $\pi_{\boldsymbol{\theta}}$. If we treat $\boldsymbol{\theta}$ as stochastic and learn its posterior distribution $p(\boldsymbol{\theta})$ in response to the expected total reward, the policy is implicitly learned in the sense that uncertainty in the parameter is transferred into the policy distribution in prediction. Following (Houthooft et al., 2016; Liu et al., 2017), the objective function is defined as:

$$\max_p \{\mathbb{E}_{p(\boldsymbol{\theta})}[J(\pi_{\boldsymbol{\theta}})] - \alpha \mathsf{KL}(p \| p_0)\} \quad (11)$$

[‡]We assume the reward function to be deterministic, for simplicity; stochastic rewards can be addressed similarly.

where $p_0(\boldsymbol{\theta})$ is the prior of $\boldsymbol{\theta}$; $\alpha \in [0, +\infty)$ is the temperature hyper-parameter to balance exploitation and exploration in the policy. If we use an uninformative prior, $p_0(\theta) = \text{const}$, the KL term is simplified to the entropy as $\max_p\{\mathbb{E}_{p(\boldsymbol{\theta})}[J(\pi_{\boldsymbol{\theta}})] + \alpha\mathcal{H}(p)\}$. By taking the derivative of the objective function, the optimal distribution is shown to have a simple closed form of $p(\boldsymbol{\theta}) \propto \exp\left(J(\pi_{\boldsymbol{\theta}})/\alpha\right)$ (Liu et al., 2017). This formulation is equivalent to a Bayesian formulation of parameter $\boldsymbol{\theta}$, where $p(\boldsymbol{\theta})$ can be seen as the "posterior" distribution, and $\exp(J(\pi_{\boldsymbol{\theta}})/\alpha)$ is the "likelihood" function. A variational (posterior) distribution for $\boldsymbol{\theta}$, denoted as $\mu(\boldsymbol{\theta})$, is learned by solving an appropriate gradient-flow problem. We define an energy functional characterizing the similarity between the current parameter distribution and the true distribution induced by the total reward as

$$\begin{aligned} F(\mu) &\triangleq -\int J(\pi_{\boldsymbol{\theta}})\mu(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta} + \int \mu(\boldsymbol{\theta})\log\mu(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta} \\ &= \mathsf{KL}\left(\mu \| p_{\boldsymbol{\theta}}\right) , \end{aligned} \tag{12}$$

The energy functional defines a landscape determined by the expected total reward, and obtains its minimum when $\mu = p_{\boldsymbol{\theta}}$.

Proposition 5 *For the gradient flow with energy functional defined in (12), μ converges to $p_{\boldsymbol{\theta}}$ in the infinite-time limit.*

To solve the above gradient-flow problem, one can apply the JKO scheme with a stepsize h (we follow previous notation to use subscript k to denote discrete-time solutions and superscript h to denote the stepsize):

$$\mu_{k+1}^{(h)} = \arg\min_{\mu} \mathsf{KL}\left(\mu \| p_{\boldsymbol{\theta}}\right) + \frac{W_2^2(\mu, \mu_k^{(h)})}{2h} . \tag{13}$$

The above problem can be directly solved with gradient descent by adopting the particle approximation described in Section 3. Specifically, let the current particles be $(\boldsymbol{\theta}^{(i)})_{i=1}^M$. When calculating $\frac{\partial \mathsf{KL}(\mu\|p_{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}^{(i)}}$ as in (7), we need to evaluate $\nabla_{\boldsymbol{\theta}^{(i)}} J(\pi_{\boldsymbol{\theta}^{(i)}})$. This can be approximated with REINFORCE (Williams, 1992) or advantage actor critic (Schulman et al., 2016). For example, with REINFORCE,

$$\nabla_{\boldsymbol{\theta}^{(i)}} J(\pi_{\boldsymbol{\theta}^{(i)}}) \approx \frac{1}{T}\sum_{t=1}^{T} \gamma^{t-1} \nabla_{\boldsymbol{\theta}^{(i)}} \log \pi_{\boldsymbol{\theta}^{(i)}}(\mathbf{a}_t \,|\, \mathbf{s}_t) \hat{Q}^{\pi}(\mathbf{s}_t, \mathbf{a}_t)$$

where T is a horizon parameter, and $\hat{Q}^{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ is the Q-value function. We call this variant of our framework Indirect Policy learning with WGF (IP-WGF).

Remark 6 *Assume gradients $\nabla_{\boldsymbol{\theta}^{(i)}} J(\pi_{\boldsymbol{\theta}^{(i)}})$ and $\nabla_{\boldsymbol{\theta}^{(i)}} W_2^2$ are unbiased. Under the limit of $M \to \infty$ and $h \to 0$, and based on the fact that F in (12) is convex, Lemma 3 suggests the particle approximation converges to the global minimum $p_{\boldsymbol{\theta}}$. The conclusion applies, in the next section, similarly in the direct-policy-learning case. Existing methods such as TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017b) do not have such an interpretation, thus understanding their underlying convergence is more challenging. Furthermore, these methods optimize parameters directly as fixed points, deteriorating their ability to explore when policy distributions are inappropriately defined, as stochasticity only comes from the policy distributions.*

4.2. Direct-policy learning

When the dimension of parameter space is high, as is often the case in practice, IP-WGF can suffer from computation and storage inefficiencies. In direct-policy learning, a gradient flow is defined for the distribution of policies, thus a policy is *directly* optimized during learning. This approach appears to be more efficient and flexible, and connects more directly to existing works compared with indirect-policy learning.

Specifically, we consider a general energy-based policies of the form $\pi(\mathbf{a} \,|\, \mathbf{s}) \propto \exp(-\varepsilon(\mathbf{s}, \mathbf{a})/\alpha)$ that is able to model more complex distributions (Haarnoja et al., 2017). We formulate the direct-policy learning as policy-distribution-based gradient flows. The energy functional is defined with respect to the learned policy π, thus it should depend on states. To this end, let $\varepsilon_{s,\pi}(\mathbf{a}) = -Q(\mathbf{a}_t, \mathbf{s}_t)$, where $Q(\mathbf{a}_t, \mathbf{s}_t) \triangleq r(\mathbf{a}_t = \mathbf{a}, \mathbf{s}_t = \mathbf{s}) + \mathbb{E}_{(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}, \cdots) \sim (\rho_\pi, \pi)} \sum_{l=1}^{\infty} \gamma^l r(\mathbf{s}_{t+l}, \mathbf{a}_{t+l})$. $Q(\mathbf{a}_t, \mathbf{s}_t)$ is seen to be a functional depending on the current $\mathbf{s}_t$ and $\mathbf{a}_t$, as well as the policy π. Integrating out the action $\mathbf{a}$, an energy functional characterizing the similarity of the current policy π and the optimal policy, $p_{s,\pi}(\mathbf{a} \,|\, \mathbf{s}) \propto e^{Q(\mathbf{a},\mathbf{s})}$, is readily defined as

$$\begin{aligned} F_s(\pi) &\triangleq -\int Q(\mathbf{a}, \mathbf{s})\pi(\mathbf{a} \,|\, \mathbf{s})\mathrm{d}\,\mathbf{a} + \int \pi(\mathbf{a} \,|\, \mathbf{s})\log\pi(\mathbf{a} \,|\, \mathbf{s})\mathrm{d}\,\mathbf{a} \\ &= \mathsf{KL}\left(\pi \| p_{s,\pi}\right) . \end{aligned} \tag{14}$$

Remark 7 *Soft Q-learning (Haarnoja et al., 2017) adopts $Q(\mathbf{a}, \mathbf{s}) + \mathcal{H}(\pi)$ as the objective function, where the entropy of π, $\mathcal{H}(\pi) \triangleq -\mathbb{E}_\pi[\log \pi]$, is included to add stochasticity into the corresponding Q-function. By treating the problem as a WGF, the stochasticity is modeled directly in the policy distribution, thus we do not include the entropy term, though it is of no harm to add it in.*

Proposition 8 *For a WGF with the energy functional defined in (14), $\pi(\mathbf{a} \,|\, \mathbf{s})$ converges to $p_{s,\pi}(\mathbf{a}) \propto e^{Q(\mathbf{a},\mathbf{s})}$ with $Q(\mathbf{a}, \mathbf{s})$ satisfying the following modified Bellman equation:*

$$Q(\mathbf{a}_t, \mathbf{s}_t) = r(\mathbf{a}_t, \mathbf{s}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim \rho_\pi}[V_\pi(\mathbf{s}_{t+1}) - \mathcal{H}(\pi(\cdot \,|\, \mathbf{s}_{t+1}))]$$

where $V_\pi(\mathbf{s}_{t+1}) \triangleq \log \int_{\mathcal{A}} \exp(Q(\mathbf{a}, \mathbf{s}_{t+1}))\mathrm{d}\,\mathbf{a}$.

To solve the corresponding WGF, we again adopt the JKO scheme as in (13) to optimize the policy π by particle approximation, *i.e.*, $\pi \propto \frac{1}{M}\sum_{i=1}^{M} \delta_{\mathbf{a}^{(i)}}$. One challenge is that when calculating $\frac{\partial \mathsf{KL}(\pi\|p_{s,\pi})}{\partial\, \mathbf{a}^{(i)}}$, from (7), one needs to evaluate $p_{s,\pi}(\mathbf{a}^{(i)})$, which is difficult due to the infinite time horizon and the unknown reward function $r(\mathbf{s}, \mathbf{a})$ when calculating $Q(\mathbf{a}^{(i)}, \mathbf{s})$. To address this, we approximate the **soft** Q-function, $Q(\cdot, \mathbf{s})$, with a deep neural network $Q_s^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$

parametrized by $\boldsymbol{\theta}$, *i.e.*, $p_{s,\pi}(\mathbf{a}) \propto e^{Q_s^{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})}$. The neural network $Q_s^{\boldsymbol{\theta}}$ naturally leads to a soft approximation of the standard Q-function according to Proposition 8. As a result, the learning can be done by alternating between the following two steps.

1) Optimizing the policy Given $Q_s^{\boldsymbol{\theta}}$, we could adopt the particle approximation with the JKO scheme to optimize the policy. However, since a policy is a conditional distribution, one needs to introduce a set of particles for each state. For large or continuous state space, this becomes intractable. To mitigate this problem, we propose to use a stochastic state-conditioned neural network $f^{\boldsymbol{\phi}}$ parametrized by $\boldsymbol{\phi}$ to approximate the policy. We call such a network a *sampling network*.

The input to $f^{\boldsymbol{\phi}}$ is a concatenation of a state $\mathbf{s}$ and a random-noise sample ξ drawn from a simple distribution, *e.g.*, the standard normal distribution. To optimize the sampling network, note that the JKO scheme, with energy functional (14), is written as (we rewrite π as $\pi^{\boldsymbol{\phi}}$ to explicitly indicate the dependence of π on $\boldsymbol{\phi}$):

$$\pi_{k+1}^{\boldsymbol{\phi}} = \arg\min_{\pi^{\boldsymbol{\phi}}} \mathsf{KL}\left(\pi^{\boldsymbol{\phi}} \| p_{s,\pi}\right) + \frac{W_2^2(\pi^{\boldsymbol{\phi}}, \pi_k^{\boldsymbol{\phi}})}{2h} \triangleq J_\pi^{\boldsymbol{\phi}} \ . \quad (15)$$

The outputs of $f^{\boldsymbol{\phi}}(\{\xi_i\}; \mathbf{s}_t)$ are particles $(\mathbf{a}_t^{(i)})_{i=1}^M$. Using chain rule we calculate the gradient of $\boldsymbol{\phi}$ as

$$\frac{\partial J_\pi^{\boldsymbol{\phi}}}{\partial \boldsymbol{\phi}} = \mathbb{E}_{\{\xi_i\}}\left[\frac{\partial J_\pi^{\boldsymbol{\phi}}}{\partial \mathbf{a}_t^{(i)}} \frac{\partial \mathbf{a}_t^{(i)}}{\partial \boldsymbol{\phi}}\right] .$$

Thus $\boldsymbol{\phi}$ can be updated using standard SGD, where $\partial J_\pi^{\boldsymbol{\phi}} / \partial \mathbf{a}_t^{(i)}$ represents particle gradients in the WGF, and is approximated using techniques from Section 3; $\partial \mathbf{a}_t^{(i)} / \partial \boldsymbol{\phi}$ can be calculated by standard backpropagation.

2) Optimizing the Q-network We optimize the Q-network using the Bellman error as in the soft-Q learning setting (Haarnoja et al., 2017). Specifically, in each iteration, we optimize the following objective function:

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

where $q_{\mathbf{s}_t}$ and $q_{\mathbf{a}_t}$ are arbitrary distributions with support on $\mathcal{S}$ and $\mathcal{A}$, respectively; $\hat{Q}_s^{\bar{\boldsymbol{\theta}}}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim \rho_\pi}[V_s^{\bar{\boldsymbol{\theta}}}(\mathbf{s}_{t+1})]$ is the target Q-value, with $V_s^{\bar{\boldsymbol{\theta}}}(\mathbf{s}_{t+1}) = \log \mathbb{E}_{q_{\mathbf{a}'}}[\frac{\exp(Q_s^{\bar{\boldsymbol{\theta}}}(\mathbf{s}_{t+1}, \mathbf{a}'))}{q_{\mathbf{a}'}(\mathbf{a}')}] - \mathcal{H}(q_{\mathbf{a}'})$; $\bar{\boldsymbol{\theta}}$ represents the parameters of the target Q-network, as used in standard deep Q-learning (Mnih et al., 2013). $q_{\mathbf{a}_t}$ can be set to the distribution induced by the sampling network $f^{\boldsymbol{\phi}}$ (Haarnoja et al., 2017). Alternatively, the form of $q_{\mathbf{a}_t}$ can be explicitly defined, *e.g.*, using isotropic Gaussian or mixture of Gaussian distributions. The full algorithm is given in Section G of the SM. We call this variant of our framework Direct Policy learning with WGF (DP-WGF).

Reducing Variance Note that when optimizing the Q-network, one needs to calculate the $V_s^{\boldsymbol{\theta}}$-function. This includes an integration over $q_{\mathbf{a}'}$, which endows the high variance associated with Monte Carlo integration. Consequently, we propose to learn a V-network to approximate $V_s^{\boldsymbol{\theta}}$, denoted as $\bar{V}_{\boldsymbol{\psi}}(\mathbf{s})$ with parameter $\boldsymbol{\psi}$. To learn the V-network, similar to (Haarnoja et al., 2018), we use an explicit policy distribution. As a result, we replace the sampling network $f^{\boldsymbol{\phi}}$ with a BNN discussed in Section 4.1, whose induced policy distribution is denoted $\tilde{\pi}_{\boldsymbol{\phi}}$. Intuitively, $V_s^{\boldsymbol{\theta}}(\mathbf{s})$ can be considered an approximation to the log-normalizer of $\exp(Q_s^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}))$ over $\mathbf{a}$. From the definition, the objective is defined as: $J_V(\boldsymbol{\psi}) \triangleq \mathbb{E}_{\mathbf{s}_t \sim q_{\mathbf{s}_t}}\left(\bar{V}_{\boldsymbol{\psi}}(\mathbf{s}_t) - \log \mathbb{E}_{\mathbf{a}_t \sim \tilde{\pi}_{\boldsymbol{\phi}}(\mathbf{s}_t)} \exp\left(Q_s^{\boldsymbol{\theta}}(\mathbf{s}_t, \mathbf{a}_t) / \tilde{\pi}_{\boldsymbol{\phi}}(\mathbf{a}_t \,|\, \mathbf{s}_t)\right)\right.$ $\left. - \mathbb{E}_{\mathbf{a}_t \sim \tilde{\pi}_{\boldsymbol{\phi}}(\mathbf{s}_t)} \log \tilde{\pi}_{\boldsymbol{\phi}}(\mathbf{a}_t \,|\, \mathbf{s}_t)\right)^2$. In our implementation, we find the following approximation works well: $J_V(\boldsymbol{\psi}) \triangleq \mathbb{E}_{\mathbf{s}_t \sim q_{\mathbf{s}_t}}\left(\bar{V}_{\boldsymbol{\psi}}(\mathbf{s}_t) - \mathbb{E}_{\mathbf{a}_t \sim \tilde{\pi}_{\boldsymbol{\phi}}(\mathbf{s}_t)}[Q_s^{\boldsymbol{\theta}}(\mathbf{s}_t, \mathbf{a}_t) - \log \tilde{\pi}_{\boldsymbol{\phi}}(\mathbf{a}_t \,|\, \mathbf{s}_t)]\right)^2$, which is inspired by (Haarnoja et al., 2018). We call this variant Direct Policy learning with WGF and Variance reduction (DP-WGF-V).

5. Connections with Related Works

Soft-Q learning Though motivated from different perspectives, our DP-WGF results in a similar algorithm as soft-Q learning with energy based policies (Haarnoja et al., 2017). However, DP-WGF is more general, in that we can define different sampling networks, such as a BNN, which can be optimized with the proposed IP-WGF technique.

Soft actor-critic (SAC) SAC (Haarnoja et al., 2018) is an improvement of soft-Q learning, introducing a similar V-network as ours and modeling the policy with a mixture of Gaussians. DP-WGF-V is related to SAC, but with a V-network from a different perspective (variance reduction). Importantly, we define a gradient flow for policy distributions, allowing optimization from a distribution perspective.

Trust-region methods Trust-Region methods are known to stabilize policy optimization in RL (Schulman et al., 2015). Schulman et al. (2017a) illustrate the equivalence between soft Q-learning and policy gradient. In the original TRPO setting, an objective function is optimized subject to a constraint that the updated policy is not too far from the current policy, in terms of the KL divergence (see Section C for a more detailed descriptions). The theory of TRPO suggests adding a penalty to the objective instead of adopting a constraint, resulting in a similar form as our framework. However, we use the Wasserstein distance to penalize the previous and current policies, which is a weaker metric than the KL divergence, and potentially leads to more robust solutions. This is evidenced by the development of Wasserstein GAN (Arjovsky et al.). As a result, our framework can be regarded as a trust-region-based counterpart for solving the soft Q-learning (Haarnoja et al., 2017) and SVPG (Liu et al.,

Dataset	PBP	SVGD	WGF
Boston	-2.57 $\pm$ 0.09	-2.50$\pm$0.03	$\mathbf{-2.40 \pm 0.10}$
Concrete	-3.16 $\pm$ 0.02	-3.08$\pm$0.02	$\mathbf{-2.95 \pm 0.06}$
Energy	-2.04 $\pm$ 0.02	-1.77$\pm$0.02	$\mathbf{-0.73 \pm 0.08}$
Kin8nm	0.90 $\pm$ 0.01	$\mathbf{0.98 \pm 0.01}$	0.97 $\pm$ 0.02
Naval	3.73 $\pm$ 0.01	4.09$\pm$0.01	$\mathbf{4.11 \pm 0.02}$
CCPP	-2.80 $\pm$ 0.05	-2.82$\pm$0.01	$\mathbf{-2.78 \pm 0.01}$
Winequality	-0.97 $\pm$ 0.01	-0.93$\pm$0.01	$\mathbf{-0.87 \pm 0.04}$
Yacht	-1.63 $\pm$ 0.02	-1.23$\pm$0.04	$\mathbf{-0.99 \pm 0.15}$
Protein	-2.97 $\pm$ 0.00	-2.95$\pm$0.00	$\mathbf{-2.88 \pm 0.01}$
YearPredict	-3.60$\pm$NA	-3.58 $\pm$ NA	$\mathbf{-3.57 \pm NA}$

Table 1. Averaged predictions, with standard deviations, in terms of test log-likelihood.

2017). Similar arguments hold for other trust-region methods such as PPO (Schulman et al., 2017b) and Trust-PCL (Nachum et al., 2017), which improve TRPO with either different objective or trust-region constraints.

Noisy exploration Adding noise to the parameters for noisy exploration (Fortunato et al., 2018; Plappert et al., 2018) can be interpreted as a special case of our IP-WGF framework with a single particle. Isotropic Gaussian noisy exploration corresponds to the maximum *a posterior* (MAP) solution with a Gaussian assumption on the posterior distributions of parameters, potentially leading to inferior solutions when the assumption is not met. By contrast, our method is endowed with the ability to explore multimodal distributions, by optimizing the parameter distribution directly. More details are provided in Section C of the SM.

6. Experiments

We test the proposed WGF framework from two perspectives: *i*) the effectiveness of the proposed particle-approximation method for WGF, and *ii*) the advantages of the WGF framework for policy optimization. For *i*), a standard regression model to learn optimal parameter distributions, *i.e.*, posterior distributions. For *ii*), we test our algorithms on several domains in OpenAI `rllab` and `Gym` (Duan et al., 2016). All experiments are conducted on a single Tesla P100. Detailed settings are given in the SM.

6.1. Regression

We use a single-layer BNN as a regression model. The parameters of the BNN are treated probabilistically and optimized with our WGF framework. We compare WGF, SVGD (Liu & Wang, 2016), Bayesian Dropout (Gal & Ghahramani, 2016) and PBP (Hernández-Lobato & Adams, 2015). The RMSprop optimizer is employed. Detailed experimental settings and datasets are described in Section D.2 of the SM. We adopt the root-mean-squared error (RMSE) and test log-likelihood as the evaluation criteria. The experimental results are shown in Table 1 (complete results are provided in Section D.2 of the SM). It is observed that our proposed WGF obtains better results in both metrics, partially due to the flexibility of our particle approximation algorithm, which solves the original WGF problem effectively.

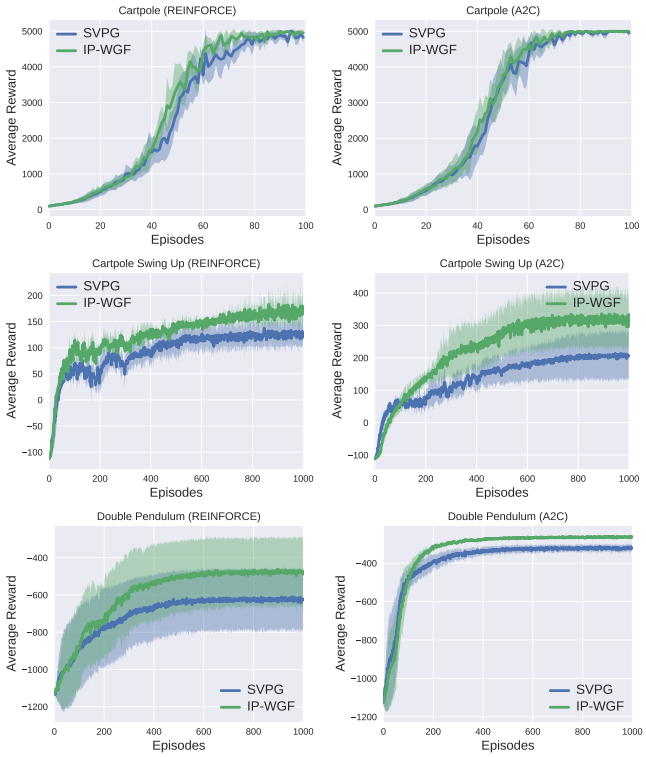

Figure 1. Learning curves by IP-WGF and SVPG with REINFORCE and A2C.

6.2. Indirect-policy learning

For this group of experiments, we compare IP-WGF with SVPG (Liu et al., 2017), a state-of-the-art method for indirect-policy learning, considering three classical continuous-control tasks: Cartpole Swing-Up, Double Pendulum, and Cartpole. Only policy parameters are updated by IP-WGF or SVPG, while the critics are updated with TD-error. We train our agents for 100 iterations on the easier Cartpole domain and 1000 iterations on the other two domains. Following the settings in (Liu et al., 2017; Houthooft et al., 2016; Zhang et al., 2018), the policy is parameterized as a two-layer (25-16 hidden units) neural network with `tanh` activation function. The maximum horizon length is set to 500. A sample size of 5000 is used for policy gradient estimation. We use $M = 16$ particles to approximate parameter distributions, and $h = 0.1$ as the discretized stepsize.

REINFORCE (Williams, 1992) and advantage actor critic (A2C) (Schulman et al., 2016) are used as strategies of policy learning. Figure 2 plots the mean (dark curves) and standard derivation (light areas) of rewards over 5 runs. It is clear that in all tasks IP-WGF consistently converges faster than SVPG, and finally converges to higher average rewards. The results are comparable to (Houthooft et al., 2016). The experiments demonstrate that employing the Wasserstein gradient flows on policy optimization improves the performance, as suggested by our theory.

		WGF-DP-V		SAC		TRPO-GAE		DDPG	
Domain	Threshold	MaxReturn.	Episodes	MaxReturn	Epsisodes	MaxReturn	Episodes	MaxReturn	Episodes
Swimmer	100	**181.60**	**76**	180.83	112	110.58	433	49.57	N/A
Walker	3000	**4978.59**	**2289**	4255.05	2388	3497.81	3020	2138.42	N/A
Hopper	2000	**3248.76**	**678**	3146.51	736	2604	1749	1317	N/A
Humanoid	2000	3077.84	**18740**	2212.51	26476	**5411.15**	32261	2230.60	34652

Table 2. WGF-DP-V, TRPO, SAC, and DDPG results showing the max average rewards attained and the episodes to cross specific reward thresholds. WGF-DP-V often learn more sample-efficiently than the baselines, and WGF-DP-V can solve difficult domains such as Humanoid better than DDPG.

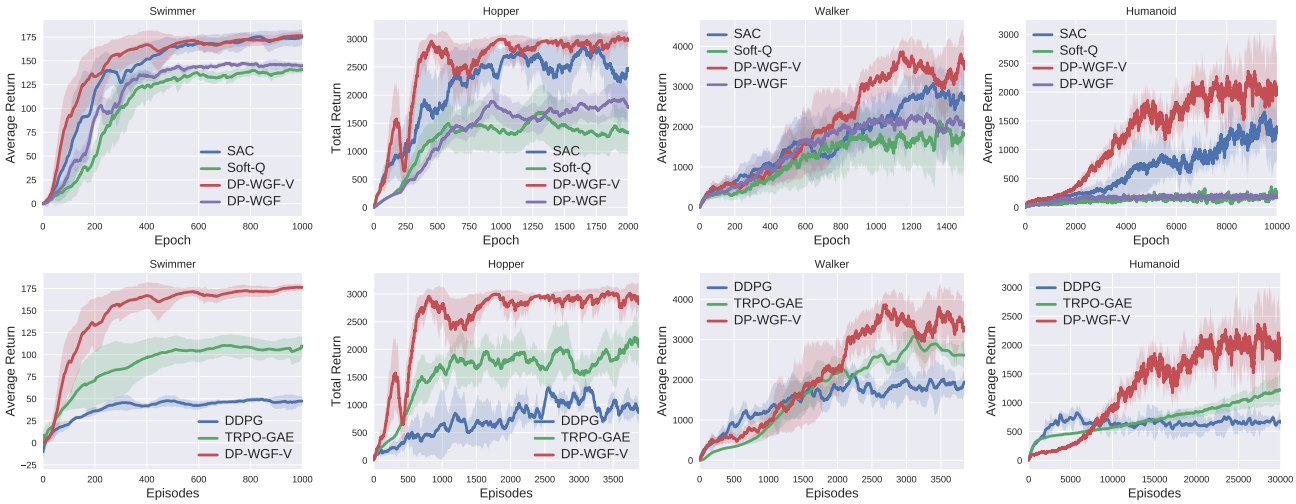

Figure 2. Average return in MuJoCo tasks by Soft-Q, SAC and DP-WGF-V (first row), and by DDPG, TRPO-GAE and DP-WGF-V (second row). From left to right, the tasks are: Swimmer, Hopper, Walker and Humanoid, respectively.

6.3. Direct-policy Learning

We compare our DP-WGF and DP-WGF-V frameworks with existing off-policy and on-policy deep RL algorithms on several tasks in MuJoCo, *e.g.*, SAC, Soft-Q, DDPG (off-policy) and TRPO-GAE (on-policy). Our DP-WGF-V is considered to be an off-policy actor-critic method. For all methods, value function and policy are parameterized as two-layer (128-128 hidden units) neural networks with `tanh` as the activation function. The maximum horizon length is set to 1000 when simulating expected total rewards. Three easier tasks (Swimmer, Hopper and Walker) in MuJoCo can be solved by a wide range of algorithms; while the more complex benchmark, the 21-dimensional Humanoid, is known to be very difficult to solve with off-policy algorithms (Duan et al., 2016). Implementation details of the algorithms are specified in Section E.3 of the SM.

Effectiveness of the Wasserstein trust-region We evaluate DP-WGF-V against SAC, and DP-WGF against Soft-Q on four Mujoco tasks, as they are closely related to our algorithms. Figure 2 (first row) plots average returns over epochs on the tasks. Similarly, our WGF-based methods converge faster and better than their counterparts due to the introduction of WGFs. Furthermore, by variance reduction, DP-WGF-V significantly outperforms DP-WGF on all tasks.

Comparisons with popular baselines Finally we compare DP-WGF-V with TRPO-GAE (Schulman et al., 2016) and DDPG (Lillicrap et al., 2016) on the same Mujoco tasks. In general, TRPO-GAE has been a state-of-the-art method for policy optimization. Figure 2 (second row) plots average returns over episodes, and it is observed that DP-WGF-V consistently outperforms other algorithms. Table 2 summarizes some key statistics, including the best attained average rewards and the episodes to reach the reward thresholds. It is observed that DP-WGF-V consistently outperform TRPO-GAE and DDPG in terms of sample complexity, and often achieves higher rewards than TRPO-GAE. A particularly notable case, on Humanoid, shows DP-WGF-V substantially outperforms TRPO-GAE in terms of sample efficiency, while DDPG cannot learn a good policy at all.

7. Conclusion

We lift policy optimization to the space of probabilistic distributions, and interpret it as Wasserstein gradient flows. Two types of WGFs are defined for the task, one on the parameter-distribution space and the other on the policy-distribution space. The WGFs are solved by a new particle-approximation-based algorithm, where gradients of particles are calculated in closed forms. Under some circumstance, optimization on probability-distribution space is convex, thus it is easier to deal with compared to existing methods. Experiments are conducted on a number of reinforcement-learning tasks, demonstrating the superiority of the proposed framework compared to related algorithms.

References

Ambrosio, L., Gigli, N., and Savaré, G. *Gradient Flows in Metric Spaces and in the Space of Probability Measures*. Lectures in Mathematics ETH Zrich, 2005.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein GAN. In *NIPS*.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. In *ICML*, 2015.

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

Carrillo, J. A., Craig, K., and Patacchini, F. S. A blob method for diffusion. (arXiv:1709.09195), 2017.

Chen, C., Zhang, R., Wang, W., Li, B., and Chen, L. A unified particle-optimization framework for scalable Bayesian sampling. In *UAI*, 2018.

Cottet, G. H. and Koumoutsakos, P. D. *Vortex methods*. Cambridge University Press, 2000.

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

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. Noisy networks for exploration. In *ICLR*, 2018.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In *ICML*, 2016.

Gobbino, M. Minimizing movements and evolution problems in Euclidean spaces. *Annali di Matematica Pura ed Applicata*, 1999.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *ICLR*, 2017.

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

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and D. Meger, David. Deep reinforcement learning that matters. In *AAAI*, 2018.

Hernández-Lobato, J. M. and Adams, R. P. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *ICML*, 2015.

Hinton, Geoffrey E, Srivastava, Nitish, and Swersky, Kevin. Rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural Networks for Machine Learning, Coursera*, 2012.

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

Jordan, R., Kinderlehrer, D., and Otto, F. The variational formulation of the Fokker-Planck equation. *SIAM Journal on Mathematical Analysis*, 29(1):1–17, 1998.

Kaelbling, Leslie Pack, Littman, Michael L, and Moore, Andrew W. Reinforcement learning: A survey. In *JAIR*, 1996.

Kakade, Sham M. A natural policy gradient. In *NIPS*, 2002.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Li, Y. Deep reinforcement learning: An overview. (arXiv:1701.07274), 2017.

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

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

Liu, Y., Ramachandran, P., Liu, Q., and Peng, J. Stein variational policy gradient. In *UAI*, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing Atari with deep reinforcement learning. *arXiv:1312.5602*, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 2015.

Mnih, Volodymyr, Badia, Adria Puigdomenech, Mirza, Mehdi, Graves, Alex, Lillicrap, Timothy, Harley, Tim, Silver, David, and Kavukcuoglu, Koray. Asynchronous methods for deep reinforcement learning. In *ICML*, pp. 1928–1937, 2016.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Trust-PCL: An off-policy trust region method for continuous control. (arXiv:1707.01891), 2017.

O'Donoghue, Brendan, Munos, Remi, Kavukcuoglu, Koray, and Mnih, Volodymyr. Pgq: Combining policy gradient and q-learning. *arXiv:1611.01626*, 2016.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. Parameter space noise for exploration. In *ICLR*, 2018.

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

Rulla, J. Error analysis for implicit approximations to solutions to Cauchy problems. *SIAM Journal on Numerical Analysis*, 33(1):68–87, 1996.

Russo, G. Deterministic diffusion of particles. *Communications on Pure and Applied Mathematics*, 1990.

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

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

Schulman, J., Abbeel, P., and Chen, X. Equivalence between policy gradients and soft q-learning. (arXiv:1704.06440), 2017a.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. (arXiv:1707.06347), 2017b.

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

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. 1998.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, 2000.

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

Watkins, C. J. and Dayan, P. Q-learning. *Machine Learning*, 1992.

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

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

Zhang, Ruiyi, Li, Chunyuan, Chen, Changyou, and Carin, Lawrence. Learning structural weight uncertainty for sequential decision-making. In *AISTATS*, 2018.

Supplemental Material for Policy Optimization as Wasserstein Gradient Flows

A. More Details on Preliminaries

We provide more details on some parts of the preliminaries section in the main text.

A.1. Gradient Flows on the Euclidean Space

For a smooth (convex) function[§] $F : \mathbb{R}^n \to \mathbb{R}$, a starting point $\mathbf{x}_0 \in \mathbb{R}^n$. The gradient flow of $F(\mathbf{x})$ is defined as the solution of the following function:

$$\begin{cases} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = -\nabla F(\mathbf{x}(t)), & \text{for } t > 0 \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases} \tag{16}$$

This is a standard Cauchy problem (Rulla, 1996), which has a unique solution if ∇F is Lipschitz continuous. When F is non-differentiable, we can replace the gradient with the subgradient, defined as $\partial F(\mathbf{x}) \triangleq \{\mathbf{p}' \in \mathbb{R}^n : F(\mathbf{y}) \geq F(\mathbf{x}) + \mathbf{p}\cdot(\mathbf{y}-\mathbf{x}), \forall \mathbf{y} \in \mathbb{R}^n\}$. Note $\partial F(\mathbf{x}) = \{\nabla F(\mathbf{x})\}$ if F is differentiable at $\mathbf{x}$. In this case, the gradient flow formula above is replaced as: $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} \in -\partial F(\mathbf{x}(t))$.

Numerical solution Exact solution to the gradient-flow problem (16) is typically intractable. Numerical methods is a default choice. A standard method to solve (16) is called *Minimizing Movement Scheme* (MMS), which is an iterative scheme that evolves $\mathbf{x}$ along the gradient of F on the current point for a small step in each iteration. Specifically, let the current point to be $\mathbf{x}_k$, the next point is defined as $\mathbf{x}_{k+1} = \mathbf{x}_k - \nabla F(\mathbf{x}_{k+1})h$, with h being the stepsize. Note $\mathbf{x}_{k+1}$ is equivalent to solving the following optimization problem:

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} F(\mathbf{x}) + \frac{\| \mathbf{x} - \mathbf{x}_k \|^2}{2h} .$$

To explicitly spell out the dependency of $\mathbf{x}_k$ w.r.t. h, we rewrite $\mathbf{x}_k$ as $\mathbf{x}_k^{(h)}$. The numerical scheme can be proved to be accurate. Specifically, define $\mathbf{v}_{k+1}^{(h)} \triangleq \frac{\mathbf{x}_{k+1}^{(h)} - \mathbf{x}_k^{(h)}}{h}$. Also define two curves $\mathbf{x}^h, \tilde{\mathbf{x}}^h : [0, T] \to \mathbb{R}^n$ for $t \in (kh, (k+1)h]$ as:

$$\mathbf{x}^h(t) = \mathbf{x}_{k+1}^{(h)}, \;\; \tilde{\mathbf{x}}^h(t) = \mathbf{x}_k^{(h)} + (t - kh)\, \mathbf{v}_{k+1}^{(h)} \;\; .$$

The MMS is proved to converge to the original gradient flow (Ambrosio et al., 2005), stated in the following theorem.

[§]We will focus on the convex case since this the case for many gradient flows on the space of probability measures, detailed later.

Theorem 9 *Let* $\mathbf{v}^h(t) \triangleq \mathbf{v}_{k+1}^{(h)}$ *defined above. Suppose* $F(\mathbf{x}_0) < +\infty$ *and* $\inf F > -\infty$. *If*[¶] $h \to 0$, $\tilde{\mathbf{x}}^h$ *and* $\mathbf{x}^h$ *converge uniformly to a same curve* $\mathbf{x}(t)$, *and* $\mathbf{v}^h$ *weakly converges in* $\mathcal{L}^2$ *to a vector function* $\mathbf{v}(t)$, *such that* $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{v}$. *Furthermore, if the partial derivatives of* F *exist and are continuous, we have* $\mathbf{v}(t) = -\nabla F(\mathbf{x}(t))$ *for all* t.

B. Sketch Proofs for RL with WGF

Proof [Proof of Proposition 5]

We provide two methods for the proof.

The first method directly uses property of gradient flows. Note that the WGF is defined as

$$\begin{aligned} \partial_\tau \mu_\tau &= -\nabla \cdot (\mathbf{v}_\tau\, \mu_\tau) = \nabla \cdot \left(\mu_\tau \nabla (\frac{\delta F}{\delta \mu_\tau}(\mu_\tau)) \right) \\ &\triangleq -\nabla_W F(\mu_\tau) \, . \end{aligned}$$

Denote the inner product in the probability space induced by W_2 as $\langle \cdot, \cdot \rangle_W$, we have

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\tau} F(\mu_\tau) &= \langle \nabla_W F(\mu_\tau), \frac{\mathrm{d}}{\mathrm{d}\tau} \mu_\tau \rangle_W \\ &= -\langle \nabla_W F(\mu_\tau), \nabla_W F(\mu_\tau) \rangle_W \, . \end{aligned} \tag{17}$$

For any $\tau_1 \geq \tau_0$, integrating (17) over $[t_0, t_1]$, we have

$$\begin{aligned} & F(\mu_{\tau_1}) - F(\mu_{\tau_0}) \\ =& -\int_{\tau_0}^{\tau_1} \langle \nabla_W F(\mu_\tau), \nabla_W F(\mu_\tau) \rangle_W \mathrm{d}\tau \leq 0 \, , \end{aligned}$$

where the last inequality holds due to the positiveness of the norm operator. Consequently, we have $F(\mu_{\tau_1}) \leq F(\mu_{\tau_0})$, which means the energy functional F decreases over time. In our case, the energy functional is defined as the KL divergence, which is convex in terms of distributions. As a result, evolving μ_τ along the gradient flow would reach the global minimum of the energy functional, *i.e.*, $\lim_{\tau \to \infty} \mu_\tau = e^{J(\pi_{\boldsymbol{\theta}})}$.

The second way uses property of the Fokker-Planck equation for diffusions. Since the WGF with energy functional in (12) is equivalent to a Fokker-Planck equation. Specifically,

[¶]h can also be a decreasing-stepsize sequence $\{h_k\}$ such that $h_k \to 0$.

according to Section 2.2, the solution of the gradient flow is described by the following Fokker-Planck equation:

$$\partial_\tau \mu_\tau = \nabla \cdot (-\mu_\tau \nabla J(\pi_{\boldsymbol{\theta}}) + \nabla \cdot (\mu_\tau)) \ , \qquad (18)$$

On the other hand, it is well known that the unique invariant probability measure for the FP equation (18) is:

$$\mu = e^{J(\pi_{\boldsymbol{\theta}})} = \lim_{\tau \to \infty} \mu_\tau \ .$$

This completes the proof. ■

Proof [Proof of Proposition 8] The first part of Proposition 8, stating that $\pi(\mathbf{a} \,|\, \mathbf{s})$ converges to $p_{s,\pi}(\mathbf{a}) \propto e^{Q(\mathbf{a},\mathbf{s})}$, follows by the same argument as the proof of Proposition 5. Now we derive the soft Bellman equation.

This follows from the definition of $Q(\mathbf{a}_t, \mathbf{s}_t)$, *i.e.*,

$$\begin{aligned}
&Q(\mathbf{a}_t, \mathbf{s}_t) \\
=&r(\mathbf{a}_t, \mathbf{s}_t) + \mathbb{E}_{(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}, \cdots) \sim (\rho_\pi, \pi)} \sum_{l=1}^{\infty} \gamma^l r(\mathbf{s}_{t+l}, \mathbf{a}_{t+l}) \\
=&r(\mathbf{a}_t, \mathbf{s}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim \rho_\pi} \mathbb{E}_{\mathbf{a}_{t+1} \sim \pi} \left[r(\mathbf{a}_{t+1}, \mathbf{s}_{t+1}) \right. \\
&\left. + \mathbb{E}_{(\mathbf{s}_{t+2}, \mathbf{a}_{t+2} \cdots) \sim (\rho_\pi, \pi)} \sum_{l=1}^{\infty} \gamma^l r(\mathbf{s}_{t+1+l}, \mathbf{a}_{t+1+l}) \right]
\end{aligned}$$

Since $\pi(\mathbf{a} \,|\, \mathbf{s}) = e^{Q(\mathbf{a},\mathbf{s}) - V_\pi(\mathbf{s})}$ where $V_\pi(\mathbf{s}) = \int_{\mathcal{A}} Q(\mathbf{a}, \mathbf{s}_{t+1}) \mathrm{d}\, \mathbf{a}$, we have

$$\begin{aligned}
Q(\mathbf{a}_t, \mathbf{s}_t) = &\, r(\mathbf{a}_t, \mathbf{s}_t) \\
&+ \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim \rho_\pi} \left[V_\pi(\mathbf{s}_{t+1}) - \mathcal{H}(\pi(\cdot \,|\, \mathbf{s}_{t+1})) \right]
\end{aligned}$$

■

C. More Details on Related Works

For reference, in addition to Section 5, we provide more details on the connection of our framework compared to existing methods.

Connections with trust region methods Trust Region methods can stabilize policy optimization in RL (Nachum et al., 2017; Kakade, 2002). We can also show the connection between the proposed method and TRPO (Schulman et al., 2015). In TRPO, an objective function is maximized subjected to a constraint on the size of policy update. Specifically,

$$\max_{\boldsymbol{\phi}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi^\phi(\cdot \,|\, \mathbf{s})}{\pi^{\phi_{k-1}}(\cdot \,|\, \mathbf{s})} \hat{A}_k \right] \qquad (19)$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t \left[\mathrm{KL}[\pi^\phi(\cdot \,|\, \mathbf{s}), \pi^{\phi_{k-1}}(\cdot \,|\, \mathbf{s})] \right] \leq \delta \qquad (20)$$

Here, π_ϕ is a stochastic policy; ϕ_{k-1} is the vector of policy parameters before the k-th update; $\hat{A}_k$ is an estimator of the advantage function at timestep k. The theory of TRPO suggests using a penalty instead of a constraint, *i.e.*, solving the unconstrained optimization problem,

$$\max_{\boldsymbol{\phi}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi^\phi(\cdot \,|\, \mathbf{s})}{\pi^{\phi_{t-1}}(\cdot \,|\, \mathbf{s})} \hat{A}_t - \beta \mathrm{KL}[\pi^\phi(\cdot \,|\, \mathbf{s}), \pi^{\phi_{t-1}}(\cdot \,|\, \mathbf{s})] \right] \qquad (21)$$

In our proposed framework, the Wasserstein distance between $\pi^\phi(\cdot \,|\, \mathbf{s})$ and $\pi^{\phi_{k-1}}(\cdot \,|\, \mathbf{s})$, 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 (Arjovsky et al.). As a result, our framework can be regarded as a trust region based counterpart for solving the soft Q-learning problem (Haarnoja et al., 2017).

Connections with noisy exploration In our framework, adding noise to the parameters, leading to noisy exploration can be interpreted as a special case of indirect policy learning with single particle. As shown in (Fortunato et al., 2018; Plappert et al., 2018), independent Gaussian noisy linear layer is defined as.

$$y \overset{\text{def}}{=} \quad (\mu^w + \sigma^w \odot \epsilon^w) x + \mu^b + \sigma^b \odot \epsilon^b, \qquad (22)$$

The parameters $\mu^w \in \mathbb{R}^{q \times p}$, $\mu^b \in \mathbb{R}^q$, $\sigma^w \in \mathbb{R}^{q \times p}$ and $\sigma^b \in \mathbb{R}^q$ are learnable whereas $\epsilon^w \in \mathbb{R}^{q \times p}$ and $\epsilon^b \in \mathbb{R}^q$ are random noises, where p and q are the number of hidden units of connected layers.

It corresponds to the maximum a posterior (MAP) with a Gaussian assumption on the posterior distributions of parameters (weight uncertainty). In our framework, we can explicitly (Liu & Wang, 2016) or implicitly (Blundell et al., 2015) define the weight uncertainty. Especially, when employing SGLD (Welling & Teh, 2011) to approximate the posterior distributions of parameters, we will using noisy gradient instead of noisy weights in the parameter space.

Previous work, such as DDPG (Lillicrap et al., 2016), adding noise to the action to encourage exploration can be regarded as a special case of DP-WGF. Adding noise in parameter space has shown superiority with action space, but the computational cost of employing particles to approximate parameter distribution is much higher than that of directly approximate policy distribution. Previous work (Fortunato et al., 2018; Plappert et al., 2018) made a trade-off and optimize the MAP instead of the distribution.

D. Extensive Experiments

To optimize over the discretized WGF via the JKO scheme (6), to need to specify the discretized stepsize h. In addition, we have an additional hyparameter λ in the gradient formula of W_2^2. Also note that we can only evaluate the gradient of W_2^2 up to a constant, there needs to a parameter balancing the gradient of the energy functional F and the Wasserstein term. We denote this hyparameter as ϵ. In the experiments, if not explicitly stated, the default setting for these parameters are $\epsilon = 0.4$, and $\lambda = \texttt{med}^2/\log M$. Here med is the median of the pairwise distance between particles of consecutive policies. Adam (Kingma & Ba, 2015) optimizer is used for all experiments, except the BNN regression, for which we use RMSPorp (Hinton et al., 2012).

D.1. Comparative Evaluation

DP-WGF-V learns substantially faster than popular baselines on four tasks. In the Humanoid task, even though TRPO-GAE does not outperforms DP-WGF-V within the range depicted in the Figure 2, it achieves good final rewards after more episodes. The quantitative results in our experiments are also comparable to results reported by other methods in prior work (Duan et al., 2016; Gu et al., 2017; Henderson et al., 2018; O'Donoghue et al., 2016; Mnih et al., 2016), showing sample efficiency and good performance.

D.2. BNN for regression

For SVGD-based methods, we use a RBF kernel $\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}') = \exp(-\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2^2/m)$, with the bandwidth set to $m = \texttt{med}^2/\log M$. Here med is the median of the pairwise distance between particles. We use a single-layer BNN for regression tasks. Following (Blundell et al., 2015), 10 UCI public datasets are considered: 100 hidden units for 2 large datasets (Protein and YearPredict), and 50 hidden units for the other 8 small datasets. We repeat the experiments 20 times for all datasets except for Protein and YearPredict, which we repeat 5 times and once, respectively, for computation consideration (Blundell et al., 2015). The experiment settings are almost identical to those in (Blundell et al., 2015), except that the prior of covariances follow $\mathsf{Inv\text{-}Gamma}(1, 0.1)$. The batch size for the two large datasets is set to 1000, while it is 100 for the small datasets. The datasets are randomly split into 90% training and 10% testing. Table 3 shows the complete results for different models on all the datasets.

D.3. Toy example in a multi-goal environment

We use the similar toy example as in soft Q-learning to show that our proposed , where the environment is defined as a multi-modal distribution,

Figure 3 illustrates a 2D multi-goal environment. The left one shows trajectories from a policy learned with DP-WGF-V. The x and y axes correspond to 2D positions (states). The agent is initialized near the origin, and the first step of trajectory is omitted. Red dots are depicted goals and the environment is terminated once the distance between the agents and some goal meets predefined threshold. The level curves show the distance to the goal.

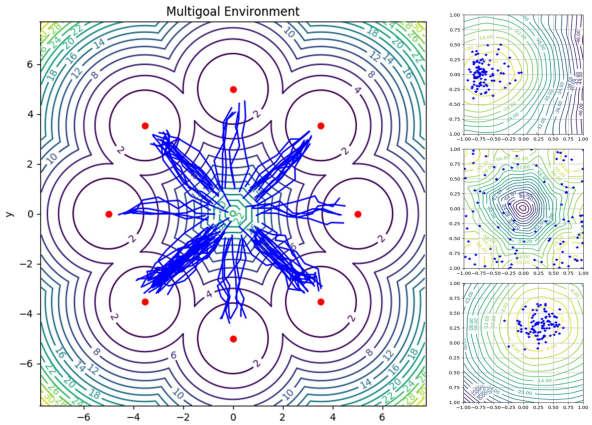

Figure 3. DP-WGF-V on multi-goal Environment.

Q-values at three selected states (-2.5, 0), (0, 0), (2.5, 2.5) are presented on the right, depicted by level curves (yellow: high values, red: low values). The x and y axes correspond to 2D velocity (actions) bounded between -1 and 1. Actions sampled from the policy are shown as blue stars. The experiments shows that our methods have the ability to learn multi-goal policies while achieving better stability and rewards than soft-Q learning.

D.4. Hyperparameter Sensitivity

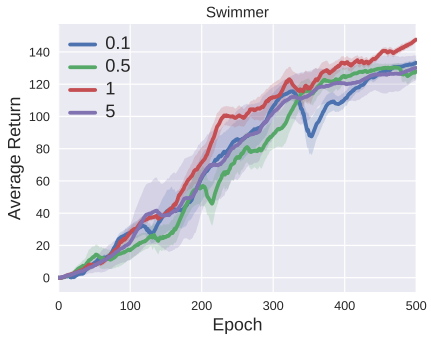

Figure 4. Sensitivity of Hyper-parameters

We further conduct experiments on Swimmer-v1 task to analysis the influence of different Wasserstein-2 scale ϵ. We run the algorithm for 500 epochs and 5 re-runs. Figure 4(a) shows the mean of average return against epoch. From the experiments, with appropriate ϵ, the learning curves become more stable and achieves higher final rewards; while

	Test RMSE ↓				Test Log likelihood ↑			
Dataset	Dropout	PBP	SVGD	WGF	Dropout	PBP	SVGD	WGF
Boston	4.32 ± 0.29	3.01 ± 0.18	2.96±0.10	$\mathbf{2.46 \pm 0.34}$	-2.90 ± 0.07	-2.57 ± 0.09	-2.50±0.03	$\mathbf{-2.40 \pm 0.10}$
Concrete	7.19 ± 0.12	5.67 ± 0.09	5.32±0.10	$\mathbf{4.59 \pm 0.29}$	-3.39 ± 0.02	-3.16 ± 0.02	-3.08±0.02	$\mathbf{-2.95 \pm 0.06}$
Energy	2.65 ± 0.08	1.80 ± 0.05	1.37±0.05	$\mathbf{0.48 \pm 0.04}$	-2.39 ± 0.03	-2.04 ± 0.02	-1.77±0.02	$\mathbf{-0.73 \pm 0.08}$
Kin8nm	0.10 ± 0.00	0.10 ± 0.00	$\mathbf{0.09 \pm 0.00}$	$\mathbf{0.09 \pm 0.00}$	0.90 ± 0.01	0.90 ± 0.01	$\mathbf{0.98 \pm 0.01}$	0.97 ± 0.02
Naval	0.01 ± 0.00	.01± 0.00	0.00±0.00	$\mathbf{0.00 \pm 0.00}$	3.73 ± 0.12	3.73 ± 0.01	4.09±0.01	$\mathbf{4.11 \pm 0.02}$
CCPP	4.33 ± 0.04	4.12± 0.03	4.03±0.03	$\mathbf{3.88 \pm 0.06}$	-2.89 ± 0.02	-2.80 ± 0.05	-2.82±0.01	$\mathbf{-2.78 \pm 0.01}$
Winequality	0.65 ± 0.01	0.64 ± 0.02	0.61±0.01	$\mathbf{0.57 \pm 0.03}$	-0.98 ± 0.01	-0.97 ± 0.01	-0.93±0.01	$\mathbf{-0.87 \pm 0.04}$
Yacht	6.89 ± 0.67	1.02 ± 0.05	0.86±0.05	$\mathbf{0.56 \pm 0.16}$	-3.43 ± 0.16	-1.63 ± 0.02	-1.23±0.04	$\mathbf{-0.99 \pm 0.15}$
Protein	4.84 ± 0.03	4.73 ± 0.01	4.61±0.01	$\mathbf{4.24 \pm 0.02}$	-2.99 ± 0.01	-2.97 ± 0.00	-2.95±0.00	$\mathbf{-2.88 \pm 0.01}$
YearPredict	$9.03 \pm$ NA	$8.88 \pm$ NA	8.68± NA	$\mathbf{8.66 \pm NA}$	$-3.62 \pm$ NA	-3.60±NA	$-3.58 \pm$ NA	$\mathbf{-3.57 \pm NA}$

Table 3. Averaged predictions with standard deviations in terms of RMSE and log-likelihood on test sets.

too large scale of ϵ will reduce the final rewards of policy, since the update size is excessively restricted. The results also show that the scale ϵ of Wasserstein trust-region is not parameter sensitive.

E. Implementation Details

E.1. Smoothing previous policy

Towards Wasserstein-2 distance, we need to use consecutive to compute policies $W_2^2(\pi^{\overline{\boldsymbol{\phi}}}(\cdot|\,\mathbf{s}_t), \pi^{\boldsymbol{\phi}}(\cdot|\,\mathbf{s}_t))$. For the previous policy $\pi^{\overline{\boldsymbol{\phi}}}(\cdot|\,\mathbf{s}_t)$, there are two strategy to get it. *i*) policy of last iteration, *i.e.* $\overline{\boldsymbol{\phi}} = \boldsymbol{\phi}_{k-1}$. *ii*) moving average of prior policy, *i.e.* $\overline{\boldsymbol{\phi}} = (1-\tau)\overline{\boldsymbol{\phi}} + \tau\boldsymbol{\phi}_{k-1}$. Empirically, when the learning curve is stable, (e.g. Half-Cheetah-v1), adopting strategy *i*) is helpful, and strategy *ii*) will reduce the speed of convergence and may lead lower final rewards; Otherwise, strategy *ii*) will help stabilize the training, and speed up the convergence.

Table 4. Shared parameters of direct policy learning

Parameter	Symbol	Value
horizon		500
batch size		5000
learning rate		5×10^{-3}
discount	γ	0.99
hidden units		[25, 16]
variance (prior)		0.01
temperature	α	$\{6, 7, 8, 9, 10, 11\}$

E.2. Indirect Policy learning

For the easy task, Cartpole, all agents are trained for 100 episodes. For the two complex tasks, Cartpole Swing-Up and Double Pendulum, all agents are trained up to 1000 episodes. SVPG and IP-WGF shared the same hyperparameters, except the temperature, for which we performed a grid search over $\alpha \in \{6, 7, 8, 9, 10, 11\}$.

E.3. Direct-Policy learning

We use OpenAI gym[‖] (Brockman et al., 2016) and rl-lab[**] (Duan et al., 2016) baselines implementations for TRPO and DDPG. SAC[††] and Soft-Q[‡‡] implementation are used, and we use recommended parameters.

Hyperparameters Table 5 lists the common DP-WGF-V, DP-WGF, SAC and Soft-Q parameters used in the comparative evaluation in Figure 2, and Table 2 lists the parameters that varied across the environments. For SAC, we use 4 components of mixture Gaussian. For DP-WGF and Soft-Q, 32 particles are used to approximate policy distributions.

Table 5. Shared parameters of indirect policy learning

Parameter	Symbol	Value
horizon		1000
batch size		64
learning rate		$3\cdot10^{-4}$
discount	γ	0.99
target smoothing coefficient	τ	0.01
number of layers (3 networks)		2
number of hidden units per layer		128
gradient steps		1
scale of Wasserstein trust-region		0.4

Table 6. Environment Specific Parameters

Environment	DoFs	Reward Scale	Replay Pool
Swimmer	2	100	10^6
Hopper-v1	3	1	10^6
Walker2d-v1	6	3	10^6
Humanoid	21	3	10^6

[‖] https://github.com/openai/baselines
[**] https://github.com/rll/rllab/tree/master/examples
[††] https://github.com/haarnoja/sac
[‡‡] https://github.com/haarnoja/softqlearning

F. Demos

Demos of our framework on a set of RL tasks can be accessed online via: https://sites.google.com/view/wgf4rl/ .

G. Algorithm Details

For completeness, we list the detailed algorithms for IP-WGF, DP-WGF and DP-WGF-V in Algorithms 1, 2 and 3, respectively.

Algorithm 1 DP-WGF

Require: $\mathcal{D} = \emptyset$; initialize $\boldsymbol{\theta}, \boldsymbol{\phi} \sim$ some (prior) distribution.
Target parameters: $\overline{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}, \overline{\boldsymbol{\phi}} \leftarrow \boldsymbol{\phi}$
for each epoch **do**
  for each t **do**
    % Collect expereince
    Sample an action $\mathbf{a}_t$ from policy $\pi^{\boldsymbol{\phi}}(\cdot|\,\mathbf{s}_t)$.
    Sample next state from the environment: $\mathbf{s}_{t+1} \sim p_{\mathbf{s}}(\mathbf{s}_{t+1}\,|\,\mathbf{s}_t, \mathbf{a}_t)$
    Save the new experience in the replay memory: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1}\}$
    % Sample from the replay memory
    $\{(\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)}, r_t^{(i)}, \mathbf{s}_{t+1}^{(i)})\}_{i=0}^N \sim \mathcal{D}$.
    % Update Q function
    Compute empirical values $\hat{V}^{\overline{\boldsymbol{\theta}}}(\mathbf{s}_{t+1}^{(i)})$
    Compute empirical gradient $\hat{\nabla}_{\boldsymbol{\theta}} J_Q(\boldsymbol{\theta})$
    Update $\boldsymbol{\theta}$ according to it using ADAM
    % Update policy
    Compute $W_2^2(\pi^{\overline{\boldsymbol{\phi}}}(\cdot|\,\mathbf{s}_t), \pi^{\boldsymbol{\phi}}(\cdot|\,\mathbf{s}_t))$,
    Compute empirical gradient $\hat{\nabla}_{\boldsymbol{\phi}} J_\pi(\boldsymbol{\phi})$
    Update prior policy parameters: $\overline{\boldsymbol{\phi}} \leftarrow \boldsymbol{\phi}$
    Update $\boldsymbol{\theta}$ according to it using ADAM
    % Update target
    Update target Q function parameters:
    $\overline{\boldsymbol{\theta}} \leftarrow \tau\boldsymbol{\theta} + (1-\tau)\overline{\boldsymbol{\theta}}$
  end for
end for

Algorithm 2 IP-WGF

Require: Initialize policy particles $\boldsymbol{\Theta} \sim$ some (prior) distribution as a Bayesian neural network.
for each iteration **do**
  Reset FIFO replay pool R
  for each timestep t in episodes **do**
    Sample $\mathbf{a}_t$ from $\pi_\phi(\cdot|\,\mathbf{s}_t)$
    Sample next state from the environment: $\mathbf{s}_{t+1} \sim p_{\mathbf{s}}(\mathbf{s}_{t+1}\,|\,\mathbf{s}_t, \mathbf{a}_t)$
    Save experience in to FIFO replay pool R:
    for each particles $\boldsymbol{\theta}^{(i)} \in \boldsymbol{\Theta}$ **do**
      Compute $W_2^2(\overline{\boldsymbol{\Theta}}^{(i)}, \boldsymbol{\Theta}^{(i)})$
      Compute empirical gradient $\nabla_{\boldsymbol{\theta}^{(i)}} J(\pi_{\boldsymbol{\theta}^{(i)}})$
      Save current particles $\overline{\boldsymbol{\theta}}^{(i)} \leftarrow \boldsymbol{\theta}^{(i)}$
      Update policy particle $\boldsymbol{\theta}^{(i)}$
    end for
  end for
end for

Algorithm 3 DP-WGF-V

Require: $\mathcal{D} = \emptyset$; initialize $\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi} \sim$ some (prior) distribution. Target parameters: $\overline{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}, \overline{\boldsymbol{\phi}} \leftarrow \boldsymbol{\phi}$
for each epoch **do**
  for each t **do**
    % Collect expereince
    Sample an action $\mathbf{a}_t$ from policy $\pi^{\boldsymbol{\phi}}(\cdot|\,\mathbf{s}_t)$.
    Sample next state from the environment: $\mathbf{s}_{t+1} \sim p_{\mathbf{s}}(\mathbf{s}_{t+1}\,|\,\mathbf{s}_t, \mathbf{a}_t)$
    Save the new experience in the replay memory: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1}\}$
    % Sample from the replay memory
    $\{(\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)}, r_t^{(i)}, \mathbf{s}_{t+1}^{(i)})\}_{i=0}^N \sim \mathcal{D}$.
    % Update Q function
    Compute empirical gradient $\hat{\nabla}_{\boldsymbol{\theta}} J_Q(\boldsymbol{\theta})$
    Update $\boldsymbol{\theta}$ according to it using ADAM
    % Update value function
    Compute empirical gradient $\hat{\nabla}_{\boldsymbol{\psi}} J_V(\boldsymbol{\psi})$
    Update $\boldsymbol{\psi}$ according to it using ADAM
    % Update policy
    Compute $W_2^2(\pi^{\overline{\boldsymbol{\phi}}}(\cdot|\,\mathbf{s}_t), \pi^{\boldsymbol{\phi}}(\cdot|\,\mathbf{s}_t))$,
    Compute empirical gradient $\hat{\nabla}_{\boldsymbol{\phi}} J_\pi^\phi$
    Update prior policy parameters: $\overline{\boldsymbol{\phi}} \leftarrow \boldsymbol{\phi}$
    Update $\boldsymbol{\phi}$ according to it using ADAM
    % Update target
    Update target value parameters:
    $\overline{\boldsymbol{\psi}} \leftarrow \tau\boldsymbol{\psi} + (1-\tau)\overline{\boldsymbol{\psi}}$
  end for
end for