Learning to Score Behaviors for Guided Policy Optimization

Aldo Pacchiano [*1] **Jack Parker-Holder** [*2] **Yunhao Tang** [*3] **Anna Choromanska** [4] **Krzysztof Choromanski** [5]
Michael I. Jordan [1]

Abstract

We introduce a new approach for comparing reinforcement learning policies, using Wasserstein distances (WDs) in a newly defined latent behavioral space. We show that by utilizing the dual formulation of the WD, we can learn score functions over policy behaviors that can in turn be used to lead policy optimization towards (or away from) (un)desired behaviors. Combined with smoothed WDs, the dual formulation allows us to devise efficient algorithms that take stochastic gradient descent steps through WD regularizers. We incorporate these regularizers into two novel on-policy algorithms, Behavior-Guided Policy Gradient and Behavior-Guided Evolution Strategies, which we demonstrate can outperform existing methods in a variety of challenging environments. We also provide an open source demo[1].

1. Introduction

One of the key challenges in reinforcement learning (RL) is to efficiently incorporate the behaviors of learned policies into optimization algorithms (Lee & Popovic, 2010; Meyerson et al., 2016; Conti et al., 2018). The fundamental question we aim to shed light on in this paper is:

What is the right measure of similarity between two policies acting on the same underlying MDP and how can we devise algorithms to leverage this information for RL?

In simple terms, the main thesis motivating the methods we propose is that:

Two policies may perform similar actions at a local level but result in very different global behaviors.

We propose to define *behaviors* via so-called Behavioral Policy Embeddings (henceforth referred to as Policy Embeddings), which can be both on policy and off policy.

On policy embeddings are achieved via what we call Behavioral Embeddings Maps (BEMs) - functions mapping trajectories of a policy into a latent behavioral space representing trajectories in a compact way. We define the policy embedding as the pushforward distributions over trajectory embeddings as a result of applying a BEM to the policy's trajectories. Importantly, two policies with distinct distributions over trajectories may result in the same probabilistic embedding. Off policy embeddings in contrast correspond to state and policy evaluation pairs resulting of evaluating the policy on states sampled from a probing state distribution that can be chosen independently from the policy.

Both embedding mechanisms result in probabilistic Policy Embeddings, which allow us to identify a policy with a distribution with support on an embedding space. Policy Embeddings provide us a way to rigorously define dissimilarity between policies. We do this by equipping them with metrics defined on the manifold of probability measures, namely a class of Wasserstein distances (WDs, (Villani, 2008)). There are several reasons for choosing WDs:

- **Flexibility**. We can use any cost function between embeddings of trajectories, allowing the distance between policy embeddings to arise organically from an interpretable distance between embedding points.
- **Non-injective BEMs**. Different trajectories may be mapped to the same embedding point (for example in the case of the last-state embedding). This precludes the use of likelihood-based distances such as the KL divergence (Kullback & Leibler, 1951), which we discuss in Section 6.
- **Behavioral Test Functions**. Solving the dual formulation of the WD objective yields a pair of test functions over the space of embeddings, used to score trajectories or state policy pairs (see: Sec. 5.2).

[*]Equal contribution [1]University of California Berkeley [2]University of Oxford [3]Columbia University [4]New York University [5]Google Brain Robotics. Correspondence to: Aldo Pacchiano <pacchiano@berkeley.edu>, Jack Parker-Holder <jackph@robots.ox.ac.uk>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

[1]Available at https://github.com/behaviorguidedRL/BGRL. We emphasize this is the exact code from our experiments, but a demo to build intuition and clarify our methods.

The Behavioral Test Functions, underpin all our algorithms, directing optimization towards desired behaviors. To learn them, it suffices to define the embedding type and BEM (if required) and the cost function between points in the resulting behavioral manifold. To mitigate the computational burden of computing WDs, we rely on their entropy-regularized formulations. This allows us to update the learned test functions in a computationally efficient manner via stochastic gradient descent (SGD) on a Reproducing Kernel Hilbert Space (RKHS). We develop a novel method for stochastic optimal transport based on random feature maps (Rahimi & Recht, 2008) to produce compact and memory-efficient representations of learned behavioral test functions. Finally, having laid the groundwork for comparing policies via behavior-driven trajectory or state-policy pairs scores, we address our core question by introducing two new on-policy RL algorithms:

- **Behavior Guided Policy Gradients (BGPG)**: We propose to replace the KL-based trust region from (Schulman et al., 2015) with a WD-based in the behavior space.
- **Behavior Guided Evolution Strategies (BGES)**: BGES improves on Novelty Search (Conti et al., 2018) by jointly optimizing for reward and *novelty* using the WD in the behavior space.

We also demonstrate a way to harness our methodology for imitation and repulsion learning (Section 5.2), showing the universality of the proposed techniques.

2. Motivating Behavior-Guided Reinforcement Learning

Throughout this paper we prompt the reader to think of a policy as a distribution over its behaviors, induced by the policy's (possibly stochastic) map from state to actions and the unknown environment dynamics. We care about summarizing (or embedding) behaviors into succinct representations that can be compared with each other (via a cost/metric). These comparisons arise naturally when answering questions such as: Has a given trajectory achieved a certain level of reward? Has it visited a certain part of the state space? We think of these summaries or embeddings as characterizing the behavior of the trajectory or relevant state policy-pairs. We formalize these notions in Section 3.

We show that by identifying policies with the embedding distributions that result of applying the embedding function (summary) to their trajectories, and combining this with the provided cost metric, we can induce a topology over the space of policies given by the Wasserstein distance over their embedding distributions. The methods we propose can be thought of as ways to leverage this "behavior" geometry for a variety of downstream applications such as policy optimization and imitation learning.

This topology emerges naturally from the sole definition of an embedding map (behavioral summary) and a cost function. Crucially these choices occur in the semantic space of behaviors as opposed to parameters or visitation frequencies[2]. One of the advantages of choosing a Wasserstein geometry is that non-surjective trajectory embedding maps are allowed. This is not possible with a KL induced one (in non-surjective cases, computing the likelihood ratios in the KL definition is in general intractable). In Sections 4 and 5 we show that in order to get a handle on this geometry, we can use the dual formulation of the Wasserstein distance to learn functions (Behavioral Test Functions) that can provide scores on trajectories which then can be added to the reward signal (in policy optimization) or used as a reward (in Imitation Learning).

In summary, by defining an embedding map of trajectories into a behavior embedding space equipped with a metric[3], our framework allows us to learn "reward" signals (Behavioral Test Functions) that can serve to steer policy search algorithms through the "behavior geometry" either in conjunction with a task specific reward (policy optimization) or on their own (e.g. Imitation Learning). We develop versions of on policy RL algorithms which we call Behavior Guided Policy Gradient (BGPG) and Behavior Guided Evolution Strategies (BGES) that enhance their baseline versions by the use of learned Behavioral Test Functions. Our experiments in Section 7 show this modification is useful. We also provide a simple example for repulsion learning and Imitation Learning, where we only need access to an expert's embedding. Our framework also has obvious applications to safety, learning policies that avoid undesirable behaviors.

A final important note is that in this work we only consider simple heuristics for the embeddings, as used in the existing literature. For BGES, these embeddings are those typically used in Quality Diversity algorithms (Pugh et al., 2016), while for BGPG we reinterpret the action distribution currently used in KL-based trust regions (Schulman et al., 2017; 2015). We emphasize the focus of this paper is on introducing the framework to score these behaviors to guide policy optimization.

3. Defining Behavior in Reinforcement Learning

A Markov Decision Process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{R})$. Here $\mathcal{S}$ and $\mathcal{A}$ stand for the sets of states and actions respectively, such that for $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$: $\mathrm{P}(s'|a, s)$ is the probability that the system/agent transitions from s

[2]If we choose an appropriate embedding map our framework handles visitation frequencies as well.

[3]The embedding space can be discrete or continuous and the metric need not be smooth, and can be for example a simple discrete $\{0, 1\}$ valued criterion

to s' given action a and $\mathrm{R}(s', a, s)$ is a reward obtained by an agent transitioning from s to s' via a. A policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$ is a (possibly randomized) mapping (parameterized by $\theta \in \mathbb{R}^d$) from $\mathcal{S}$ to $\mathcal{A}$. Let $\Gamma = \{\tau = s_0, a_0, r_0, \cdots s_H, a_H, r_H \text{ s.t. } s_i \in \mathcal{S}, a_i \in \mathcal{A}, r_i \in \mathbb{R}\}$ be the set of possible trajectories enriched by sequences of partial rewards under some policy π. The undiscounted reward function $\mathcal{R} : \Gamma \to \mathbb{R}$ (which expectation is to be maximized by optimizing θ) satisfies $\mathcal{R}(\tau) = \sum_{i=0}^{H} r_i$, where $r_i = R(s_{i+1}, a_i, s_i)$.

3.1. Behavioral Embeddings

In this work we identify a policy with what we call a Policy Embedding. We focus on two types of Policy Embeddings both of which are probabilistic in nature, on policy and off policy embeddings, the first being trajectory based and the second ones state-based.

3.1.1. On Policy Embeddings

We start with a Behavioral Embedding Map (BEM), $\Phi : \Gamma \to \mathcal{E}$, mapping trajectories to embeddings (Fig. 1), where $\mathcal{E}$ can be seen as a behavioral manifold. On Policy Embeddings can be for example: a) **State-Based**, such as the final state $\Phi_1(\tau) = s_H$ b) **Action-based:** such as the concatenation of actions $\Phi_4(\tau) = [a_0, ..., a_H]$ or c) **Reward-based:** the total reward $\Phi_5(\tau) = \sum_{t=0}^{H} r_t$, reward-to-go vector $\Phi_6(\tau) = \sum_{t=0}^{H} r_t \left(\sum_{i=0}^{t} e_i\right)$ (where $e_i \in \mathbb{R}^{H+1}$ is a one-hot vector corresponding to i with dimension index from 0 to H). Importantly, the mapping does not need to be surjective, as we see on the example of the final state embedding.

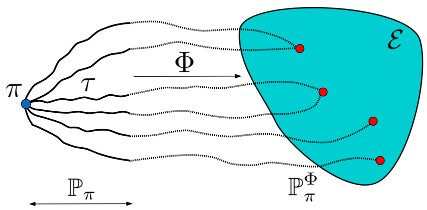


Figure 1: **Behavioral Embedding Maps (BEMs)** map trajectories to points in the behavior embedding space $\mathcal{E}$. Two trajectories may map to the same point in $\mathcal{E}$.

Given a policy π, we let $\mathbb{P}_\pi$ denote the distribution induced over the space of trajectories Γ and by $\mathbb{P}_\pi^\Phi$ the corresponding pushforward distribution on $\mathcal{E}$ induced by Φ. We call P_π^Φ the policy embeddings of a policy π. A policy π can be fully characterized by the distribution $\mathbb{P}_\pi$ (see: Fig. 1).

Additionally, we require $\mathcal{E}$ to be equipped with a metric (or cost function) $C : \mathcal{E} \times \mathcal{E} \to \mathbb{R}$. Given two trajectories τ_1, τ_2 in Γ, $C(\Phi(\tau_1), \Phi(\tau_2))$ measures how different these trajectories are in the behavior space. We note that some embeddings are only for the tabular case ($|\mathcal{S}|, |\mathcal{A}| < \infty$) while others are universal.

3.1.2. Off Policy Embeddings

Let $\mathbb{P}_\mathcal{S}$ be some "probe" distribution over states $\mathcal{S}$ and π be a policy. We define $\mathbb{P}_\pi^{\Phi_\mathcal{S}}$ to be the distribution of pairs $(s, \pi(s))$ for $s \sim \mathbb{P}_\mathcal{S}$. We identify $\mathcal{E}$ with the product space $\mathcal{S} \times \Delta_\mathcal{A}$ (where $\Delta_\mathcal{A}$ denotes the set of distributions over $\mathcal{A}$) endowed with an appropriate metric $C : \mathcal{E} \times \mathcal{E} \to \mathbb{R}$.In our experiments we identify $\mathcal{C}$ with the l_2 norm over $\mathcal{E}$ and $\mathbb{P}_\mathcal{S}$ with a mechanism that samples states from a buffer of visited states. We only add an $\mathcal{S}$ to the notation for $\mathbb{P}_\pi^{\Phi_\mathcal{S}}$ when distinguishing from on-policy embeddings is needed.

This definition allows the "probing" distribution $\mathbb{P}_\mathcal{S}$ to be off policy, independent of the policy at hand. If C is a norm and $\mathbb{P}_\mathcal{S}$ has mass only in user-relevant areas of the state space, a WD of zero between two policies (whose embeddings use the same probing distribution) implies they behave equally where the user cares. Our off Policy Embeddings are of the form $(s, \pi(s))$ but other choices are valid.

4. Wasserstein Distance & Optimal Transport Problem

Let μ, ν be (Radon) probability measures over domains $\mathcal{X} \subseteq \mathbb{R}^m, \mathcal{Y} \subseteq \mathbb{R}^n$ and let $\mathcal{C} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be a cost function. For $\gamma > 0$, a *smoothed* Wasserstein Distance is defined as:

$$\mathrm{WD}_\gamma(\mu, \nu) := \min_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X} \times \mathcal{Y}} C(\mathbf{x}, \mathbf{y}) d\pi(\mathbf{x}, \mathbf{y}) + \Sigma, \quad (1)$$

where $\Sigma = \gamma \mathrm{KL}(\pi|\xi)$, $\Pi(\mu, \nu)$ is the space of couplings (joint distributions) over $\mathcal{X} \times \mathcal{Y}$ with marginal distributions μ and ν, $\mathrm{KL}(\cdot|\cdot)$ denotes the KL divergence between distributions π and ξ with support $\mathcal{X} \times \mathcal{Y}$ defined as: $\mathrm{KL}(\pi|\xi) = \int_{\mathcal{X} \times \mathcal{Y}} \left(\log\left(\frac{d\pi}{d\xi}(\mathbf{x}, \mathbf{y})\right)\right) d\pi(\mathbf{x}, \mathbf{y})$ and ξ is a reference measure over $\mathcal{X} \times \mathcal{Y}$. When the cost is an ℓ_p distance and $\gamma = 0$, WD_γ is also known as the Earth mover's distance and the corresponding optimization problem is known as the *optimal transport problem* (OTP).

4.1. Wasserstein Distance: Dual Formulation

We will use smoothed WDs to derive efficient regularizers for RL algorithms. To arrive at this goal, we first need to consider the dual form of Equation 1. Under the subspace topology (Bourbaki, 1966) for $\mathcal{X}$ and $\mathcal{Y}$, let $\mathcal{C}(\mathcal{X})$ and $\mathcal{C}(\mathcal{Y})$ denote the space of continuous functions over $\mathcal{X}$ and $\mathcal{Y}$ respectively. The choice of the subspace topology ensures our discussion encompasses the discrete case.

Let $C : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be a cost function, interpreted as the "ground cost" to move a unit of mass from x to y. Define $\mathbb{I}$ as the function outputting values of its input predicates. Using Fenchel duality, we can obtain the following dual formulation of the problem in Eq. 1:

$$\mathrm{WD}_\gamma(\mu, \nu) = \max_{\lambda_\mu \in \mathcal{C}(\mathcal{X}), \lambda_\nu \in \mathcal{C}(\mathcal{Y})} \Psi(\lambda_\mu, \lambda_\nu), \quad (2)$$

where $\Psi(\lambda_\mu, \lambda_\nu) = \int_{\mathcal{X}} \lambda_\mu(\mathbf{x}) d\mu(\mathbf{x}) - \int_{\mathcal{Y}} \lambda_\nu(\mathbf{y}) d\nu(\mathbf{y}) - E_C(\lambda_\mu, \lambda_\nu)$ and the damping term $E_C(\lambda_\mu, \lambda_\nu)$ equals:

$$E_C(\lambda_\mu, \lambda_\nu) = \mathbb{I}(\gamma > 0) \int_{\mathcal{X}\times\mathcal{Y}} \rho(\mathbf{x}, \mathbf{y}) d\xi(\mathbf{x}, \mathbf{y}) + \mathbb{I}(\gamma = 0)\mathbb{I}(\mathcal{A}) \tag{3}$$

for $\rho(\mathbf{x}, \mathbf{y}) = \gamma \exp(\frac{\lambda_\mu(\mathbf{x}) - \lambda_\nu(\mathbf{y}) - C(\mathbf{x},\mathbf{y})}{\gamma})$ and $\mathcal{A} = [(\lambda_\mu, \lambda_\nu) \in \{(u, v) \text{ s.t. } \forall (\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y} : u(\mathbf{x}) - v(\mathbf{y}) \leq C(\mathbf{x}, \mathbf{y})\}]$.

We will set the damping distribution $d\xi(\mathbf{x}, \mathbf{y}) \propto 1$ for discrete domains and $d\xi(\mathbf{x}, \mathbf{y}) = d\mu(\mathbf{x})d\nu(\mathbf{y})$ otherwise.

If $\lambda_\mu^*, \lambda_\nu^*$ are the functions achieving the maximum in Eq. 2, and γ is sufficiently small then $\mathrm{WD}_\gamma(\mu, \nu) \approx \mathbb{E}_\mu [\lambda_\mu^*(\mathbf{x})] - \mathbb{E}_\nu [\lambda_\nu^*(\mathbf{y})]$, with equality when $\gamma = 0$. When for example $\gamma = 0$, $\mathcal{X} = \mathcal{Y}$, and $C(x, x) = 0$ for all $x \in \mathcal{X}$, it is easy to see $\lambda_\mu^*(x) = \lambda_\nu^*(x) = \lambda^*(x)$ for all $x \in \mathcal{X}$. In this case the difference between $\mathbb{E}_\mu [\lambda^*(\mathbf{x})]$ and $\mathbb{E}_\mu [\lambda^*(\mathbf{y})]$ equals the WD. In other words, the function λ^* gives higher scores to regions of the space $\mathcal{X}$ where μ has more mass. This observation is key to the success of our algorithms in guiding optimization towards desired behaviors.

4.2. Computing λ_μ^* and λ_ν^*

We combine several techniques to make the optimization of objective from Eq. 2 tractable. First, we replace $\mathcal{X}$ and $\mathcal{Y}$ with the functions from a RKHS corresponding to universal kernels (Micchelli et al., 2006). This is justified since those function classes are dense in the set of continuous functions of their ambient spaces. In this paper we choose the RBF kernel and approximate it using random Fourier feature maps (Rahimi & Recht, 2008) to increase efficiency. Consequently, the functions λ learned by our algorithms have the following form: $\lambda(\mathbf{x}) = (\mathbf{p}^\lambda)^\top \phi(\mathbf{x})$, where ϕ is a random feature map with m standing for the number of random features and $\mathbf{p}^\lambda \in \mathbb{R}^m$. For the RBF kernel, ϕ is defined as follows: $\phi(\mathbf{z}) = \frac{1}{\sqrt{m}} \cos(\mathbf{G}\mathbf{z} + \mathbf{b})$ for $\mathbf{z} \in \mathbb{R}^d$, where $\mathbf{G} \in \mathbb{R}^{m \times d}$ is a Gaussian with iid entries taken from $\mathcal{N}(0, 1)$, $b \in \mathbb{R}^m$ with iid b_is such that $b_i \sim \mathrm{Unif}[0, 2\pi]$ and the cos function acts elementwise.

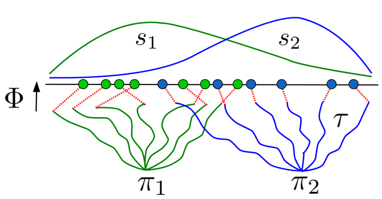


Figure 2: Two policies π_1 (green) and π_2 (blue) whose BEMs map trajectories to points in the real line.

Henceforth, when we refer to optimization over λ, we mean optimizing over corresponding dual vectors $\mathbf{p}^\lambda$ associated with λ. We can solve for the optimal dual functions by running Stochastic Gradient Descent (SGD) over the dual objective in Eq. 2. Algorithm 1 is the random features equivalent of Algorithm 3 in (Genevay et al., 2016). Given input kernels κ, ℓ and a fresh sample $(x_t, y_t) \sim \mu \bigotimes \nu$ the SGD step w.r.t. the current iterates $\mathbf{p}_{t-1}^\mu, \mathbf{p}_{t-1}^\nu$ satisfies:

Algorithm 1 Random Features Wasserstein SGD

Input: kernels κ, ℓ over $\mathcal{X}, \mathcal{Y}$ respectively with corresponding random feature maps ϕ_κ, ϕ_ℓ, smoothing parameter γ, gradient step size α, number of optimization rounds M, initial dual vectors $\mathbf{p}_0^\mu, \mathbf{p}_0^\nu$.

for $t = 0, \cdots, M$ **do**

 1. Sample $(x_t, y_t) \sim \mu \bigotimes \nu$.
 2. Update: $\binom{\mathbf{p}_t^\mu}{\mathbf{p}_t^\nu}$ using Equation 4.

Return: $\mathbf{p}_M^\mu, \mathbf{p}_M^\nu$.

$$F(\mathbf{p}_1, \mathbf{p}_2, x, y) = \exp\left(\frac{(\mathbf{p}_1)^\top \phi_\kappa(x) - (\mathbf{p}_2)^\top \phi_\ell(x) - C(x, y)}{\gamma}\right)$$

$$\begin{pmatrix} \mathbf{p}_{t+1}^\mu \\ \mathbf{p}_{t+1}^\nu \end{pmatrix} = \begin{pmatrix} \mathbf{p}_t^\mu \\ \mathbf{p}_t^\nu \end{pmatrix} + (1 - F(\mathbf{p}_t^\mu, \mathbf{p}_t^\nu, x_t, y_t))\, v_t, \tag{4}$$

where $v_t = \frac{\alpha}{\sqrt{t}}(\phi_\kappa(x_t), -\phi_\ell(y_t))^\top$. An explanation and proof of these formulae is in Lemma C.2 in the Appendix. If $\mathbf{p}_*^\mu, \mathbf{p}_*^\nu$ are the optimal dual vectors, $p_* = (\mathbf{p}_*^\mu, \mathbf{p}_*^\nu)^\top$, $(x_1, y_1), \cdots, (x_k, y_k) \overset{\text{i.i.d}}{\sim} \mu \bigotimes \nu$, $\mathbf{v}_i^{\kappa,\ell} = (\phi_\kappa(x_i), -\phi_\ell(y_i))^\top$ for all i, and $\hat{\mathbb{E}}$ denotes the empirical expectation over the k samples $\{(x_i, y_i)\}_{i=1}^k$, Algorithm 1 can be used to get an estimator of $\mathrm{WD}_\gamma(\mu, \nu)$ as:

$$\widehat{\mathrm{WD}}_\gamma(\mu, \nu) = \hat{\mathbb{E}}\left[\langle \mathbf{p}_*, \mathbf{v}_i^{\kappa,\ell} \rangle - \frac{F(\mathbf{p}_*^\mu, \mathbf{p}_*^\nu, x_i, y_i)}{\gamma}\right] \tag{5}$$

5. Behavior-Guided Reinforcement Learning

We explain now how to get practical algorithms based on the presented methods. Denote by π_θ a policy parameterized by $\theta \in \mathbb{R}^d$. The goal of policy optimization algorithms is to find a policy maximizing, as a function of the policy parameters, the expected total reward $\mathcal{L}(\theta) := \mathbb{E}_{\tau \sim \mathbb{P}_{\pi_\theta}} [\mathcal{R}(\tau)]$.

5.1. Behavioral Test Functions

If $C : \mathcal{E} \times \mathcal{E} \to \mathbb{R}$ is a cost function defined over behavior space $\mathcal{E}$, and π_1, π_2 are two policies, then in the case of **On-Policy** Embeddings:

$$\mathrm{WD}_\gamma(\mathbb{P}_{\pi_1}^\Phi, \mathbb{P}_{\pi_2}^\Phi) \approx \mathop{\mathbb{E}}_{\tau \sim \mathbb{P}_{\pi_1}} [\lambda_1^*(\Phi(\tau))] - \mathop{\mathbb{E}}_{\tau \sim \mathbb{P}_{\pi_2}} [\lambda_2^*(\Phi(\tau))],$$

where λ_1^*, λ_2^* are the optimal dual functions. The maps $s_1 := \lambda_1^* \circ \Phi : \Gamma \to \mathbb{R}$ and $s_2 := \lambda_2^* \circ \Phi : \Gamma \to \mathbb{R}$ define score functions over the space of trajectories. If γ is close to zero, the score function s_i gives higher scores to trajectories from π_i whose behavioral embedding is common under π_i but rarely appears under π_j for $j \neq i$ (Fig. 2). In the case of **Off-Policy** Embeddings:

$$\mathrm{WD}_\gamma(\mathbb{P}_{\pi_1}^{\Phi_\mathcal{S}}, \mathbb{P}_{\pi_2}^{\Phi'_\mathcal{S}}) \approx \mathop{\mathbb{E}}_{S \sim \mathbb{P}_\mathcal{S}} [\lambda_1^*(S, \pi_1(S))] - \mathop{\mathbb{E}}_{S \sim \mathbb{P}'_\mathcal{S}} [\lambda_2^*(S, \pi_2(S))],$$

where λ_1^*, λ_2^* are maps from state policy pairs $(S, \pi_1(S))$ to scores, and $\mathbb{P}_{\mathcal{S}}, \mathbb{P}'_{\mathcal{S}}$ are probing distributions.

5.2. Repulsion and Imitation Learning

To illustrate the intuition behind behavioral test functions and on policy embeddings, we introduce an algorithm for multi-policy repulsion learning based on our framework. Algorithm 2 maintains two policies $\pi^{\mathbf{a}}$ and $\pi^{\mathbf{b}}$.

Algorithm 2 Behvaior-Guided Repulsion Learning

Input: $\beta, \eta > 0$, $M \in \mathbb{N}$

Initialize: Initial stochastic policies $\pi_0^{\mathbf{a}}, \pi_0^{\mathbf{b}}$, parametrized by $\theta_0^{\mathbf{a}}, \theta_0^{\mathbf{b}}$ respectively, Behavioral Test Functions $\lambda_1^{\mathbf{a}}, \lambda_2^{\mathbf{b}}$

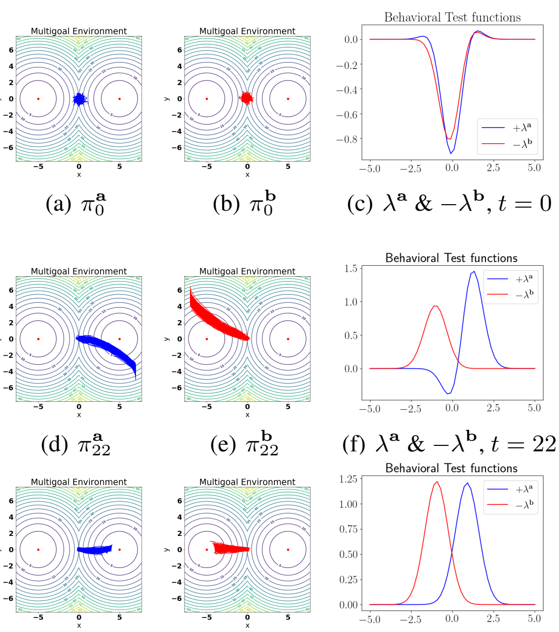
for $t = 1, \ldots, T$ **do**

1. Collect $\{\tau_i^{\mathbf{a}}\}_{i=1}^M \sim \mathbb{P}_{\pi_{t-1}^{\mathbf{a}}}$ and $\{\tau_i^{\mathbf{b}}\}_{i=1}^M \sim \mathbb{P}_{\pi_{t-1}^{\mathbf{b}}}$.
2. Form $\tilde{R}_{\mathbf{c}}(\tau_1, \tau_2)$ for $\mathbf{c} \in \{\mathbf{a}, \mathbf{b}\}$ using Equation 6.
3. For $\mathbf{c} \in \{\mathbf{a}, \mathbf{b}\}$ and $(\tau_1, \tau_2) \sim \{\tau_i^{\mathbf{a}}\}_{i=1}^M \times \{\tau_i^{\mathbf{b}}\}_{i=1}^M$ use REINFORCE (Williams, 1992) to perform update:

$$\theta_t^{\mathbf{c}} = \theta_{t-1}^{\mathbf{c}} + \eta \nabla_\theta \tilde{R}_{\mathbf{c}}(\tau_1, \tau_2)$$

5. Update $\lambda_1^{\mathbf{a}}, \lambda_2^{\mathbf{b}}$ with $\{\tau_i^{\mathbf{a}}, \tau_i^{\mathbf{b}}\}_{i=1}^M$ via Algorithm 1.

Each policy is optimized by taking a policy gradient step (using the REINFORCE gradient estimator (Williams, 1992)) to optimize surrogate rewards $\tilde{\mathcal{R}}_{\mathbf{a}}$ and $\tilde{\mathcal{R}}_{\mathbf{b}}$.



(a) $\pi_0^{\mathbf{a}}$ (b) $\pi_0^{\mathbf{b}}$ (c) $\lambda^{\mathbf{a}}$ & $-\lambda^{\mathbf{b}}$, $t = 0$

(d) $\pi_{22}^{\mathbf{a}}$ (e) $\pi_{22}^{\mathbf{b}}$ (f) $\lambda^{\mathbf{a}}$ & $-\lambda^{\mathbf{b}}$, $t = 22$

(g) $\pi_{118}^{\mathbf{a}}$ (h) $\pi_{118}^{\mathbf{b}}$ (i) $\lambda^{\mathbf{a}}$ & $-\lambda^{\mathbf{b}}$, $t = 118$

Figure 3: a) and b) Initial state of policies $\pi^{\mathbf{a}}, \pi^{\mathbf{b}}$ and Test functions $\lambda^{\mathbf{a}}, \lambda^{\mathbf{b}}$. d)-i) Policy evolution and Test Functions.

These combine the signal from the task's reward function $\mathcal{R}$ and the repulsion score encoded by the input BEM Φ and behavioral test functions $\lambda^{\mathbf{a}}$ and $\lambda^{\mathbf{b}}$:

$$\tilde{R}_{\mathbf{c}}(\tau_{\mathbf{a}}, \tau_{\mathbf{b}}) = \mathcal{R}(\tau_{\mathbf{c}}) + \beta \widehat{\mathrm{WD}}_\gamma(\mathbb{P}^\Phi_{\pi^{\mathbf{a}}}, \mathbb{P}^\Phi_{\pi^{\mathbf{a}}}), \mathbf{c} \in \{\mathbf{a}, \mathbf{b}\} \quad (6)$$

We test Algorithm 2 on an environment consisting of a particle that needs to reach one of two goals on the plane. Policies output a velocity vector and stochasticity is achieved by adding Gaussian noise to it. The embedding Φ maps trajectories τ to their mean displacement along the $x-$axis. Fig. 3 shows how the policies' behavior evolves throughout optimization and how the Test Functions guide the optimization by favouring the two policies to be far apart. The experiment details are in the Appendix (Section B.4). A related guided trajectory scoring approach to imitation learning is explored in Appendix B.3.

5.3. Algorithms

We propose to solve a WD-regularized objective to tackle behavior-guided policy optimization. All of our algorithms hinge on trying to maximize an objective of the form:

$$F(\theta) = \mathcal{L}(\theta) + \beta \mathrm{WD}_\gamma(\mathbb{P}^\Phi_{\pi_\theta}, \mathbb{P}^\Phi_{\mathrm{b}}), \quad (7)$$

where $\mathbb{P}^\Phi_{\mathrm{b}}$ is a base distribution[4] over behavioral embeddings (possibly dependent on θ) and $\beta \in \mathbb{R}$ could be positive or negative. Although the base distribution $\mathbb{P}^\Phi_{\mathrm{b}}$ could be arbitrary, our algorithms will instantiate $\mathbb{P}^\Phi_{\mathrm{b}} = \frac{1}{|\mathcal{S}|} \cup_{\pi' \in \mathcal{S}} \mathbb{P}^\Phi_{\pi'}$ for some family of policies $\mathcal{S}$ (possibly satisfying $|\mathcal{S}| = 1$) we want the optimization to attract to / repel from.

In order to compute approximate gradients for F, we rely on the dual formulation of the WD. After substituting the composition maps resulting from Eq. 5.1 into Eq. 7, we obtain, for **on-policy** embeddings:

$$F(\theta) \approx \mathbb{E}_{\tau \sim \mathbb{P}_{\pi_\theta}}[\mathcal{R}(\tau) + \beta s_1(\tau)] - \beta \mathbb{E}_{\phi \sim \mathbb{P}^\Phi_{\mathrm{b}}}[\lambda_2^*(\phi)], \quad (8)$$

where $s_1 : \Gamma \to \mathbb{R}$ equals $s_1 = \lambda_1^* \circ \Phi$, the Behavioral Test Function of policy π_θ and λ_2^* is the optimal dual function of embedding distribution $\mathbb{P}^\Phi_{\mathbf{b}}$. Consequently $\nabla_\theta F(\theta) \approx \nabla_\theta \mathbb{E}_{\tau \sim \mathbb{P}_{\pi_\theta}}[\mathcal{R}(\tau) + \beta s_1(\tau)]$. We learn a score function s_1 over trajectories that can guide our optimization by favoring those trajectories that show desired global behaviors. For **off-policy** embeddings, with state probing distributions $\mathbb{P}_{\mathcal{S}}$ and $\mathbb{P}^b_{\mathcal{S}}$ the analogous to Equation 9 is:

$$F(\theta) \approx \mathbb{E}_{\tau \sim \mathbb{P}_{\pi_\theta}}[\mathcal{R}(\tau)] + \beta \mathbb{E}_{(S, \pi_\theta(S)) \sim \mathbb{P}^{\Phi_{\mathcal{S}}}}[\lambda_1^*(S, \pi_\theta(S))] - \beta \mathbb{E}_{(S, \pi_b(S)) \sim \mathbb{P}^{\Phi_{\mathcal{S}}}_{\mathrm{b}}}[\lambda_2^*(S, \pi_b(S))], \quad (9)$$

Consequently, if $\mathbb{P}^{\Phi_{\mathcal{S}}}_{\mathrm{b}}$ is independent from θ:

$$\nabla_\theta F(\theta) \approx \nabla_\theta \mathbb{E}_{\tau \sim \mathbb{P}_{\pi_\theta}}[\mathcal{R}(\tau)] + \beta \mathbb{E}_{s \sim \mathbb{P}_{\mathcal{S}}}[\nabla_\theta \lambda_1^*(s, \pi_\theta(s))].$$

Eq. 8 and 9 are approximations to the true objective from Eq. 7 whenever $\gamma > 0$. In practice, the entropy regularization requires a damping term $E_C(\lambda_1^*, \lambda_2^*)$ as defined in Equation 3. If $\xi(\mathbb{P}^\Phi_{\pi_\theta}, \mathbb{P}^\Phi_{\mathrm{b}})$ is the damping joint distribution of

[4] Possibly using off policy embeddings.

choice and $\rho(\phi_1, \phi_2) = \gamma \exp\left(\frac{\lambda_{\pi_\theta}(\phi_1) - \lambda_{\mathrm{b}}(\phi_2) - C(\phi_1, \phi_2)}{\gamma}\right)$ (for off policy embeddings ϕ is a state policy pair $(S, \pi(S))$), the damping term equals: $\mathbb{E}_{\phi_1, \phi_2 \sim \xi(\mathbb{P}^{\Phi}_{\pi_\theta}, \mathbb{P}^{\Phi}_{\mathrm{b}})}\left[\rho(\phi_1, \phi_2)\right]$. Gradients ∇_θ through E_C can be derived using a similar logic as the gradients above. When the embedding space $\mathcal{E}$ is not discrete and $\mathbb{P}^{\Phi}_{\mathrm{b}} = \mathbb{P}^{\Phi}_{\pi}$ for some policy π, we let $\xi(\mathbb{P}^{\Phi}_{\pi_\theta}, \mathbb{P}^{\Phi}_{\mathrm{b}}) = \mathbb{P}^{\Phi}_{\pi_\theta} \bigotimes \mathbb{P}^{\Phi}_{\pi}$, otherwise $\xi(\mathbb{P}^{\Phi}_{\pi_\theta}, \mathbb{P}^{\Phi}_{\mathrm{b}}) = \frac{1}{|\mathcal{E}|^2}\mathbb{1}$, a uniform distribution over $\mathcal{E} \times \mathcal{E}$.

All of our methods perform a version of alternating SGD optimization: we take certain number of SGD steps over the internal dual Wasserstein objective, followed by more SGD steps over the outer objective having fixed the test functions.

We consider two approaches to optimizing this objective. Behavior-Guided Policy Gradient (BGPG) explores in the action space as in policy gradient methods (Schulman et al., 2015; 2017), while Behavior-Guided Evolution Strategies (BGES) considers a black-box optimization problem as in Evolution Strategies (ES, (Salimans et al., 2017)).

5.4. Behavior-Guided Policy Gradient (BGPG)

Here we present the Behavior-Guided Policy Gradient (BGPG) algorithm (Alg. 3). Specifically, we maintain a stochastic policy π_θ and compute policy gradients as in prior work (Schulman et al., 2015).

Algorithm 3 Behavior-Guided Policy Gradient

Input: Initialize stochastic policy π_0 parametrized by θ_0, $\beta < 0, \eta > 0$, $M \in \mathbb{N}$

for $t = 1, \ldots, T$ **do**

1. Run π_{t-1} in the environment to get advantage values $A^{\pi_{t-1}}(s, a)$ and trajectories $\{\tau_i^{(t)}\}_{i=1}^{M}$
2. Update policy and test functions via several alternating policy gradient steps over $F(\theta)$.
3. Use samples from $\mathbb{P}_{\pi_{t-1}} \bigotimes \mathbb{P}_{\pi_\theta}$ and Algorithm 1 to update λ_1, λ_2 and take SGA step $\theta_t = \theta_{t-1} + \eta \hat{\nabla}_\theta \hat{F}(\theta_{t-1})$

For **on-policy** embeddings the objective function $F(\theta)$ takes the form:

$$F(\theta) = \mathop{\mathbb{E}}_{\tau_1, \tau_2 \sim \mathbb{P}_{\pi_{t-1}} \bigotimes \mathbb{P}_{\pi_\theta}}\left[\hat{R}(\tau_1, \tau_2)\right], \qquad (10)$$

where $\hat{R}(\tau_1, \tau_2) = \sum A^{\pi_{t-1}}(s_i, a_i)\frac{\pi_\theta(a_i|s_i)}{\pi_{t-1}(a_i|s_i)} + \widehat{\mathrm{WD}}_\gamma(\mathbb{P}^{\Phi}_{\pi_{t-1}}, \mathbb{P}^{\Phi}_{\pi_\theta})$. To optimize the Wasserstein distance we use Algorithm 1. Importantly, stochastic gradients of $F(\theta)$ can be approximated by samples from π_θ. In its simplest form, the gradient $\hat{\nabla}_\theta \hat{F}$ can be computed by the vanilla policy gradient over the advantage component and using the REINFORCE estimator through the components involving Test Functions acting on trajectories from $\mathbb{P}_{\pi_\theta}$. For **off-policy** embeddings, $\hat{\nabla}_\theta \hat{F}$ can be computed by sampling from the product of the state probing distributions. Gradients through the differentiable test functions can be computed by the chain rule: $\nabla_\theta \lambda(S, \pi_\theta(S)) = (\nabla_\phi \lambda(\phi))^\top \nabla_\theta \phi$ for $\phi = (S, \pi_\theta(S))$.

BGPG can be thought of as a variant of Trust Region Policy Optimization with a Wasserstein penalty. As opposed to vanilla TRPO, the optimization path of BGPG flows through policy parameter space while encouraging it to follow a smooth trajectory through the geometry of the behavioral manifold. We proceed to show that given the right embedding and cost function, we can prove a monotonic improvement theorem for BGPG, showing that our methods satisfy at least similar guarantees as TRPO.

Furthermore, Let $V(\pi)$ be the expected reward of policy π and $\rho_\pi(s) = \mathbb{E}_{\tau \sim \mathbb{P}_\pi}\left[\sum_{t=0}^{T} \mathbf{1}(s_t = s)\right]$ be the visitation measure.

Two distinct policies π and $\tilde{\pi}$ can be related via the equation (see: (Sutton et al., 1998)) $V(\tilde{\pi}) = V(\pi) + \int_{\mathcal{S}} \rho_{\tilde{\pi}}(s)\left(\int_{\mathcal{A}} \tilde{\pi}(a|s) A^{\pi}(s, a) da\right) ds$ and the linear approximations to V around π via: $L(\tilde{\pi}) = V(\pi) + \int_{\mathcal{S}} \rho_{\pi}(s)\left(\int_{\mathcal{A}} \tilde{\pi}(a|s) A^{\pi}(s, a) da\right) ds$ (see: (Kakade & Langford, 2002)). Let $\mathcal{S}$ be a finite set. Consider the following embedding $\Phi^s : \Gamma \to \mathbb{R}^{|\mathcal{S}|}$ defined by $\left(\Phi(\tau)\right)_s = \sum_{t=0}^{T} \mathbf{1}(s_t = s)$ and related cost function defined as: $C(\mathbf{v}, \mathbf{w}) = \|\mathbf{v} - \mathbf{w}\|_1$. Then $\mathrm{WD}_0(\mathbb{P}^{\Phi^s}_{\tilde{\pi}}, \mathbb{P}^{\Phi^s}_{\pi})$ is related to visitation frequencies since $\mathrm{WD}_0(\mathbb{P}^{\Phi^s}_{\tilde{\pi}}, \mathbb{P}^{\Phi^s}_{\pi}) \geq \sum_{s \in \mathcal{S}} |\rho_\pi(s) - \rho_{\tilde{\pi}}(s)|$. These observations enable us to prove an analogue of Theorem 1 from (Schulman et al., 2015) (see Section C.2 for the proof), namely:

Theorem 5.1. *If* $\mathrm{WD}_0(\mathbb{P}^{\Phi^s}_{\tilde{\pi}}, \mathbb{P}^{\Phi^s}_{\pi}) \leq \delta$ *and* $\epsilon = \max_{s,a} |A^{\pi}(s, a)|$, *then* $V(\tilde{\pi}) \geq L(\tilde{\theta}) - \delta\epsilon$.

As in (Schulman et al., 2015), Theorem 5.1 implies a policy improvement guarantee for BGPG.

5.5. Behavior Guided Evolution Strategies (BGES)

ES takes a black-box optimization approach to RL, by considering a rollout of a policy, parameterized by θ as a black-box function F. This approach has gained in popularity recently (Salimans et al., 2017; Mania et al., 2018; Choromanski et al., 2019). If we take this approach to optimizing the objective in Eq. 7, the result is a black-box optimization algorithm which seeks to maximize the reward and simultaneously maximizes or minimizes the difference in behavior from the base embedding distribution $\mathbb{P}^{\Phi}_{\mathrm{b}}$. We call it Behavior-Guided Evolution Strategies (BGES) algorithm (see: Alg. 4).

When $\beta > 0$, and we take $\mathbb{P}^{\Phi}_{\mathrm{b}} = \mathbb{P}^{\Phi}_{\pi_{t-1}}$, BGES resembles the NSR-ES algorithm from (Conti et al., 2018), an instantiation of *novelty search* (Lehman & Stanley, 2008). The positive weight on the WD-term enforces newly constructed policies to be behaviorally different from the previous ones while the $\mathcal{R}-$term drives the optimization to maximize the reward. The key difference in our approach is the probabilistic embedding map, with WD rather than Euclidean distance. We show in Section 7.2 that BGES outperforms NSR-ES for challenging exploration tasks.

Algorithm 4 Behavior-Guided Evolution Strategies

Input: learning rate η, noise standard deviation σ, iterations T, BEM Φ, β (> 0 for repulsion, < 0 for imitation).

Initialize: Initial policy π_0 parametrized by θ_0, Behavioral Test Functions λ_1, λ_2. Evaluate policy π_0 to return trajectory τ_0

for $t = 1, \ldots, T-1$ **do**

1. Sample $\epsilon_1, \cdots, \epsilon_n$ independently from $\mathcal{N}(0, I)$.
2. Evaluate policies $\{\pi_t^k\}_{k=1}^n$ parameterized by $\{\theta_t + \sigma\epsilon_k\}_{k=1}^n$, get rewards R_k and trajectories τ_k for all k.
3. Update λ_1 and λ_2 using Algorithm 1.
4. Approximate $\widehat{\mathrm{WD}}_\gamma(\mathbb{P}^\Phi_{\pi_t^k}, \mathbb{P}^\Phi_{\pi_t})$ plugging in λ_1, λ_2 into Eq. 5 for each perturbed policy π_k
5. Update Policy: $\theta_{t+1} = \theta_t + \eta \nabla_{ES} F$, where:

$$\nabla_{ES} F = \frac{1}{\sigma} \sum_{k=1}^{n} [(1-\beta)(R_k - R_t) + \beta \widehat{\mathrm{WD}}_\gamma(\mathbb{P}^\Phi_{\pi_t^k}, \mathbb{P}^\Phi_{\pi_t})]\epsilon_k$$

6. Related Work

Our work is related to research in multiple areas in neuroevolution and machine learning:

Behavior Characterizations: The idea of directly optimizing for behavioral diversity was introduced by (Lehman & Stanley, 2008) and (Lehman, 2012), who proposed to search directly for *novelty*, rather than simply assuming it would naturally arise in the process of optimizing an objective function. This approach has been applied to deep RL (Conti et al., 2018) and meta-learning (Gajewski et al., 2019). In all of this work, the policy is represented via a behavioral characterization (BC), which requires domain knowledge. In our setting, we move from deterministic BCs to stochastic behavioral embeddings, thus requiring the use of metrics capable of comparing probabilistic distributions.

Distance Metrics: WDs have been used in many applications in machine learning where guarantees based on distributional similarity are required (Jiang et al., 2019; Arjovsky et al., 2017). We make use of WDs in our setting for a variety of reasons. First and foremost, the dual formulation of the WD allows us to recover Behavioral Test Functions, providing us with behavior-driven trajectory scores. In contrast to KL divergences, WDs are sensitive to user-defined costs between pairs of samples instead of relying only on likelihood ratios. Furthermore, as opposed to KL divergences, it is possible to take SGD steps using entropy-regularized Wasserstein objectives. Computing an estimator of the KL divergence is hard without a density model. Since in our framework multiple unknown trajectories may map to the same behavioral embedding, the likelihood ratio between two embedding distributions may be ill-defined.

WDs for RL: We are not the first to propose using WDs in RL. (Zhang et al., 2018) have recently introduced Wasserstein Gradient Flows (WGFs), which casts policy optimization as gradient descent flow on the manifold of corresponding probability measures, where geodesic lengths are given as second-order WDs. We note that computing WGFs is a nontrivial task. In (Zhang et al., 2018) this is done via particle approximation methods, which we show in Section 7 is substantially slower than our methods. The WD has also been employed to replace KL terms in standard Trust Region Policy Optimization (Richemond & Maginnis, 2017). This is a very special case of our more generic framework (cf. Section 5.3). In (Richemond & Maginnis, 2017) it is suggested to solve the corresponding RL problems via Fokker-Planck equations and diffusion processes, yet no empirical evidence of the feasibility of this approach is provided. We propose general practical algorithms and provide extensive empirical evaluation.

Distributional RL Distributional RL (DRL, (Bellemare et al., 2017)) expands on traditional off-policy methods (Mnih et al., 2013) by attempting to learn a distribution of the return from a given state, rather than just the expected value. These approaches have impressive experimental results (Bellemare et al., 2017; Dabney et al., 2018), with a growing body of theory (Rowland et al., 2018; Qu et al., 2019; Bellemare et al., 2019; Rowland et al., 2019). Superficially it may seem that learning a distribution of returns is similar to our approach to PPEs, when the BEM is a distribution over rewards. Indeed, reward-driven embeddings used in DRL can be thought of as special cases of the general class of BEMs. We note two key differences: 1) DRL methods are off-policy whereas our BGES and BGPG algorithms are on-policy, and 2) DRL is typically designed for discrete domains, since Q-Learning with continuous action spaces is generally much harder. Furthermore, we note that while the WD is used in DRL, it is only for the convergence analysis of the DRL algorithm (Bellemare et al., 2017).

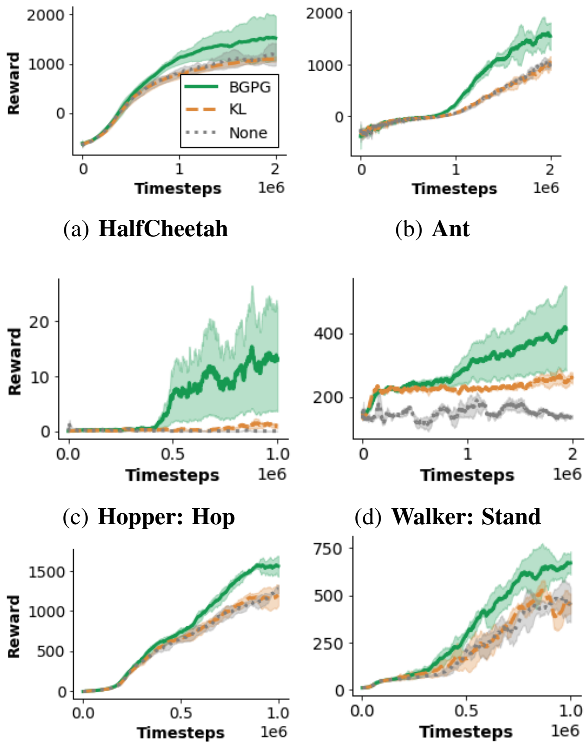
7. Experiments

Here we seek to test whether our approach to RL translates to performance gains by evaluating BGPG and BGES, versus their respective baselines for a range of tasks. For each subsection we provide additional details in the Appendix.

7.1. Behavior-Guided Policy Gradient

Our key question is whether our techniques lead to outperformance for BGPG vs. baseline TRPO methods using KL divergence, which are widely used in the reinforcement learning community. For the BEM, we use the concatenation-of-actions, as used already in TRPO. We consider a variety of challenging problems from the DeepMind Control Suite

(Tassa et al., 2018) and Roboschool (RS). In Fig. 4 we see that BGPG does indeed outperform KL-based TRPO methods, with gains across all six environments. We also confirm results from (Schulman et al., 2015) that a trust region typically improves performance.



(a) **HalfCheetah** (b) **Ant**

(c) **Hopper: Hop** (d) **Walker: Stand**

(e) **RS: HalfCheetah** (f) **RS: Walker2d**

Figure 4: **BGPG vs. TRPO:** We compare BGPG and TRPO (KL divergence) on several continuous control tasks. As a baseline we also include results without a trust region ($\beta = 0$ in Algorithm 3). Plots show the mean $\pm$ std across 5 random seeds.

Wall Clock Time: To illustrate computational benefits of alternating optimization of WD in BGPG, we compare it to the method introduced in (Zhang et al., 2018). In practice, the WD across different state samples can be optimized in a batched manner, details of which are in the Appendix. In Table 7.1 we see that BGPG is substantially faster.

	(Zhang et al., 2018)	BGPG
Pendulum	3720	777
Hopper: Stand	26908	10817
Hopper: Hop	23542	12820
Walker: Stand	13497	4082

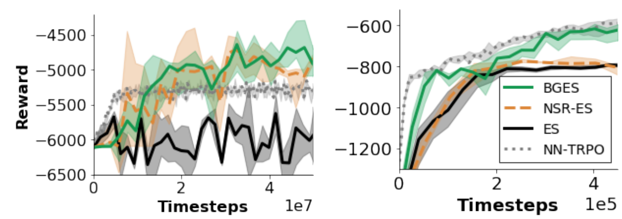
Table 1: Clock time (s) to achieve a normalized reward of 90% of the best achieved. All experiments were run on the same CPU.

7.2. Behavior-Guided Evolution Strategies

Next we seek to evaluate the ability for BGES to use its behavioral repulsion for exploration.

Deceptive Rewards A common challenge in RL is *deceptive* rewards. These arise since agents can only learn from data gathered via experience in the environment. To test BGES in this setting, we created two intentionally deceptive environments. In both cases the agent is penalized at each time step for its distance from a goal. The deception comes from a barrier, which means initially positive rewards from moving directly forward will lead to a suboptimal policy.

We consider two agents—a two-dimensional point and a larger $\mathrm{quadruped}$. Details are provided in the Appendix (Section B). We compare with state-of-the-art **on-policy** methods for exploration: $\mathrm{NSR\text{-}ES}$ (Conti et al., 2018), which assumes the BEM is deterministic and uses the Euclidean distance to compare policies, and $\mathrm{NoisyNet\text{-}TRPO}$ (Fortunato et al., 2018). We used the reward-to-go and final state BEMs for the $\mathrm{quadruped}$ and point respectively.



(a) **Quadruped** (b) **Point**

Figure 5: **Deceptive Rewards.** Plots show the mean $\pm$ std across 5 random seeds for two environments: $\mathrm{Quadruped}$ and Point.

Policies avoiding the wall correspond to rewards: $R > -5000$ and $R > -800$ for the $\mathrm{quadruped}$ and point respectively. In the prior case an agent needs to first learn how to walk and the presence of the wall is enough to prohibit vanilla ES from even learning forward locomotion. As we see in Fig. 5, BGES is the only method that drives the agent to the goal in *both* settings.

8. Conclusion and Future Work

In this paper we proposed a new paradigm for on-policy learning in RL, where policies are embedded into expressive latent behavioral spaces and the optimization is conducted by utilizing the repelling/attraction signals in the corresponding probabilistic distribution spaces. The use of Wasserstein distances (WDs) guarantees flexibility in choosing cost funtions between embedded policy trajectories, enables stochastic gradient steps through corresponding regularized objectives (as opposed to KL divergence methods) and provides an elegant method, via their dual formulations, to quantify behaviorial difference of policies through the behavioral test functions. Furthermore, the dual formulations give rise to efficient algorithms optimizing RL objectives regularized with WDs.

We also believe the presented methods shed new light on several other challenging problems of modern RL, including:

learning with safety guarantees (a repelling signal can be used to enforce behaviors away from dangerous ones) or anomaly detection for reinforcement learning agents (via the above score functions). Finally, we are interested in extending our method to the off policy setting.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 449–458. JMLR.org, 2017. URL http://dl.acm.org/citation.cfm?id=3305381.3305428.

Bellemare, M. G., Roux, N. L., Castro, P. S., and Moitra, S. Distributional reinforcement learning with linear function approximation. In Chaudhuri, K. and Sugiyama, M. (eds.), *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pp. 2203–2211. PMLR, 16–18 Apr 2019.

Bourbaki, N. *General Topology: Elements of Mathematics*. Addison-Wesley, 1966.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Choromanski, K., Pacchiano, A., Parker-Holder, J., Tang, Y., Jain, D., Yang, Y., Iscen, A., Hsu, J., and Sindhwani, V. Provably robust blackbox optimization for reinforcement learning. *CoRR*, abs/1903.02993, 2019.

Conti, E., Madhavan, V., Such, F. P., Lehman, J., Stanley, K. O., and Clune, J. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems 31, NeurIPS*, pp. 5032–5043, 2018.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. Implicit quantile networks for distributional reinforcement learning. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1096–1105, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. Openai baselines. https://github.com/openai/baselines, 2017.

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

Gajewski, A., Clune, J., Stanley, K. O., and Lehman, J. Evolvability es: Scalable and direct optimization of evolvability. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, pp. 107–115, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6111-8. doi: 10.1145/3321707.3321876.

Genevay, A., Cuturi, M., Peyré, G., and Bach, F. Stochastic optimization for large-scale optimal transport. In *Advances in neural information processing systems*, pp. 3440–3448, 2016.

Jiang, R., Pacchiano, A., Stepleton, T., Jiang, H., and Chiappa, S. Wasserstein fair classification. *arXiv preprint arXiv:1907.12059*, 2019.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. 2:267–274, 2002.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kullback, S. and Leibler, R. A. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03 1951. doi: 10.1214/aoms/1177729694. URL https://doi.org/10.1214/aoms/1177729694.

Lee, S. J. and Popovic, Z. Learning behavior styles with inverse reinforcement learning. *ACM Trans. Graph.*, 29(4):122:1–122:7, 2010. doi: 10.1145/1778765.1778859. URL https://doi.org/10.1145/1778765.1778859.

Lehman, J. *Evolution through the Search for Novelty*. PhD thesis, University of Central Florida, 2012.

Lehman, J. and Stanley, K. O. Exploiting open-endedness to solve problems through the search for novelty. In *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI*. MIT Press, 2008.

Mania, H., Guy, A., and Recht, B. Simple random search of static linear policies is competitive for reinforcement learning. *Advances in Neural Information Processing Systems 31, NeurIPS*, 2018.

Meyerson, E., Lehman, J., and Miikkulainen, R. Learning behavior characterizations for novelty search. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016*, pp. 149–156, 2016. doi: 10.1145/2908812.2908929. URL https://doi.org/10.1145/2908812.2908929.

Micchelli, C. A., Xu, Y., and Zhang, H. Universal kernels. *Journal of Machine Learning Research*, 7:2651–2667, 2006. URL http://jmlr.org/papers/v7/micchelli06a.html.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. Playing atari with deep reinforcement learning. *ArXiv*, abs/1312.5602, 2013.

Pugh, J. K., Soros, L. B., and Stanley, K. O. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 2016.

Qu, C., Mannor, S., and Xu, H. Nonlinear distributional gradient temporal-difference learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5251–5260, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T. (eds.), *Advances in Neural Information Processing Systems 20*, pp. 1177–1184. Curran Associates, Inc., 2008.

Richemond, P. H. and Maginnis, B. On wasserstein reinforcement learning and the fokker-planck equation. *CoRR*, abs/1712.07185, 2017. URL http://arxiv.org/abs/1712.07185.

Rowland, M., Bellemare, M., Dabney, W., Munos, R., and Teh, Y. W. An analysis of categorical distributional reinforcement learning. In Storkey, A. and Perez-Cruz, F. (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 29–37, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR.

Rowland, M., Dadashi, R., Kumar, S., Munos, R., Bellemare, M. G., and Dabney, W. Statistics and samples in distributional reinforcement learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5528–5536, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR, abs/1703.03864*, 2017.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 1889–1897, 2015. URL http://jmlr.org/proceedings/papers/v37/schulman15.html.

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

Sutton, R. S., Barto, A. G., et al. *Introduction to Reinforcement Learning*, volume 135. MIT press Cambridge, 1998.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Villani, C. Optimal transport: Old and new. *Springer*, 2008.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Zhang, R., Chen, C., Li, C., and Carin, L. Policy optimization as wasserstein gradient flows. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 5741–5750, 2018. URL http://proceedings.mlr.press/v80/zhang18a.html.