Effective Diversity in Population-Based Reinforcement Learning

Jack Parker-Holder [*1] **Aldo Pacchiano** [*2] **Krzysztof Choromanski** [3] **Stephen Roberts** [1]

Abstract

Exploration is a key problem in reinforcement learning, since agents can only learn from data they acquire in the environment. With that in mind, maintaining a population of agents is an attractive method, as it allows data be collected with a diverse set of behaviors. This behavioral diversity is often boosted via multi-objective loss functions. However, those approaches typically leverage mean field updates based on pairwise distances, which makes them susceptible to cycling behaviors and increased redundancy. In addition, explicitly boosting diversity often has a detrimental impact on optimizing already fruitful behaviors for rewards. As such, the reward-diversity trade off typically relies on heuristics. Finally, such methods require behavioral representations, often handcrafted and domain specific. In this paper, we introduce an approach to optimize all members of a population simultaneously. Rather than using pairwise distance, we measure the volume of the entire population in a behavioral manifold, defined by task-agnostic behavioral embeddings. In addition, our algorithm Diversity via Determinants (DvD), adapts the degree of diversity during training using online learning techniques. We introduce both evolutionary and gradient-based instantiations of DvD and show they effectively improve exploration without reducing performance when better exploration is not required.

1. Introduction

Reinforcement Learning (RL) considers the problem of an agent taking actions in an environment to maximize total (discounted/expected) reward (Sutton and Barto, 1998). An agent can typically only learn from experience acquired in the environment, making exploration crucial to learn high performing behaviors.

Training a *population* of agents provides is a promising approach to gathering a diverse range of experiences, often with the same (wall-clock) training time (Recht et al., 2011; Li et al., 2019). Population-based methods are particularly prominent in the Neuroevolution community (Stanley et al., 2019), but have recently been of increased interest in RL (Jung et al., 2020; Liu et al., 2019; Khadka and Tumer, 2018; Jung et al., 2020; Khadka et al., 2019; Pourchot and Sigaud, 2019). One particularly exciting class of neuroevolution methods is Quality Diversity (QD, (Pugh et al., 2016)) where algorithms explicitly seek high performing, yet diverse behaviors (Cully et al., 2015). However, these methods have a few key shortcomings.

Typically, each agent is optimized with a mean field assumption, only considering its individual contribution to the population's joint reward-diversity objective. Consequently, cycles may arise, whereby different members of the population constantly switch between behaviors. This may prevent any single agent exploiting a promising behavior. This phenomenon motivates the MAP-Elites algorithm (Mouret and Clune, 2015; Cully et al., 2015), whereby only one solution may lie in each quadrant of a pre-defined space.

This pre-defined space is a common issue with QD (Kistemaker and Whiteson, 2011). Behavioral characterizations (BCs) often have to be provided to the agent, for example, in locomotion it is common to use the final (x, y) coordinates. As such, automating the discovery of BCs is an active research area (Gaier et al., 2020). In RL, gradients are usually taken with respect to the actions taken by an agent for a set of sampled states. This provides a natural way to embed a policy behavior (Jung et al., 2020; Hong et al., 2018; Doan et al., 2019). Incidentally, the geometry induced by such embeddings has also been used in popular trust region algorithms (Schulman et al., 2018; 2015).

In this paper we formalize this approach and define *behavioral embeddings* as the actions taken by policies. We measure the diversity of the entire population as the volume of the inter-agent kernel (or similarity) matrix, which we show has theoretical benefits compared to pairwise distances. In our approach agents are still optimizing for their *local* rewards and this signal is a part of their hybrid objective that

[*]Equal contribution [1]Department of Engineering Science, University of Oxford [2]Depeartment of Engineering and Computer Science, UC Berkeley [3]Google Brain Robotics. Correspondence to: Jack Parker-Holder <jackph@robots.ox.ac.uk>.

4th Lifelong Learning Workshop at ICML 2020, Copyright 2020 by the author(s).

also takes into account the *global* goal of the population - its diversity.

However, we note that it is remains a challenge to set the diversity-reward trade off. If misspecified, we may see fruitful behaviors disregarded. Existing methods either rely on a (potentially brittle) hyperparameter, or introduce an annealing-based heuristic (Conti et al., 2018). To ensure diversity is promoted *effectively*, our approach is to adapt the diversity-reward objective using Thompson sampling (Russo et al., 2017; Russo and Van Roy, 2016). This provides us with a principled means to trade-off reward vs. diversity through the lens of multi-armed bandits (Slivkins, 2019). Combining these insights, we introduce Diversity via Determinants (DvD).

DvD is a general approach, and we introduce two implementations: one building upon Evolution Strategies, DvD-ES, which is able to discover diverse solutions in multi-modal environments, and an off-policy RL algorithm, DvD-TD3, which leads to greater performance on the challenging $\mathrm{Humanoid}$ task. Crucially, we show DvD still performs well when diversity is not required.

This paper is organized as follows. **(1)** We begin by introducing main concepts in Sec. 2 and providing strong motivation for our DvD algorithm in Sec. 3. **(2)** We then present our algorithm in Sec. 4. **(3)** In Sec. 5 we discuss related work in more detail. **(4)** In Sec. 6 we provide empirical evidence of the effectiveness of DvD. **(5)** We conclude and explain broader impact in Sec. **??** and provide additional technical details in the Appendix (including ablation studies and proofs).

2. Preliminaries

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 $s_t, s_{t+1} \in \mathcal{S}$ and $a_t \in \mathcal{A}$: $\mathrm{P}(s_{t+1}|s_t, a_t)$ is the probability that the system/agent transitions from s_t to s_{t+1} given action a_t and $r_t = r(s_t, a_t, s_{t+1})$ is a reward obtained by an agent transitioning from s_t to s_{t+1} via a_t.

A policy $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$ is a (possibly randomized) mapping (parameterized by $\theta \in \mathbb{R}^d$) from $\mathcal{S}$ to $\mathcal{A}$. Policies π_θ are typically represented by neural networks, encoded by parameter vectors $\theta \in \mathbb{R}^d$ (their flattened representations). Model free RL methods are either on-policy, focusing on gradients of the policy from samples (Schulman et al., 2015; 2018), or off-policy, targeting learning a value function (Mnih et al., 2013; Fujimoto et al., 2018; Haarnoja et al., 2018).

In on-policy RL, the goal is to optimize parameters θ of π_θ such that an agent deploying policy π_θ in the environment given by a fixed MDP maximizes $R(\tau) = \sum_{t=1}^{T} r_t$, the total (expected/discounted) reward over a rollout time-step horizon T (assumed to be finite). The typical objective is as follows:

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] \qquad (1)$$

where $P(\tau|\theta) = \rho(s_0) \prod_{t=1}^{T} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$, for initial state probability $\rho(s_0)$ and transition dynamics $P(s_{t+1}|s_t, a_t)$, which is often deterministic. Policy gradient (PG) methods (Schulman et al., 2018; 2015), consider objectives of the following form:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau)\right], \qquad (2)$$

which can be approximated with samples (in the action space) by sampling $a_t \sim \pi_\theta(a_t|s_t)$.

An alternative formulation is Evolution Strategies (ES, (Salimans et al., 2017)), which has recently shown to be effective (Such et al., 2017; Choromanski et al., 2018; Mania et al., 2018). ES methods optimize Equation 1 by considering $J(\pi_\theta)$ to be a blackbox function $F : \mathbb{R}^d \rightarrow \mathbb{R}$ as taking as input parameters $\theta \in \mathbb{R}^d$ of a policy π_θ and outputting $R(\tau)$. One benefit of this approach is potentially achieving deep exploration (Plappert et al., 2018; Fortunato et al., 2018).

A key benefit of ES methods is they naturally allow us to maintain a population of solutions, which has been used to boost diversity. Novelty search methods (Conti et al., 2018; Lehman and Stanley, 2011) go further and explicitly augment the loss function with an additional term, as follows:

$$J(\pi_\theta) = R(\tau_\theta) + \lambda d(\tau_\theta) \qquad (3)$$

where $\lambda > 0$ is the reward-diversity trade-off. We assume the policy and environment are deterministic, so τ_θ is the only possible trajectory, and $d(\pi_{\theta^i}) = \frac{1}{M} \sum_{j \in M, j \neq i} ||BC(\pi^i_\theta) - BC(\pi^j_\theta)||_2$ for some l-dimensional behavioral mapping $BC : \tau \rightarrow \mathbb{R}^l$.

This formulation has been shown to boost exploration, and as such, there have been a variety of attempts to incorporate novelty-based metrics into RL algorithms (Jung et al., 2020; Doan et al., 2019; Hong et al., 2018). Ideally, we would guide optimization in a way which would evenly distribute policies in areas of the embedding space, which correspond to high rewards. However, the policy embeddings (BCs) used are often based on heuristics which may not generalize to new environments (Kistemaker and Whiteson, 2011). In addition, the single sequential updates may lead high performing policies away from an improved reward (as is the case in the $\mathrm{Humanoid}$ experiment in (Conti et al., 2018)). Finally, cycles may become present, whereby the population moves from one area of the feature space to a new area and back again (Mouret and Clune, 2015).

Below we address these issues, and introduce a new objective which updates all agents simultaneously.

3. Diversity via Determinants

Here we introduce our task agnostic embeddings and formalize our approach to optimize for population-wide diversity.

3.1. Task Agnostic Behavioral Embeddings

Many popular policy gradient algorithms (Schulman et al., 2015; 2018) consider a setting whereby a new policy $\pi_{\theta_{t+1}}$ is optimized with a constraint, typically of the following form:

$$\mathbb{E}_{s\sim\pi_{\theta_t}}[\mathrm{KL}[\pi_{\theta_t}(\cdot|s), \pi_{\theta_{t+1}}(\cdot|s)]] \leq \delta \tag{4}$$

for $\delta \geq 0$ and where KL represents the KL-divergence. This requirement can be considered as a constraint on the behavior of the new policy (Pacchiano et al., 2019). Despite the remarkable success of these algorithms, there has been little consideration for action-based behavior embeddings for novelty search methods.

Inspired by this approach, we choose to represent policies as follows:

Definition 3.1. *Let θ^i be a vector of neural network parameters encoding a policy π_{θ^i} and let $\mathcal{S}$ be a finite set of states. The **Behavioral Embedding** of θ^i is defined as: $\phi(\theta^i) = \{\pi_{\theta^i}(.|s)\}_{s\in\mathcal{S}}$.*

This approach allows us to represent the behavior of policies in vectorized form, as $\phi : \theta \to \mathbb{R}^l$, where $l = |a| \times N$, where $|a|$ is the dimensionality of each action $a \in \mathcal{A}$ and N is the total number of states. When N is the number of states in a finite MDP, the policies are determimistic and the embedding is the as in Definition 3.1, we have:

$$\phi(\theta^i) = \phi(\theta^j) \iff \pi_{\theta^i} = \pi_{\theta^j}, \tag{5}$$

where θ^i, θ^j are vectorized parameters. In other words, the policies are the same since they always take the same action in every state of the finite MDP. Note that this does not imply that $\theta^i = \theta^j$.

3.2. Joint Population Update

Equipped with this notion of a behavioral embedding, we can now consider a means to compare two policies. Consider a smooth kernel k, such that $k(x_1, x_2) \leq 1$, for $x_1, x_2 \in \mathbb{R}^d$. A popular choice of kernel is the squared exponential (SE), defined as follows:

$$k_{\mathrm{SE}}(x_1, x_2) = \exp\left(-\frac{||x_1 - x_2||^2}{2l^2}\right), \tag{6}$$

for some length-scale $l > 0$. Now moving back to policy embeddings, we extend our previous analysis to the kernel or similarity between two embeddings, as follows:

$$k(\phi(\theta^i), \phi(\theta^j)) = 1 \iff \pi_{\theta^i} = \pi_{\theta^j} \tag{7}$$

We consider two policies to be behaviorally orthogonal if $k(\phi(\theta^i), \phi(\theta^j)) = 0$. With a flexible way of measuring inter-policy similarity at hand, we can define the *population-wide* diversity as follows:

Definition 3.2. ***(Population Diversity)*** *Consider a finite set of M policies, parameterized by $\Theta = \{\theta^1, ..., \theta^M\}$, with $\theta^i \in \mathbb{R}^d$. We denote $\mathrm{Div}(\Theta) \overset{\mathrm{def}}{=} \det(K(\phi(\theta^i_t), \phi(\theta^j_t))^M_{i,j=1}) = \det(\mathbf{K})$, where $K : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}$ is a given kernel function. Matrix $\mathbf{K}$ is positive semidefinite since all principal minors of $\det(\mathbf{K})$ are non-negative.*

This formulation is heavily inspired by Determinantal Point Processes (DPPs, (Kulesza and Taskar, 2012)), a mechanism which produces diverse subsets by sampling proportionally to the determinant of the kernel matrix of points within the subset. From a geometric perspective, the determinant of the kernel matrix represents the volume of a parallelepiped spanned by feature maps corresponding to the kernel choice. We seek to maximize this volume, effectively "filling" the feature (or behavior) space.

Now consider the DvD loss function, as follows:

$$J(\Theta_t) = \underbrace{\sum_{i=1}^{M} \mathbb{E}_{\tau\sim\pi_{\theta^i}}[R(\tau)]}_{\text{individual rewards}} + \underbrace{\lambda_t \mathrm{Div}(\Theta_t)}_{\text{population diversity}} \tag{8}$$

where $\lambda_t \in (0, 1)$ is the trade-off between reward and diversity. This fundamentally differs from Equation 3, since we directly optimize for Θ_t rather than separately considering $\{\theta^i\}^M_{i=1}$.

Theorem 3.3. *Let M be a finite, tabular MDP with $\tilde{M} \geq M$ distinct optimal policies $\{\pi_i\}^{\tilde{M}}_{i=1}$ all achieving a cumulative reward of $\mathcal{R}$ and such that the reward value $\mathcal{R}(\pi)$ of any suboptimal policy π satisfies $\mathcal{R}(\pi) + \Delta < \mathcal{R}$ for some $\Delta > 0$. There exists $\lambda_t > 0$, such that the objective in Equation 8 can only be maximized if the population contains M distinct optimal solutions.*

The proof of Theorem 3.3 is in the Appendix (Sec. 9), where we also show that for the case of the squared exponential kernel, the first order approximation to the determinant is in fact related to the mean pairwise L2-distance. However, for greater population sizes, this first order approximation is zero, implying the determinant comes from higher order terms.

We have now shown theoretically that our formulation for diversity can recover diverse high performing solutions. This result shows the importance of utilizing the determinant to measure diversity rather than the pairwise distance.

Using the determinant to quantify diversity prevents the undesirable clustering phenomenon, where a population

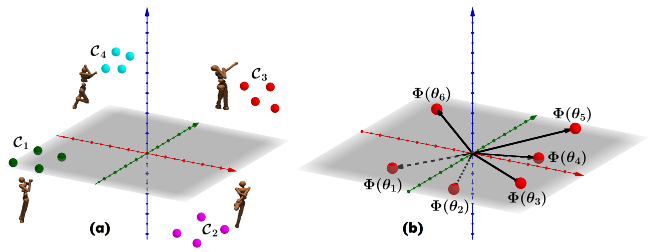


Figure 1. Determinant vs. pairwise distance. (a): populations of agents split into four clusters with agents within cluster discovering similar policies. (b): embedded policies $\phi(\theta_1), ..., \phi(\theta_6)$ lie in a grey hyperplane. In (a) resources within a cluster are wasted since agents discover very similar policies. In (b) all six embeddings can be described as linear combinations of embeddings of fewer canonical policies. In both settings the mean pairwise distance will be high but diversity as measured by determinants is low.

evolves to a collection of conjugate classes. To illustrate this point, consider a simple scenario, where all M agents are partitioned into k clusters of size $\frac{M}{k}$ each for $k = o(M)$. By increasing the distance between the clusters one can easily make the novelty measured as an average distance between agents' embedded policies as large as desired, but that is not true for the corresponding determinants which will be zero if the similarity matrix is low rank. Furthermore, even if all pairwise distances are large, the determinants can be still close to zero if spaces where agents' high-dimensional policy embeddings live can be accurately approximated by much lower dimensional ones. Standard methods are too crude to measure novelty in such a way (see: Fig. 1).

Next we provide a simple concrete example to demonstrate this phenomenon.

3.3. An Illustrative Example: Tabular MDP

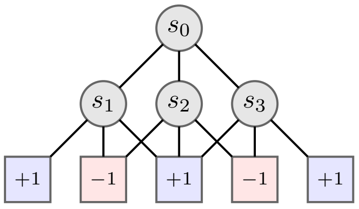


Figure 2. A simple MDP.

Consider the simple MDP in Fig. 2. There are four states $\mathcal{S} = \{s_1, s_2, s_3, s_4\}$, each has three actions, $\mathcal{A} = \{\text{-}1, 0, 1\}$ corresponding to left, down and right respectively. In addition, there are five terminal states, three of which achieve the maximum reward $(+1)$. Let $\phi^* = \{\phi(\theta^i)\}_{i=1}^5 = \{[\text{-}1, \text{-}1], [\text{-}1, 1], [0, 0], [1, \text{-}1], [1, 1]\}$, be the set of optimal policies. If we have a population of five agents, each achieving a positive reward, the determinant of the 5×5 kernel matrix is only > 0 if the population of agents is exactly ϕ^*. This may seem trivial, but note the same is not true for the pairwise distance. If we let $d(\Theta) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j>i}||\phi(\theta^i) - \phi(\theta^j)||_2$, then ϕ^* does not maximize d. One such example which achieves a higher value would be $\phi' = \{[\text{-}1, \text{-}1], [\text{-}1, 1], [1, \text{-}1], [1, 1], [1, 1]\}$. See the following link for a colab demonstration of this example: https://bit.ly/2XAlirX.

4. DvD Algorithm

4.1. Approximate Embeddings

In most practical settings the state space is intractably or infinitely large. Therefore, we must sample the states $\{s_i\}_{i=1}^n$, where $n < N$, and compute the embedding as an expectation as follows:

$$\phi(\theta_i) = \mathbb{E}_{s \sim \mathcal{S}}[\{\pi_{\theta^i}(.|s)\}] \tag{9}$$

In our experiments we choose to randomly sample the states s, which corresponds to frequency weights. Other possibilities include selecting diverse ensembles of states via DPP-driven sampling or using probabilistic models to select less frequently visited states. We explore each of these options in the experiments where we show the representative power of this action-based embedding is not overly sensitive to these design choices (see: Fig. 7). However, it remains a potential degree of freedom for future improvement in our method, potentially allowing a far smaller number of states to be used.

4.2. Adaptive Exploration

Optimizing Equation 8 relies on a user-specified degree of priority for each of the two objectives (λ_t). We formalize this problem through the lens of multi-armed bandits, and adaptively select λ_t such that we encourage favoring the reward or diversity at different stages of optimization.

Specifically, we use Thompson Sampling (Thompson, 1933; Srinivas et al., 2010; Agrawal and Goyal, 2012; Russo et al., 2017; Russo and Van Roy, 2016). Let $\mathcal{K} = \{1, \cdots, K\}$ denote a set of arms available to the decision maker (learner) who is interested in maximizing its expected cumulative reward. The optimal strategy for the learner is to pull the arm with the largest mean reward. At the beginning of each round the learner produces a sample mean from its mean reward model for each arm, and pulls the arm from which it obtained the largest sample. After observing the selected arm's reward, it updates its mean reward model.

Let π_t^i be the learner's reward model for arm i at time t. When $t = 0$ the learner initializes each of its mean reward models to prior distributions $\{\pi_0^i\}_{i=1}^K$. At any other time $t > 0$, the learner starts by sampling mean reward candidates $\mu_i \sim \pi_{t-1}^i$ and pulling the arm: $i_t = \arg\max_{i \in \mathcal{K}} \mu_i$. After observing a true reward sample r_t from arm i_t, the learner updates its posterior distribution for arm i_t. All the posterior distributions over arms $i \neq i_t$ remain unchanged.

In this paper we make use of a Bernoulli model for the reward signal corresponding to the two arms ($\lambda = 0, \lambda = 0.5$). At any time t, the chosen arm's sample reward is the indicator variable $r_t = \mathbf{1}\left(R_{t+1} > R_t\right)$ where R_t denotes the reward observed at θ_t and R_{t+1} that at θ_{t+1}. We make a simplifying stationarity assumption and disregard the changing nature of the arms' means in the course of optimization. We use beta distributions to model both the priors and the posteriors of the arms' means. For a more detailed description of the specifics of our methodology please see the Appendix (Sec. 10.1).

We believe this adaptive mechanism could also be used for count-based exploration methods or intrinsic rewards (Schmidhuber, 2010), and note very recent work using a similar approach to vary exploration in off-policy methods (Schaul et al., 2019; Badia et al., 2020) and model-based RL (Ball et al., 2020). Combining these insights, we obtain the DvD algorithm. Next we describe two practical implementations of DvD.

4.3. DvD-ES Algorithm

At each timestep, the set of policies $\Theta_t = \{\theta_t^i\}_{i=1}^M$ are simultaneously perturbed, with rewards computed locally and diversity computed globally. These two objectives are combined to produce a blackbox function with respect to Θ_t. At every iteration we sample Mk Gaussian perturbation vectors $\{\mathbf{g}_i^m\}_{i=1,\ldots,k}^{m=1,\ldots,M}$. We use two partitionings of this Mk-element subset that illustrate our dual objective - high local rewards and large global diversity. The first partitioning assigns to m^{th} worker a set $\{\mathbf{g}_1^m, ..., \mathbf{g}_k^m\}$. These are the perturbations used by the worker to compute its local rewards. The second partitioning splits $\{\mathbf{g}_i^m\}_{i=1,\ldots,k}^{m=1,\ldots,M}$ into subsets: $\mathcal{D}_i = \{\mathbf{g}_i^1, ..., \mathbf{g}_i^M\}$. Instead of measuring the contribution of an individual $\mathbf{g}_i^m$ to the diversity, we measure the contribution of the entire $\mathcal{D}_i$. This motivates the following definition of diversity:

$$\mathrm{Div}_t(i) = \mathrm{Div}_t(\theta_t^1 + \mathbf{g}_i^1, ..., \theta_t^M + \mathbf{g}_i^M). \qquad (10)$$

Thus, the DvD-ES gradient update is the following:

$$\theta_{t+1}^m = \theta_t^m + \frac{\eta}{k\sigma}\sum_{i=1}^{k}[(1-\lambda_t)R_i^m + \lambda_t \mathrm{Div}_t(i)]\mathbf{g}_i^m. \quad (11)$$

where $\sigma > 0$ is the smoothing parameter (Nesterov and Spokoiny, 2017; Salimans et al., 2017), k is the number of ES-sensings, η is the learning rate, and the embeddings are computed by sampling states from the most recent iteration.

4.4. DvD-TD3 Algorithm

It is also possible to compute analytic gradients for the diversity term in Equation 8. This means we can update policies with respect to the joint diversity using automatic differentiation.

Lemma 4.1. *The gradient of* $\log\left(\det(\mathbf{K})\right)$ *with respect to* $\mathbf{\Theta} = \theta^1, \cdots, \theta^M$ *equals:* $\nabla_\theta \log\left(\det(\mathbf{K})\right) = -\left(\nabla_\theta \Phi(\theta)\right)\left(\nabla_\Phi \mathbf{K}\right)\mathbf{K}^{-1}$, *where* $\phi(\theta) = \phi(\theta^1)\cdots\phi(\theta^M)$.

The proof of this lemma is in the Appendix, Sec. 9.2.

Inspired by (Jung et al., 2020), we introduce DvD-TD3, using multiple policies to collect data for a shared replay buffer. This is done by dividing the total data collection by the number of policies. When optimizing the policies, we use an augmented loss function, as in Equation 8, and make use of the samples in the existing batch for the embedding.

5. Related Work

Neuroevolution methods (Stanley et al., 2019), seek to maximize the reward of a policy through approaches strongly motivated by natural biological processes. They typically work by perturbing a policy, and either computing a gradient (as in Evolution Strategies) or selecting the top performing perturbations (as in Genetic Algorithms). The simplicity and scalability of these methods have led to their increased popularity in solving RL tasks (Salimans et al., 2017; Choromanski et al., 2018; 2019; Conti et al., 2018; Such et al., 2017).

Neuroevolution methods often make use of behavioral representations (Conti et al., 2018; Gajewski et al., 2019). In (Conti et al., 2018) it is proposed to use of a population of agents, each of which would seek to jointly maximize the reward and difference/*novelty* in comparison to other policies, quantified as the mean pairwise distance. Another approach, Evolvabilty ES (Gajewski et al., 2019) seeks to learn policies which can quickly adapt to a new task, by maximizing the variance or entropy of behaviors generated by a small perturbation. The MAP-Elites (Mouret and Clune, 2015) algorithm is conceptually similar to ours, the authors seek to find quality solutions in differing dimensions. However, these dimensions need to be pre-specified, whereas our method can be considered a learned version of this approach. To the best of our knowledge, only one recent neuroevolutionary approach (Jackson and Daley, 2019) uses the actions of a policy to represent behaviors, albeit in a genetic context over discrete actions.

There has recently been interest in unsupervised learning of diverse behaviors (Eysenbach et al., 2019; Hartikainen et al., 2020). These methods are similar in principle to novelty search without a reward signal, but instead focus on diversity in behaviors defined by the states they visit. Another class of algorithms making use of behavioral representations (Gupta et al., 2018) focuses on meta learning in the behavioral space, however they require pre-training on similar tasks in order to learn a new one. A meta learning approach from (Rusu et al., 2019) proposes using a latent generative representation of model parameters, which could be thought of as a behavioral embedding of the policy. Finally, (Wang and Liu, 2019) propose a functional approach for learning diversified parameters. As an iterative method, it is still subject to the undesirable cycling phenomenon.

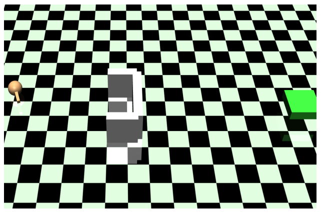
6. Experiments

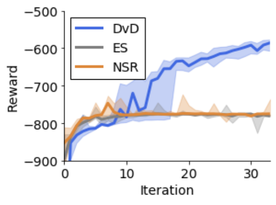
Here evaluate DvD-ES and DvD-TD3 in a variety of challenging settings.

6.1. Finding Diverse Solutions with DvD-ES

We compare DvD-ES against vanilla ES (referred to as ES), as well as NSR-ES from (Conti et al., 2018), both of which updates each population member sequentially. For both DvD and NSR-ES, we use the same embedding, with 20 randomly selected states. We use this for all **all** DvD-ES experiments.

We parameterize our policies with two hidden layer neural networks, with tanh activations (more details are in the Appendix, Section 8.2). All x-axes are presented in terms of iterations. Comparisons are made fair by dividing the number of iterations for two sequential algorithms (ES and NSR-ES) by the population size. All experiments made use of the ray (Moritz et al., 2017) library for parallel computing, with experiments run on a 32-core machine.


(a) point **environment**

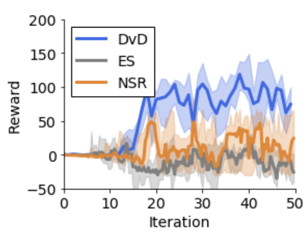


(b) **Median curves**

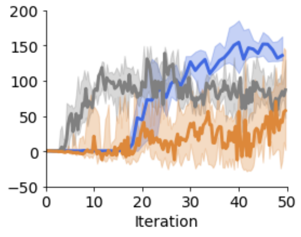
Figure 3. a) point environment b) median best performing agent across ten seeds. IQR shaded.

Exploration We begin with a simple environment, whereby a two dimensional point agent is given a reward equal to the negative distance away from a goal. The agent is separated from its goal by the wall (see: Fig. 3a)). We ran ten seeds, with hyperparameters described in detail in the Appendix (Table 4) and with population of size $M = 5$. As we see, both vanilla ES and NSR-ES fail to get past the wall (a reward of -800), yet DvD is able to solve the environment for **all 10 seeds**.

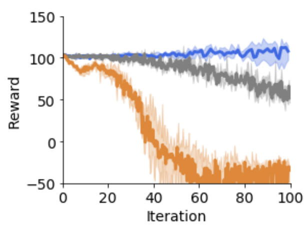
Multi-Modal Environments A key attribute of a QD algorithm is the ability to learn a diverse set of high performing solutions. This is often demonstrated qualitatively, through videos of learned gaits, and thus hard to scientifically prove. To test this we create environments where an agent can be rewarded for multiple different behaviors, as is typically done in multi-objective RL (Yang et al., 2019). Concretely, our environments are based on the Cheetah and Ant, where we assign rewards for *both* Forward and Backward tasks, which commonly used in meta-RL (Finn et al., 2017; Rothfuss et al., 2019). We can then evaluate the population of agents on both individual tasks, to quantitatively evaluate the diversity of the solutions.



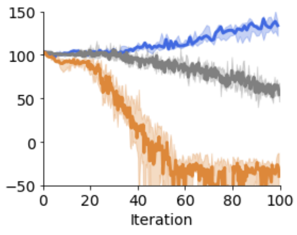
(a) **Cheetah: Forward**



(b) **Cheetah: Backward**



(c) **Ant: Forward**



(d) **Ant: Backward**

Figure 4. The median best performing agent across ten seeds for multi-modal tasks and two environments: Cheetah and Ant. The plots show median curves with IQR shaded.

In both settings we used a population size of $M = 3$, which is sufficient to learn both tasks. In Fig. 4 we see that DvD is able to learn both modes in both environments. For the Cheetah, the Backward task appears simpler to learn, and with no diversity term vanilla ES learns this task quickly, but subsequently performs poorly in the Forward task (with inverse reward function). For Ant, the noise from two separate tasks makes it impossible for vanilla ES to learn at all.

Single Mode Environments Now we consider the problem of optimizing tasks which have only one optimal solution or at least, with a smaller distance between optimal solutions (in the behavioral space), and an informative reward function. In this case, overly promoting diversity may lead to worse performance on the task at hand, as seen in NSR-ES ((Conti et al., 2018), Fig. 1.(c)). We test this using four widely studied continuous control tasks from OpenAI Gym. In all cases we use a population size of $M = 5$, we provide additional experimental details in the Appendix (see Section 8.2).

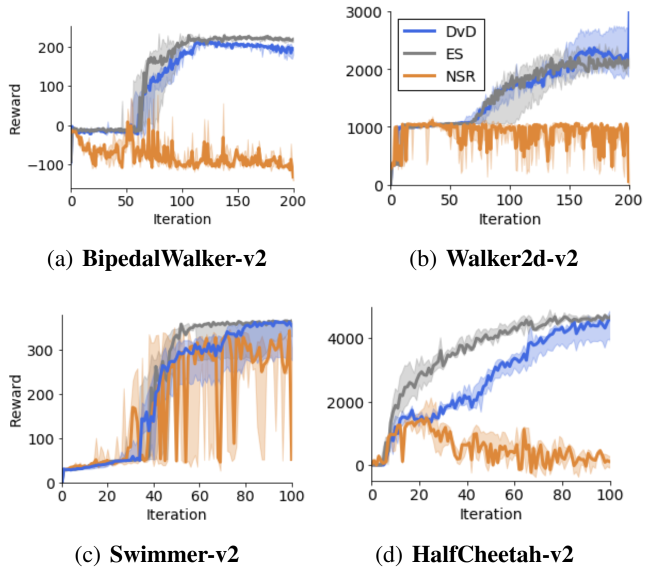


Figure 5. The median best performing agent across five seeds for four different environments.

As we see in Fig. 5, in all four environments NSR-ES fails to train (which is consistent with the vanilla Humanoid experiment from (Conti et al., 2018)) in contrast to DvD that shows only a minimal drop in performance vs. the vanilla ES method which is solely focusing on the reward function. This promising result implies that we gain the benefit of diversity without compromising on tasks, where it is not required. We note that DvD outperforms ES in Walker2d, which is known to have a deceptive local optimum at 1000 induced by the survival bonus (Mania et al., 2018).

These experiments enable us also to demonstrate the cyclic behaviour that standard novelty search approaches suffer from (see: Section 1). NSR-ES often initially performs well, before subsequently abandoning successful behaviors in the pursuit of novelty.

6.2. Teaching a Humanoid to Run with DvD-TD3

We now evaluate DvD-TD3 on the challenging Humanoid environment from the Open AI Gym (Brockman et al., 2016). The Humanoid-v2 task requires a significant degree of exploration, as there is a well-known local optimum at 5000, since the agent is provided with a survival bonus of 5 per timestep (Mania et al., 2018).

We train DvD-TD3 with $M = 5$ agents, where each agent has its own neural network policy, but a shared Q-function. We benchmark against both a single agent ($M = 1$), which is vanilla TD3, and then what we call ensemble TD3 (E-TD3) where $M = 5$ but there is no diversity term. We initialize all methods with $25,000$ random timesteps, where we set $\lambda_t = 0$ for DvD-TD3. We train each for a total of one million timesteps, and repeat for 7 seeds. The results are shown in Fig. 6.

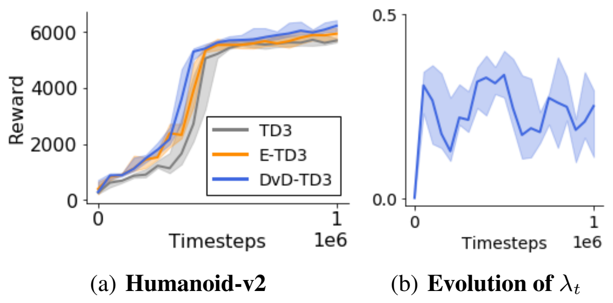


Figure 6. a) Median curves from 7 seeds, IQR shaded. b) The evolution of λ_t during training.

As we see, DvD-TD3 achieves better sample efficiency, as well as stronger maximum performance. For comparison, the median best agents for each algorithm were: DvD-TD3: 6091, E-TD3: 5654 and TD3: 5727. This provides t-statistics of 2.35 and 2.29 for DvD-TD3 vs. E-TD3 and TD3, using Welch's unequal variances t-test. Both of these are statistically significant ($p < 0.05$). As far as we are aware, DvD-TD3 is the first algorithm to get over 6000 on Humanoid-v2 in one million timesteps. This means the agent has achieved forward locomotion, rather than simply standing still (at 5000). Previous results for Soft Actor Critic (SAC) have only reached 6000 at three million timesteps (Haarnoja et al., 2018; Ciosek et al., 2019).

We note that while we maintain full policy networks for each member of the population, it is likely possible to implement the population with multiple heads (as in (Osband et al., 2016)). Space was not an issue in our case, but may be more important when using DvD for larger tasks (e.g. training from pixels).

7. Conclusion

In this paper we introduced DvD, a novel method for promoting diversity in population-based methods for reinforcement learning. DvD addresses the issue of cycling by utilizing a joint population update via determinants corresponding to ensembles of policies' embeddings. Furthermore, DvD adapts the reward-diversity trade off in an online fashion, which facilitates flexible and effective diversity. We demonstrated across a variety of challenging tasks that DvD not only finds diverse, high quality solutions but also manages to maintain strong performances in one-good-solution settings.

References

Agrawal, S. and Goyal, N. (2012). Analysis of Thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 39–1.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D., and Blundell, C. (2020). Agent57: Outperforming the atari human benchmark. *CoRR*, abs/2003.13350.

Ball, P., Parker-Holder, J., Pacchiano, A., Choromanski, K., and Roberts, S. (2020). Ready policy one: World building through active learning. *CoRR*, abs/2002.02693.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI Gym.

Choromanski, K., Pacchiano, A., Parker-Holder, J., Tang, Y., Jain, D., Yang, Y., Iscen, A., Hsu, J., and Sindhwani, V. (2019). Provably robust blackbox optimization for reinforcement learning. In *The Conference on Robot Learning (CoRL)*.

Choromanski, K., Rowland, M., Sindhwani, V., Turner, R. E., and Weller, A. (2018). Structured evolution with compact architectures for scalable policy optimization. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 969–977.

Ciosek, K., Vuong, Q., Loftin, R., and Hofmann, K. (2019). Better exploration with optimistic actor critic. In Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 1787–1798. Curran Associates, Inc.

Conti, E., Madhavan, V., Such, F. P., Lehman, J., Stanley, K. O., and Clune, J. (2018). Improving exploration in Evolution Strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, pages 5032–5043, USA. Curran Associates Inc.

Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. (2015). Robots that can adapt like animals. *Nature*, 521:503–507.

Doan, T., Mazoure, B., Durand, A., Pineau, J., and Hjelm, R. D. (2019). Attraction-repulsion actor-critic for continuous control reinforcement learning. *arXiv*.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. (2019). Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135.

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. (2018). Noisy networks for exploration. In *International Conference on Learning Representations*.

Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596, Stockholmsmässan, Stockholm Sweden. PMLR.

Gaier, A., Asteroth, A., and Mouret, J. (2020). Automating representation discovery with MAP-Elites. *CoRR*, abs/2003.04389.

Gajewski, A., Clune, J., Stanley, K. O., and Lehman, J. (2019). Evolvability ES: Scalable and direct optimization of evolvability. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, pages 107–115, New York, NY, USA. ACM.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. (2018). Meta-reinforcement learning of structured exploration strategies. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 5302–5311. Curran Associates, Inc.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905.

Hartikainen, K., Geng, X., Haarnoja, T., and Levine, S. (2020). Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *International Conference on Learning Representations*.

Hong, Z.-W., Shann, T.-Y., Su, S.-Y., Chang, Y.-H., Fu, T.-J., and Lee, C.-Y. (2018). Diversity-driven exploration strategy for deep reinforcement learning. In *Advances in Neural Information Processing Systems 31*.

Jackson, E. C. and Daley, M. (2019). Novelty search for deep reinforcement learning policy network weights by action sequence edit metric distance. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, page 173–174, New York, NY, USA. Association for Computing Machinery.

Jung, W., Park, G., and Sung, Y. (2020). Population-guided parallel policy search for reinforcement learning. In *International Conference on Learning Representations*.

Khadka, S., Majumdar, S., and Tumer, K. (2019). Evolutionary reinforcement learning for sample-efficient multiagent coordination. *CoRR*, abs/1906.07315.

Khadka, S. and Tumer, K. (2018). Evolution-guided policy gradient in reinforcement learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 1188–1200. Curran Associates, Inc.

Kistemaker, S. and Whiteson, S. (2011). Critical factors in the performance of novelty search. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, page 965–972, New York, NY, USA. Association for Computing Machinery.

Kulesza, A. and Taskar, B. (2011). k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1193–1200.

Kulesza, A. and Taskar, B. (2012). *Determinantal Point Processes for Machine Learning*. Now Publishers Inc., Hanover, MA, USA.

Lehman, J. and Stanley, K. O. (2008). 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.

Lehman, J. and Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223.

Li, A., Spyra, O., Perel, S., Dalibard, V., Jaderberg, M., Gu, C., Budden, D., Harley, T., and Gupta, P. (2019). A generalized framework for population based training. *CoRR*, abs/1902.01894.

Liu, S., Lever, G., Heess, N., Merel, J., Tunyasuvunakool, S., and Graepel, T. (2019). Emergent coordination through competition. In *International Conference on Learning Representations*.

Mania, H., Guy, A., and Recht, B. (2018). Simple random search provides a competitive approach to reinforcement learning. *CoRR*, abs/1803.07055.

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

Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Paul, W., Jordan, M. I., and Stoica, I. (2017). Ray: A distributed framework for emerging AI applications. *CoRR*, abs/1712.05889.

Mouret, J.-B. and Clune, J. (2015). Illuminating search spaces by mapping elites. *ArXiv*, abs/1504.04909.

Nesterov, Y. and Spokoiny, V. (2017). Random gradient-free minimization of convex functions. *Found. Comput. Math.*, 17(2):527–566.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016). Deep exploration via bootstrapped DQN. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 4026–4034. Curran Associates, Inc.

Pacchiano, A., Parker-Holder, J., Tang, Y., Choromanska, A., Choromanski, K., and Jordan, M. I. (2019). Learning to score behaviors for guided policy optimization. *arXiv*.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. (2018). Parameter space noise for exploration. In *International Conference on Learning Representations*.

Pourchot and Sigaud (2019). CEM-RL: Combining evolutionary and gradient-based methods for policy search. In *International Conference on Learning Representations*.

Pugh, J. K., Soros, L. B., and Stanley, K. O. (2016). Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40.

Recht, B., Re, C., Wright, S., and Niu, F. (2011). Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701.

Rothfuss, J., Lee, D., Clavera, I., Asfour, T., and Abbeel, P. (2019). ProMP: Proximal meta-policy search. In *International Conference on Learning Representations*.

Russo, D., Roy, B., Kazerouni, A., and Osband, I. (2017). A tutorial on Thompson sampling. *Foundations and Trends® in Machine Learning*, 11.

Russo, D. and Van Roy, B. (2016). An information-theoretic analysis of Thompson sampling. *J. Mach. Learn. Res.*, 17(1):2442–2471.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. (2019). Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*.

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

Schaul, T., Borsa, D., Ding, D., Szepesvari, D., Ostrovski, G., Dabney, W., and Osindero, S. (2019). Adapting behaviour for learning progress. *CoRR*, abs/1912.06910.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning (ICML)*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2016-2018). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Slivkins, A. (2019). Introduction to multi-armed bandits. *Foundations and Trends in Machine Learning*, 12(1-2):1–286.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 1015–1022, Madison, WI, USA. Omnipress.

Stanley, K., Clune, J., Lehman, J., and Miikkulainen, R. (2019). Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1.

Stanley, R. P. (2011). Enumerative combinatorics volume 1 second edition. *Cambridge studies in advanced mathematics*.

Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. (2017). Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *CoRR*, abs/1712.06567.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Wang, D. and Liu, Q. (2019). Nonlinear stein variational gradient descent for learning diversified mixture models. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6576–6585, Long Beach, California, USA. PMLR.

Yang, R., Sun, X., and Narasimhan, K. (2019). A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 14610–14621. Curran Associates, Inc.

Appendix

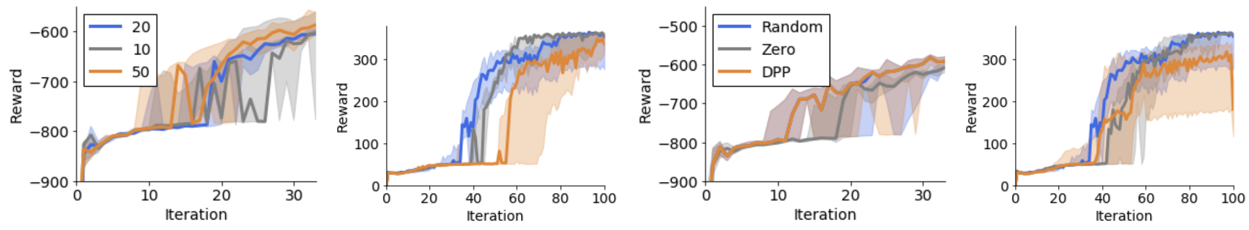
8. Additional Experiment Details

8.1. Ablation Studies

Here we seek to analyze the sensitivity of DvD to design choices made, in order to gain confidence surrounding the robustness of our method.

How sensitive is the embedding to the choice of states? One of the crucial elements of this work is task-agnostic behavioral embedding. Similar to what has been used in all trust-region based policy gradient algorithms (Schulman et al., 2015; 2018), we use a concatenation of actions to represent policy behavior. In all experiments we used this behavioral embedding for both DvD and NSR, thus rendering the only difference between the two methods to be adaptive vs. fixed diversity and joint vs. individual updates.

However, there is still a question whether the design choices we made had an impact on performance. As such, we conducted ablation studies over the number n of states chosen and different strategies for choosing states (for $n = 20$) used to construct behavioral embeddings.

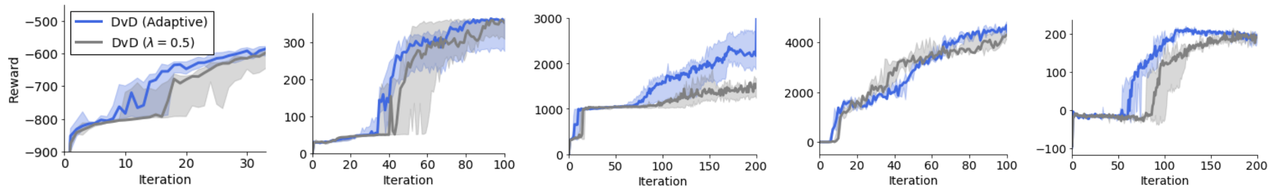


(a) **Point** (b) **Swimmer** (c) **Point** (d) **Swimmer**

Figure 7. The median best performing agent across five seeds. In a) and b) we vary the number of states selected in the random sample, in c) and d) we select 20 states but using different mechanisms. Random corresponds to uniform sampling, Zero to a zero function trained on all seen states, where we select the maximum variance states. DPP stands for a k-DPP (Kulesza and Taskar, 2011).

As we see in Fig. 7, both the number of states chosen and the mechanism for choosing them appear to have minimal impact on the performance. In all cases the point escapes the local maximum (of moving into the area surrounded by walls) and Swimmer reaches high rewards (> 300).

Do we need to adapt? In Fig. 8 we evaluate the effectiveness of the adaptive mechanism, bny running five experiments with the DvD algorithm with fixed λ. While notably we still see strong performance from the joint diversity score (vs. NSR ES in Fig 5), it is clear the adaptive mechanism boosts performance in all cases.

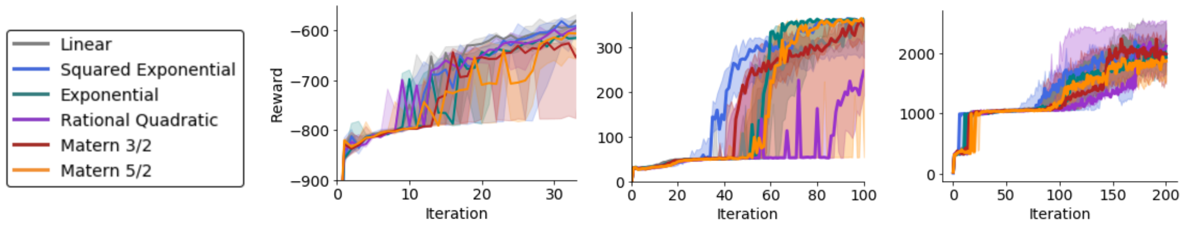


(a) **Point** (b) **Swimmer** (c) **Walker2d** (d) **HalfCheetah** (e) **BipedalWalker**

Figure 8. In this figure we show the median maximum performing agent across five seeds. The only difference between the two curves is the adaptive selection of λ.

Choice of DPP kernel For this work, we used the Squared Exponential (or RBF) kernel for all experiments. This decision was made due to its widespread use in the machine learning community and many desirable properties. However, in order to assess the quality of our method it is important to consider the sensitivity to the choice of kernel.

In Fig. 9 and Table 8.1 we show the result from 5 seeds for a variety of kernels. As we see, in almost all cases the performance is strong, and similar to the Squared Exponential kernel used in the main experiments.



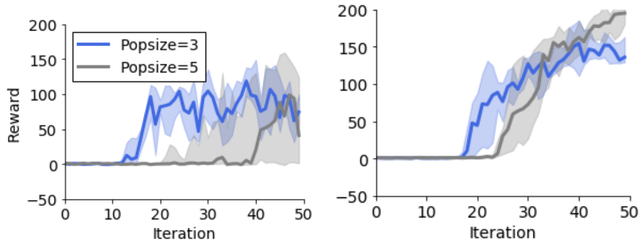
(b) **Point** (c) **Swimmer** (d) **Walker2d**

Figure 9. In this figure we show the median maximum performing agent across five seeds. The only difference between the two curves is the choice of kernel for the behavioral similarity matrix.

Table 1. This table shows the median maximum performing agent across 5 seeds. All algorithms shown are identical aside from the choice of DPP kernel. For point results are the best at 50 iterations, while for point it is 100 and Walker2d it is 200.

	Point	**Swimmer**	**Walker2d**
Squared Exponential	-547.03	354.86	1925.86
Exponential	-561.13	362.83	1929.81
Linear	-551.48	354.37	1944.95
Rational Quadratic	-548.55	246.68	2113.02
Matern $\frac{3}{2}$	-578.05	349.52	1981.66
Matern $\frac{5}{2}$	-557.69	357.88	1866.56

What if the population size is much larger than the number of modes? We also studied the relation between population size and the number of modes learned by the population. Both the tasks considered here have two modes, and intuitively a population of size $M = 3$ should be capable of learning both of them. However, we also considered the case for $M = 5$ (Fig. 10), and found that in fact a larger population was harmful for the performance. This leads to the interesting future work of adapting not only the degree of diversity but the size of the population.



(a) **Cheetah: Forward** (b) **Cheetah: Backward**

Figure 10. Ablation study for the DvD algorithm: population size. The median best performing agent across ten seeds for the Cheetah multi-task environment. In all curves we have the same setting aside from population sizes of $M = 3$ and $M = 5$.

8.2. Hyperparameters

8.3. DvD-ES

In Table 2 we show the method for generating behavioral embeddings used for NSR-ES and our method. We used these settings across **all** experiments, and did not tune them. Number of states corresponds to the number of states used for the embedding, which is the concatenation of the actions from a policy on those states. State selection refers to the mechanism for selecting the states from the buffer of all states seen on the previous iteration. The update frequency is how frequently we re-sample states to use for the embeddings. We do not want this to be too frequent or we will have differnet embeddings every iteration simply as a result of the changing states.

In Table 3 we show the hyprparameters used for the multi-modal experiments. We note the main difference between the two environments is the Ant requires more ES sensings (300 vs. 100) and a larger neural network (64-64 vs. 32-32) than the

Table 2. Configuration for the behavioral embedding, used across all experiments for our method and NSR-ES.

	Value
Number of states	20
State selection	random
Update frequency	20

Cheetah. These settings were used for all three algorithms studied.

Table 3. Parameter configurations for the multi-modal experiments.

	Cheetah	**Ant**
σ	0.001	0.001
η	0.001	0.001
h	32	64
ES-sensings	100	300
State Filter	True	True

In Table 4 we show the hyperparameters used for the uni-model and deceptive reward experiments. The main difference is the size of the neural network for point and Swimmer is smaller (16-16 vs. 32-32) since these environments have smaller state and action dimensions than the others. In addition, we note the horizon H for the point is smaller, as 50 timesteps is sufficient to reach the goal. These settings were used for all three algorithms considered.

Table 4. Parameter configurations for the single mode experiments. The only difference across all tasks was a smaller neural network used for Swimmer and point, since they have a smaller state and action dimensionality.

	point	**Swimmer**	**HalfCheetah**	**Walker2d**	**BipedalWalker**
σ	0.1	0.1	0.1	0.1	0.1
η	0.05	0.05	0.05	0.05	0.05
h	16	16	32	32	32
ES-sensings	100	100	100	100	100
State Filter	True	True	True	True	True
H	50	1000	1000	1000	1600

8.4. DvD-TD3

Our TD3 implementation comes from an open source repository. Since it is a new library from a close collaborator, we will provide the link once anonymity has been lifted.

All parameters are the same as in the original TD3 paper (Fujimoto et al., 2018), apart from neural network architectures and choice of learning rate and batch size, which mirror SAC (Haarnoja et al., 2018).

9. Theoretical Results

9.1. Proof of Theorem 3.3

Proof. We start by recalling that $\mathrm{Div}(\Theta) = \det(\mathbf{K})$. Let $\alpha_1, \cdots, \alpha_M$ be the eigenvalues of $\mathbf{K}$. Since $\mathbf{K}$ is PSD, $\alpha_i \geq 0$ for all i. The following bounds hold:

$$0 \overset{(i)}{\leq} \mathrm{Div}(\Theta) = \prod_{i=1}^{M} \alpha_i \overset{(ii)}{\leq} \left(\frac{\sum_{i=1}^{M} \alpha_i}{M} \right)^M = \left(\frac{\mathrm{trace}(\mathbf{K})}{M} \right)^M \overset{(iii)}{=} 1$$

Inequality (i) follows since $\mathbf{K}$ is a PSD matrix. Inequality (ii) is a consequence of the AM-GM inequality and Equality (iii) follows because all the diagonal entries of $\mathbf{K}$ equal 1. Let $\{\tilde{\pi}_i\}_{i=1}^{M}$ be a set of policies with at least one suboptimal policy and parametrized by $\tilde{\Theta}_t$. Wlog let $\mathcal{R}(\tilde{\pi}_1) + \Delta < \mathcal{R}$. The following holds:

$$\sum_{i=1}^{M} \mathcal{R}(\tilde{\pi}_i) + \lambda_t \mathrm{Div}(\tilde{\Theta}_t) \leq M\mathcal{R} - \Delta + \lambda_t$$

Now observe that for any set of optimal policies $\{\pi_i\}_{i=1}^M$ parametrised by Θ_t the objective in Equation 8 satisfies:

$$\sum_{i=1}^{M} \mathcal{R}(\pi_i) + \lambda_t \mathrm{Div}(\Theta_t) \geq M\mathcal{R}$$

Therefore if $\lambda_t < \Delta$, then:

$$\sum_{i=1}^{M} \mathcal{R}(\tilde{\pi}_i) + \lambda_t \mathrm{Div}(\tilde{\Theta}_t) < \sum_{i=1}^{M} \mathcal{R}(\pi_i) + \lambda_t \mathrm{Div}(\Theta_t)$$

Thus we conclude the objective in Equation 8 can only be maximised when all policies parametrised by Θ_t are optimal. Since $\lambda_t > 0$ and $\mathrm{Div}(\Theta_t)$ is nonzero only when $\{\pi_i\}_{i=1}^M$ are distinct, and there exist at least M distinct solutions, we conclude that whenever $0 < \lambda_t < \Delta$, the maximizer for Equation 8 corresponds to M distinct optimal policies.

$\square$

9.2. Proof of Lemma 3.3

In the case of deterministic behavioral embeddings, it is possible to compute gradients through the whole objective. Notice that $\det(\mathbf{K})$ is then differentiable as a function of the policy parameters. In the case of trajectory based embeddings, differentiating through the determinant is not that simple. Actually it may make sense in this case to use a $\log(\det(\mathbf{K}))$ score instead. This is because of the following lemma:

Lemma 9.1. *The gradient of* $\log\left(\det(\mathbf{K})\right)$ *with respect to* $\theta = \theta^1, \cdots, \theta^M$ *equals:*

$$\nabla_\theta \log\left(\det(\mathbf{K})\right) = -\left(\nabla_\theta \Phi(\theta)\right)\left(\nabla_\Phi \mathbf{K}\right)\mathbf{K}^{-1}$$

Where $\Phi(\theta) = \Phi(\theta^1)\cdots\Phi(\theta^M)$

Proof. We start with:

$$\nabla_{\mathbf{K}} \log\left(\det(\mathbf{K})\right) = -\mathbf{K}^{-1}$$

This is a known result[1].

Consequently,

$$\begin{aligned}\nabla_\theta \log\left(\det(\mathbf{K})\right) &= \left(\nabla_\theta \Phi(\theta)\right)\left(\nabla_\Phi \mathbf{K}\right)\left(\nabla_{\mathbf{K}} \log\left(\det(\mathbf{K})\right)\right)\\ &= -\left(\nabla_\theta \Phi(\theta)\right)\left(\nabla_\Phi \mathbf{K}\right)\mathbf{K}^{-1}\end{aligned}$$

Each of the other gradients can be computed exactly.

$\square$

9.3. Determinants vs. Distances

In this section we consider the following question. Let $k(\mathbf{x}, \mathbf{y}) = \exp(-\|x - y\|^2)$. And $\mathbf{K} \in \mathbb{R}^{M \times M}$ be the kernel matrix corresponding to M agents and resulting of computing the Kernel dot products between their corresponding embeddings $\{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$:

[1]see for example `https://math.stackexchange.com/questions/38701/how-to-calculate-the-gradient-of-log-det-ma`

Theorem 9.2. *For $M \leq 3$, the first order approximation of* $\det(\mathbf{K})$ *is proportional to the sum of the pairwise distances between $\{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$. For $M > 3$ this first order approximation equals* 0.

Proof. Consider the case of a population size $M = 3$, some policy embedding ϕ_i and the exponentiated quadratic kernel. In this setting, the diversity, measured by the determinant of the kernel (or *similarity*) matrix is as follows:

$$\begin{aligned}\det(\mathbf{K}) &= \begin{vmatrix} 1 & k(\mathbf{x}_1,\mathbf{x}_2) & k(\mathbf{x}_1,\mathbf{x}_3) \\ k(\mathbf{x}_2,\mathbf{x}_1) & 1 & k(\mathbf{x}_2,\mathbf{x}_3) \\ k(\mathbf{x}_3,\mathbf{x}_1) & k(\mathbf{x}_3,\mathbf{x}_2) & 1 \end{vmatrix} \\ &= 1 - k(\mathbf{x}_2,\mathbf{x}_3)k(\mathbf{x}_3,\mathbf{x}_2) - k(\mathbf{x}_1,\mathbf{x}_2)\big(k(\mathbf{x}_2,\mathbf{x}_1) - k(\mathbf{x}_3,\mathbf{x}_1)k(\mathbf{x}_2,\mathbf{x}_3)\big) + \\ &\quad k(\mathbf{x}_1,\mathbf{x}_3)\big(k(\mathbf{x}_2,\mathbf{x}_1)k(\mathbf{x}_3,\mathbf{x}_2) - k(\mathbf{x}_3,\mathbf{x}_1)\big) \\ &= 1 - k(\mathbf{x}_1,\mathbf{x}_2)^2 - k(\mathbf{x}_1,\mathbf{x}_3)^2 - k(\mathbf{x}_2,\mathbf{x}_3)^2 + 2k(\mathbf{x}_1,\mathbf{x}_2)k(\mathbf{x}_1,\mathbf{x}_3)k(\mathbf{x}_2,\mathbf{x}_3)\end{aligned}$$

So if we take k to be the squared exponential kernel:

$$\begin{aligned}=&1 - \exp\left(\frac{-\|x_1 - x_2\|^2}{l}\right) - \exp\left(\frac{-\|x_1 - x_3\|^2}{l}\right) - \exp\left(\frac{-\|x_2 - x_3\|^2}{l}\right) + \\ &2\exp\left(\frac{-\|x_1 - x_2\|^2 - \|x_1 - x_3\|^2 - \|x_2 - x_3\|^2}{2l}\right)\end{aligned}$$

Recall that for $|x| << 1$ small enough, $\exp(x) \approx 1 + x$. Substituting this approximation in the expression above we see:

$$\begin{aligned}\det(\mathbf{K}) &\approx \|x_1 - x_2\|^2 + \|x_1 - x_3\|^2 + \|x_2 - x_3\|^2 - \frac{\|x_1 - x_2\|^2 + \|x_1 - x_3\|^2 + \|x_2 - x_3\|^2}{2} \\ &= \frac{\|x_1 - x_2\|^2 + \|x_1 - x_3\|^2 + \|x_2 - x_3\|^2}{2},\end{aligned}$$

which is essentially the mean pairwise l_2 distance. What can we say about these differences (e.g. $\exp$ vs. not)? Does this same difference generalize to $M > 3$?

Approximation for $M > 3$

Recall that for a matrix $\mathbf{A} \in \mathbb{R}^{M \times M}$, the determinant can be written as:

$$\det(\mathbf{A}) = \sum_{\sigma \in \mathbb{S}_M} \text{sign}(\sigma) \prod_{i=1}^{M} \mathbf{A}_{i,\sigma(i)}$$

Where $\mathbb{S}_M$ denotes the symmetric group over M elements. Lets identify $A_{i,j} = \exp\left(-\frac{\|x_i - x_j\|^2}{2}\right)$. Notice that for any $\sigma \in \mathbb{S}_M$, we have the following approximation:

$$\prod_{i=1}^{M} \mathbf{A}_{i,\sigma(i)} \approx 1 - \sum_{i=1}^{M} \frac{\|x_i - x_{\sigma(i)}\|^2}{2} \tag{12}$$

Whenever for all $i, j \in [M]$ the value of $\|x_i - x_j\|^2$ is small.

We are interested in using this termwise approximation to compute an estimate of $\det(\mathbf{A})$. Plugging the approximation in Equation 12 into the formula for the determinant yields the following:

$$\det(\mathbf{A}) \approx \sum_{\sigma \in \mathbb{S}_M} \text{sign}(\sigma) \left(1 - \sum_{i=1}^{M} \frac{\|x_i - x_{\sigma(i)}\|^2}{2}\right)$$

$$= \underbrace{\sum_{\sigma \in \mathbb{S}_M} \text{sign}(\sigma)}_{\text{I}} - \underbrace{\sum_{\sigma \in \mathbb{S}_M} \text{sign}(\sigma) \sum_{i=1}^{M} \frac{\|x_i - x_{\sigma(i)}\|^2}{2}}_{\text{II}}$$

Term I equals zero as it is the sum of all signs of the permutations of $\mathbb{S}_n$ and $n > 1$.

In order to compute the value of II we observe that by symmetry:

$$\text{II} = B \sum_{i<j} \|x_i - x_j\|^2$$

For some $B \in \mathbb{R}$. We show that $B = 0$ for $M > 3$. Let's consider the set $B_{1,2}$ of permutations $\sigma \in \mathbb{S}_M$ for which the sum $\sum_{i=1}^{M} \frac{\|x_i - x_{\sigma(i)}\|^2}{2}$ contains the term $\frac{\|x_1 - x_2\|^2}{2}$. Notice that $B = \frac{1}{2}\sum_{\sigma \in B_{1,2}} \text{sign}(\sigma)$. Let's characterize $B_{1,2}$ more exactly.

Recall every permutation $\sigma \in \mathbb{S}_M$ can be thought of as a product of cycles. For more background on the cycle decomposition of permutations see (Stanley, 2011).

The term $\frac{\|x_1 - x_2\|^2}{2}$ appears whenever the cycle decomposition of σ contains a transition of the form $1 \to 2$ or $1 \leftarrow 2$. It appears twice if the cycle decomposition of σ has the cycle corresponding to a single transposition $1 \leftrightarrow 2$.

Let $\overrightarrow{B}_{1,2}$ be the set of permutations containing a transition of the form $1 \to 2$ (and no transition of the form $1 \leftarrow 2$) $\overleftarrow{B}_{1,2}$ be the set of permutations containing a transition of the form $1 \leftarrow 2$ (and no transition of the form $1 \to 2$) and finally $\overleftrightarrow{B}_{1,2}$ be the set of permutations containing the transition $1 \leftrightarrow 2$.

Notice that:

$$B = \underbrace{\sum_{\sigma \in \overrightarrow{B}_{1,2}} \text{sign}(\sigma)}_{O_1} + \underbrace{\sum_{\sigma \in \overleftarrow{B}_{1,2}} \text{sign}(\sigma)}_{O_2} + 2\underbrace{\sum_{\sigma \in \overleftrightarrow{B}_{1,2}} \text{sign}(\sigma)}_{O_3}$$

We start by showing that for $M > 3$, $O_3 = 0$. Indeed, any $\sigma \in \overleftrightarrow{B}_{1,2}$ has the form $\sigma = (1,2)\sigma'$ where σ' is the cycle decomposition of a permutation over $3, \cdots, M$. Consequently $\text{sign}(\sigma) = -\text{sign}(\sigma')$. Iterating over all possible $\sigma' \in \mathbb{S}_{M-2}$ permutations over $[3, \cdots, M]$ yields the set $\overleftrightarrow{B}_{1,2}$ and therefore:

$$\begin{aligned} O_3 &= -\sum_{\sigma' \in \mathbb{S}_{M-2}} \text{sign}(\sigma') \\ &= 0 \end{aligned}$$

The last equality holds because $M - 2 \geq 2$. We proceed to analyze the terms O_1 and O_2. By symmetry it is enough to focus on O_1. Let c be a fixed cycle structure containing the transition $1 \to 2$. Any $\sigma \in \overrightarrow{B}_{1,2}$ containing c can be written as $\sigma = c\sigma'$ where σ' is a permutation over the remaining elements of $\{1, \cdots, M\} \backslash c$ and therefore $\text{sign}(\sigma) = (-1)^{|c|-1}\text{sign}(\sigma')$. Let $\overrightarrow{B}^c_{1,2}$ be the subset of $\overrightarrow{B}_{1,2}$ containing c.

Notice that:

$$O_1 = \sum_{c | 1 \to 2 \in c, |c| \geq 3} \underbrace{\left[\sum_{\sigma \in \overrightarrow{B}^c_{1,2}} \text{sign}(\sigma)\right]}_{O_1^c}$$

Let's analyze O_1^c:

$$O_1^c = \sum_{\sigma \in \overrightarrow{B}^c_{1,2}} \text{sign}(\sigma)$$

$$= (-1)^{|c|-1} \sum_{\sigma \in \mathbb{S}_{M-|c|}} \text{sign}(\sigma)$$

If $|\{1, \cdots, M\} \backslash c| \geq 2$ this quantity equals zero. Otherwise it equals $(-1)^{|c|-1}$. We recognize two cases, when $|\{1, \cdots, M\} \backslash c| = 0$ and when $|\{1, \cdots, M\} \backslash c| = 1$. The following decomposition holds [2] :

$$O_1 = \Big| \{c | 1 \to 2 \in c, |c| \geq 3 | \{1, \cdots, M\} \backslash c| = 0\} \Big| * (-1)^{M-1} +$$
$$|\{c | 1 \to 2 \in c, |c| \geq 3 | \{1, \cdots, M\} \backslash c| = 1\}| * (-1)^{M-2}$$

A simple combinatorial argument shows that:

$$|\{c | 1 \to 2 \in c, |\{1, \cdots, M\} \backslash c| = 0\}| = (M-2)!$$

Roughly speaking this is because in order to build a size M cycle containing the transition $1 \to 2$, we only need to decide on the positions of the next $M-2$ elements, which can be shuffled in $M-2$ ways. Similarly, a simple combinatorial argument shows that:

$$|\{c | 1 \to 2 \in c, |\{1, \cdots, M\} \backslash c| = 1\}| = (M-2)!$$

A similar counting argument yields this result. First, there are $M-2$ ways of choosing the element that will not be in the cycle. Second, there are $(M-3)!$ ways of arranging the remaining elements to fill up the $M-3$ missing slots of the $M-1$ sized cycle c.

We conclude that in this case $O_1 = 0$.

□

This result implies two things:

1. When $M \leq 3$. If the gradient of the embedding vectors is sufficiently small, the determinant penalty is up to first order terms equivalent to a pairwise distances score. This may not be true if the embedding vector's norm is large.
2. When $M > 3$. The determinant diversity penalty variability is given by its higher order terms. It is therefore not equivalent to a pairwise distances score.

10. Extended Background

For completeness, we provide additional context for existing methods used in the paper.

10.1. Thompson Sampling

Let's start by defining the Thompson Sampling updates for Bernoulli random variables. We borrow the notation from Section 4.2. Let $\mathcal{K} = \{1, \cdots, K\}$ be a set of Bernoulli arms with mean parameters $\{\mu_i\}_{i=1}^K$.

Denote by π_t^i the learner's mean reward model for arm i at time t. We let the learner begin with an independent prior belief over each μ_i, which we denote π_o^i. These priors are beta-distributed with parameters $\alpha_i^o = 1, \beta_i^o = 1$:

$$\pi_o^i(\mu) = \frac{\Gamma(\alpha_i^o + \beta_i)}{\Gamma(\alpha_i^o)\Gamma(\beta_i)} \mu^{\alpha_i^o - 1}(1-\mu)^{\beta_i^o - 1},$$

Where Γ denotes the gamma function. It is convenient to use beta distributions because of their conjugacy properties. It can be shown that whenever we use a beta prior, the posterior distribution is also a beta distribution. Denote α_i^i, β_i^t as the values of parameters α_i, β_i at time t.

Let i_t be the arm selected by Thomson Sampling, as we explained in main body, at time t. After observing reward $r_t \in \{0, 1\}$ the arms posteriors are updated as follows:

[2]Here we use the assumption $M \geq 4$ to ensure that both $|\{c | 1 \to 2 \in c, |c| \geq 3 | \{1, \cdots, M\} \backslash c| = 0\}| > 0$ and $|\{c | 1 \to 2 \in c, |c| \geq 3 | \{1, \cdots, M\} \backslash c| = 1\}| > 0$.

$$(\alpha_i^{t+1}, \beta_i^{t+1}) = \begin{cases} (\alpha_i^t + r_t, \beta_i^t + (1 - r_t)) & \text{if } i = i_t \\ (\alpha_i^t, \beta_i^t) & \text{o.w.} \end{cases}$$

10.2. Reinforcement Learning Algorithms

10.2.1. Evolution Strategies

ES methods cast RL as a blackbox optimization problem. Since a blackbox function $F : \mathbb{R}^d \to \mathbb{R}$ may not even be differentiable, in practice its smoothed variants are considered. One of the most popular ones, the Gaussian smoothing (Nesterov and Spokoiny, 2017) F_σ of a function F is defined as:

$$\begin{aligned} F_\sigma(\theta) &= \mathbb{E}_{\mathbf{g} \in \mathcal{N}(0, \mathbf{I}_d)}[F(\theta + \sigma \mathbf{g})] \\ &= (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} F(\theta + \sigma \mathbf{g}) e^{-\frac{\|\mathbf{g}\|^2}{2}} d\mathbf{g}, \end{aligned}$$

where $\sigma > 0$ is a hyperparameter quantifying the smoothing level. Even if F is nondifferentiable, we can easily obtain stochastic gradients for F_σ. The gradient of the Gaussian smoothing of F is given by the formula:

$$\nabla F_\sigma(\theta) = \frac{1}{\sigma} \mathbb{E}_{\mathbf{g} \sim \mathcal{N}(0, \mathbf{I}_d)}[F(\theta + \sigma \mathbf{g}) \mathbf{g}]. \tag{13}$$

This equation leads to several Monte Carlo gradient estimators used successfully in Evolution Strategies (ES, (Salimans et al., 2017; Choromanski et al., 2018)) algorithms for blackbox optimization in RL. Consequently, it provides gradient-based policy update rules such as:

$$\theta_{t+1} = \theta_t + \eta \frac{1}{k\sigma} \sum_{i=1}^{k} R_i \mathbf{g}_i, \tag{14}$$

where $R_i = F(\theta_t + \sigma \mathbf{g}_i)$ is the reward for perturbed policy $\theta_t + \sigma \mathbf{g}_i$ and $\eta > 0$ stands for the step size.

In practice Gaussian independent perturbations can be replaced by dependent ensembles to further reduce variance of the Monte Carlo estimator of $\nabla F_\sigma(\theta)$ via quasi Monte Carlo techniques.

10.2.2. Novelty Search

In the context of population-based Reinforcement Learning, one prominent approach is the class of novelty search methods for RL (Lehman and Stanley, 2008; 2011; Conti et al., 2018). The NSR-ES algorithm (Conti et al., 2018) maintains a meta-population of M agents, and at each iteration t sequentially samples and an individual member θ_t^m. This agent is perturbed with samples $\mathbf{g}_1^m, \cdots, \mathbf{g}_k^m \sim \mathcal{N}(0, \mathbf{I}_d)$, and then the rewards $R_i^m = F(\theta_t^m + \sigma \mathbf{g}_i^m)$ and embeddings $\Phi(\theta_t^m + \sigma \mathbf{g}_i^m)$ are computed in parallel. The *novelty* of a perturbed policy is then computed as the mean Euclidean distance of its embedding to the embeddings $\Phi(\theta_t^i)$ of the remaining members of the population for $i \neq m$. In order to update the policy, the rewards and novelty scores are normalized (denoted $\widehat{R}_i^m$ and $\widehat{N}_i^m$), and the policy is updated as follows:

$$\theta_{t+1}^m = \theta_t^m + \frac{\eta}{k\sigma} \sum_{i=1}^{k} [(1 - \lambda) \widehat{R}_i^m + \lambda \widehat{N}_i^m] \mathbf{g}_i, \tag{15}$$

where the novelty weight $\lambda > 0$ is a hyperparameter. A value $\lambda = 0$ corresponds to the standard ES approach (see: Eq.14) whereas the algorithm with $\lambda = 1$ neglects the reward-signal and optimizes solely for diversity. A simple template of this approach, with a fixed population size, appears in Alg. 1.

Despite encouraging results on hard exploration environments, these algorithms contain several flaws. They lack a rigorous means to evaluate the diversity of the population as a whole, this means that M policies may fall into $N < M$ conjugacy classes, leading to the illusion of a diverse population on a mean pairwise Euclidean distance metric.

Algorithm 1 Population-Based NSR-ES

```
Input: : learning rate $\eta$, noise standard deviation $\sigma$, number of policies to maintain $M$, number of iterations $T$, embedding $\Phi$, novelty weight $\lambda$.
Initialize: $\{\theta_0^1, \ldots, \theta_0^M\}$.
for $t = 0, 1, \ldots, T-1$ do
    1. Sample policy to update: $\theta_t^m \sim \{\theta_t^1, \ldots, \theta_t^M\}$.
    2. Compute rewards $F(\theta_t^m + \sigma \mathbf{g}_k)$ for all $\mathbf{g}_1, \cdots, \mathbf{g}_k$, sampled independently from $\mathcal{N}(0, \mathbf{I}_d)$.
    3. Compute embeddings $\Phi(\theta_t^m + \sigma \mathbf{g}_k)$ for all $k$.
    4. Let $\widehat{R}_k$ and $\widehat{N}_k$ be the normalized reward and novelty for each perturbation $\mathbf{g}_k$.
    5. Update Agent via Equation 15.
```