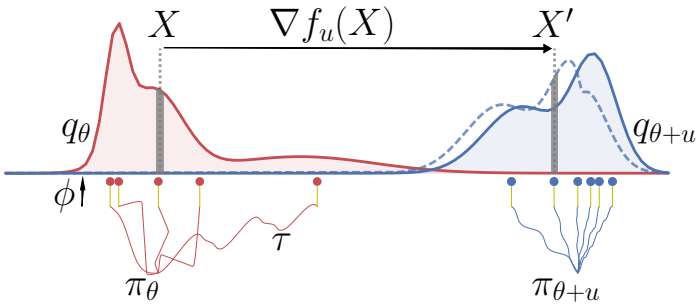

Figure 6: Wasserstein.

A Background

A.1 Policy Optimization

An agent interacting with an environment form a system that can be described by a *state* variable s belonging to a *state space* $\mathcal{S}$. In the Markov Decision Process (MDP) setting, the agent can interact with the environment by taking an action a from a set of possible actions $\mathcal{A}$ given the current state s of the system. As a consequence, the system moves to a new state s' according to a probability transition function $P(s'|a,s)$ which describes the probability of moving to state s' given the previous state s and action a. The agent also receives a partial reward r which can be expressed as a possibly randomized function of the new state s', $r = r(s')$. The agent has access to a set of possible *policies* $\pi_\theta(a|s)$ parametrized by $\theta \in \mathbb{R}^p$ and that generates an action a given a current state s. Thus, each policy can be seen as a probability distribution conditioned a state s. Using the same policy induces a whole trajectory of state-action-rewards $\tau = (s_t, a_t, r_t)_{t\geq 0}$ which can be viewed as a sample from a trajectory distribution $\mathbb{P}_\theta$ defined over the space of possible trajectories τ. Hence, for a given random trajectory τ induced by a policy π_θ, the agent receives a total discounted reward $R(\tau) := \sum_{t=1}^{\infty} \gamma^{t-1} r(s_t)$ with discount factor $0 < \gamma < 1$. This allows to define the *value* function as the expected total reward conditioned on a particular initial state s:

$$V_\theta(s_t) = \mathbb{E}_{\mathbb{P}_\theta|s_t}\left[\sum_{l=1}^{\infty} \gamma^{l-1} r(s_{l+t})\right]. \tag{16}$$

When the gradient of the *score function* $\nabla \log \pi_\theta(a|s)$ is available, the policy gradient theorem allows to express the gradient of $\mathcal{R}(\theta)$:

$$\nabla_\theta \mathcal{R}(\theta) = \mathbb{E}_{\mathbb{P}_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \nabla \log \pi_\theta(a_t|s_t) A_\theta(s_t, a_t)\right] \tag{17}$$

where the expectation is taken over trajectories τ under $\mathbb{P}_\theta$ and $A_\theta(s,a)$ represents the *advantage* function which can be expressed in terms of the value function $V_\theta(s)$ in terms of

$$A_\theta(s_t, a_t) = \mathbb{E}_{s_{t+1}|s_t,a_t}\left[r(s_{t+1}) + \gamma V_\theta(s_{t+1})\right] - V_\theta(s_t).$$

The agent seeks an optimal policy $\pi_{\theta^\star}$ that maximizes the expected total reward under the trajectory distribution: $\mathcal{R}(\theta) = \mathbb{E}_{\mathbb{P}_\theta}[R(\tau)]$.

B Wasserstein Natural gradient

Connection to the Fisher natural gradient and proximal methods. Both WNG and FNG are obtained from a proximity measure between probability distributions:

Proposition 2 *Let $D(\theta, \theta')$ be either the KL-divergence $KL(\pi_\theta, \pi_{\theta'})$ or the Wasserstein-2 distance between the behavioral distributions $W_2(q_\theta, q_{\theta'})$ and let g^D be either the FNG g^F or WNG g^W, then*

$$g_k^D = \lim_{\beta \to +\infty} \arg\max_u \beta \left(\mathcal{L}(\theta_k + \beta^{-1} u) - \mathcal{L}(\theta_k) - \frac{\beta}{2} D\left(\theta_k, \theta_k + \beta^{-1} u\right) \right) \tag{18}$$

Equation (18) simply states that the both WNG and FNG arise as limit cases of penalized objectives provided the strength of the penalty β diverges to infinity and the step-size is shrank proportionally to β^{-1}. An additional global rescaling by β of the total objective prevents it from collapsing to 0. Intuitively, performing a Taylor expansion of Equation (18) recovers an equation similar to Equation (8). Equation (18) shows that using a penalty that encourages **global** proximity between successive policies, it is possible to recover the **local** geometry of policies (captured by the local) by increasing the strength of the penalty using appropriate re-scaling. This also informally shows why both natural gradients are said to be invariant to re-parametrization ((Arbel et al., 2020, Proposition 1)), since both KL and W_2 remains unchanged if q_θ is parameterized in a different way.

C ALGORITHM FOR ESTIMATING WNG

Algorithm 3: Efficient Wasserstein Natural Gradient

Input mini-batch of samples $\{X_n\}_{n=1}^N$ distributed according to q_θ, gradient direction $\hat{g}$, basis functions $h_1, ..., h_M$, regularization parameter ϵ.
Output Wasserstein Natural gradient $\hat{g}^W$
Compute a matrix C of shape $M \times Nd$ using $C_{m,(n,i)} \leftarrow \partial_i h_m(X_n)$.
Compute similarity matrix $L \leftarrow \frac{1}{N} CC^T$.
Compute surrogate vector V using Equation (11).
for iteration$= 1, 2, ...M$ do **do**
    Use automatic differentiation on V_m to compute Jacobian matrix J in Equation (10).
end for
Compute a matrix D of shape $M \times M$ using $D \leftarrow JJ^\top + \epsilon L$.
Compute a vector b of size M using $b \leftarrow J\hat{g}$.
Solve linear system of size M : $b \leftarrow$ `solve` (D, b)
Return $\hat{g}^W \leftarrow \frac{1}{\epsilon}(\hat{g} - J^\top b)$

Algorithm 4: Similarity

Input N, M, K, $\hat{g}$, ϵ
Ouput $\hat{g}^W$
Get a minibatch of samples $Z_1, ...Z_N$.
Compute embedded samples $X_i = B_\theta(Z_i) \in \mathbb{R}^d$
Randomly sample $Y_1, ...Y_M$ from $X_1, ...X_N$ without replacement
Randomly sample indices $i_1, ..., i_M$ from $\{1, ..., p\}$ without replacement
Compute a similarity matrix C of shape $M \times Nd$:

$$C_{m,(n,i)} \leftarrow \partial_{i_m} \partial_{i+d} K(Y_m, X_n) \tag{19}$$

Compute a summary vector V of size M

$$V_m \leftarrow \frac{1}{N} \sum_{i=1}^{N} \partial_{i_m} K(Y_m, X_n) \tag{20}$$

for iteration$= 1, 2, ...M$ do **do**
    Using automatic differentiation compute the gradient of V_m w.r.t $\theta \in \mathbb{R}^p$.
    Collect gradients in a Jacobian matrix J of shape $M \times p$.
end for
Compute a matrix G of shape $M \times M$ using $G \leftarrow JJ^\top + \frac{\epsilon}{N} CC^\top$.
Collect product $J\hat{g}$ in a vector $\widetilde{g}$ of shape M.
Solve linear system of size M : $b \leftarrow$ `solve` $(G, \widetilde{g})$
Return $\frac{1}{\epsilon}(\hat{g} - J^\top b)$

D ADDITIONAL TECHNICAL DETAILS

Consider a policy π_θ and denote by $\mathcal{R}(\tau)$ the reward of a sample τ from $\mathbb{P}_{\pi_\theta}$. The agent tries to maximize the expected reward:

$$l(\theta) = \mathbb{E}_{\mathbb{P}_{\pi_\theta}}\left[\mathcal{R}(\tau)\right]. \tag{21}$$

We are interested in the setting where $l(\theta)$ is maximized using stochastic gradient methods while maintaining a small Wasserstein distance between successive Behavioral distributions. More precisely, denote by Φ the Behavioral Embedding Map as in Pacchiano et al. (2019) which maps a trajectory τ to a feature (*final state*, *actions* or *total reward*) and let $\mathbb{P}^{\phi}_{\pi_\theta}$ be the push-forward distribution of $\mathbb{P}_{\pi_\theta}$ by ϕ as in Pacchiano et al. (2019). We would like to ensure that $W_2(\mathbb{P}^{\phi}_{\pi_{\theta_{k+1}}}, \mathbb{P}^{\phi}_{\pi_{\theta_k}})$ remains small between two successive iterates θ_k and θ_{k+1}.

To this end, Pacchiano et al. (2019) consider the penalized objective:

$$f(\theta) := -l(\theta) + \frac{\beta}{2} WD_\lambda(\mathbb{P}^{\phi}_{\pi_\theta}, \mathbb{P}^{\phi}_{\pi_{\theta_k}}) \tag{22}$$

Where WD_λ is the entropy regularized Wasserstein distance, with regularization parameter $\lambda > 0$. Hence, minimizing f in θ gives the next iterate θ_{k+1} while maintaining a small distance $WD_\lambda(\mathbb{P}^{\phi}_{\pi_{\theta_{k+1}}}, \mathbb{P}^{\phi}_{\pi_{\theta_k}})$. However, this process is costly since each iteration requires minimizing Equation (22). Instead, we propose to use the Wasserstein Natural gradient estimator introduced in Arbel et al. (2020) which is formally obtained by letting β tends to ∞. In this case, Equation (22) is replaced by the following cost:

$$f(\theta) := -\nabla l(\theta_k)^\top(\theta - \theta_k) + \frac{\beta}{2}(\theta - \theta_k)^\top G(\theta_k)(\theta - \theta_k) \tag{23}$$

where $G(\theta_k)$ is the Wasserstein Information Matrix. Minimizing Equation (23) gives the following update:

$$\theta_{k+1} = \theta_k + \beta G(\theta_k)^{-1}\nabla l(\theta_k). \tag{24}$$

While computing and inverting $G(\theta_k)^{-1}$ in closed form is expensive, Arbel et al. (2020) shows that scalable approximation to Equation (24) are obtained using:

$$\theta_{k+1} = \theta_k + \frac{\beta}{\epsilon}\left(\nabla l(\theta_k) - M_{\theta_k}(f^*)\right) \tag{25}$$

where $\epsilon > 0$ is a regularization parameter and $M_\theta(f)$ is a vector whose components are given by:

$$M_\theta(f)_i = \lim_\gamma \gamma^{-1}\left(\mathbb{E}_{\mathbb{P}_{\pi_{\theta+\gamma e_i}}}[f(\phi(\tau))] - \mathbb{E}_{\mathbb{P}_{\pi_\theta}}[f(\phi(\tau))]\right) \tag{26}$$

and $f^\star$ is obtained by minimizing the objective $\mathcal{J}(f)$ over an RKHS space:

$$\mathcal{J}(f) := \mathbb{E}_{\mathbb{P}_{\pi_\theta}}[\|\nabla f(\phi(\tau))\|^2] + \frac{1}{\epsilon}\|\nabla l(\theta_k) - M_{\theta_k}(f)\|^2 + \lambda\|f\|^2_{\mathcal{H}} \tag{27}$$

To figure out exact expression of the updates, it remains to find explicit expressions for $M_\theta(f)$. We distinguish two cases **gradient-based optimization** and **gradient-free optimization**.

D.1 GRADIENT-BASED OPTIMIZATION

When the trajectories τ are obtained through an mapping $h_\theta(z, s) := \tau$ where z is a latent noise and s is a state, then Arbel et al. (2020) shows that $M_\theta(f)$ is expressed in terms of the jacobian of h_θ:

$$M_\theta(f) = \mathbb{E}_{Z,S}\left[\nabla f(h_\theta(Z, S))\nabla_\theta h_\theta(Z, S)\right] \tag{28}$$

This leads to the algorithm implemented in Arbel et al. (2020).

Sometimes the mapping $h_\theta(z, s) := \tau$ is not differentiable neither is the cost θ. In this case, it advantageous to use Evolution Strategies. In our setting, this means that the policy π_θ is replaces by a perturbed policy π_ψ where ψ is a sample from a gaussian $p_{\theta,\sigma}$ with mean θ and variance σ. This

allows to obtain gradient wrt θ after smoothing by averaging over several draws ψ. More precisely, the loss l is replaced by the averaged loss $\bar{l}(\theta)$:

$$\bar{l}(\theta) = \mathbb{E}_{\psi \sim p_{\theta,\sigma}}[l(\psi)]. \tag{29}$$

with gradient simply given by:

$$\nabla \bar{l}(\theta) := \frac{1}{\sigma} \mathbb{E}_{\psi \sim p_{\theta,\sigma}}[l(\psi)(\psi - \theta)] \tag{30}$$

Moreover, the vector $M_\theta(f)$ is replaced by an averaged one:

$$\bar{M}_\theta(f)_i = \lim_\gamma \gamma^{-1} \left(\mathbb{E}_{\psi \sim p_{\theta+\gamma e_i,\sigma}}[\mathbb{E}_{\mathbb{P}_{\pi_\psi}}[f(\phi(\tau))]] - \mathbb{E}_{\psi \sim p_{\theta,\sigma}}[\mathbb{E}_{\mathbb{P}_{\pi_\psi}}[f(\phi(\tau))]] \right) \tag{31}$$

In this case it is easy to see that $\bar{M}_\theta(f)$ admits a simple expression:

$$\bar{M}_\theta(f) = \frac{1}{\sigma} \mathbb{E}_{\psi \sim p_{\theta,\sigma}}[\mathbb{E}_{\mathbb{P}_{\pi_\psi}}[f(\phi(\tau))](\psi - \theta)] \tag{32}$$

Finally, the objective in Equation (27) is replaced by an averaged one $\bar{\mathcal{J}}(f)$:

$$\bar{\mathcal{J}}(f) := \mathbb{E}_{\psi \sim p_{\theta,\sigma}} \mathbb{E}_{\mathbb{P}_{\pi_\psi}}[\|\nabla f(\phi(\tau))\|^2] + \frac{1}{\epsilon} \|\nabla \bar{l}(\theta_k) - \bar{M}_{\theta_k}(f^\star)\|^2 + \lambda \|f\|^2_{\mathcal{H}} \tag{33}$$

This avoids computing the jacobian wrt to $h_\theta(Z, S)$ which might be unknown. Finally, minimizing $\bar{\mathcal{J}}(f)$ can be done using Nystrom methods as in Arbel et al. (2020).

Expression of the estimator Let ψ^n be noisy parameters with mean θ and X_n the corresponding behavioral embeddings. Let $(Y_m)_{1 \leq m \leq M}$ be M uniform subsamples from $(X_n)_{1 \leq n \leq N}$. The estimator of the natural gradient is given by:

$$\widehat{\nabla^W \mathcal{L}(\theta)} = \frac{1}{\epsilon} \left(D(\theta)^{-1} - D(\theta)^{-1} T^\top \left(T D(\theta)^{-1} T^\top + \lambda \epsilon K + \frac{\epsilon}{N} C C^\top \right)^\dagger T D(\theta)^{-1} \right) \widehat{\nabla \mathcal{L}(\theta)}, \tag{34}$$

where C and K are matrices in $\mathbb{R}^{M \times Nd}$ and $\mathbb{R}^{M \times M}$ given by

$$C_{m,(n,i)} = \partial_{i_m} \partial_{i+d} k(Y_m, X_n), \qquad K_{m,m'} = \partial_{i_m} \partial_{i_{m'}+d} k(Y_m, Y_{m'}), \tag{35}$$

while T is a matrix in $\mathbb{R}^{M \times q}$ given by:

$$T = \frac{1}{N\sigma} \sum_{i=1}^{N} \partial_{i_m} k(Y_m, X_n)(\psi^n - \theta) \tag{36}$$

The matrix $D(\theta)$ is diagonal and can be computed in a similar way as in Arbel et al. (2020).

E Additional Experimental Details

E.1 Policy Gradient Tasks

We conserve all baseline and shared hyperparameters used by Pacchiano et al. (2019). More precisely, for each task we ran a hyperparameter sweep over learning rates in the set {1e-5, 5e-5, 1e-4, 3e-4}, and used the concatenation-of-actions behavioral embedding $\Phi(\tau) = [a_0, a_1, \ldots, a_T]$ with the base network implementation the same as Dhariwal et al. (2017).

The WNG hyperparameters were also left the same as in Arbel et al. (2020).

E.2 Evolution Strategies Tasks

As with the policy gradient tasks, we conserved all baseline and shared hyperparameters used by Pacchiano et al. (2019). Specifically, for the point task, we set the learning rate to be $\eta = 0.1$, the standard deviation of the noise to be $\sigma = 0.01$, the rollout length H was 50 time steps, and the behavioral embedding function to be the last state $\Phi(\tau) = s_H$. For the quadruped task we set $\eta = 0.02$, $\sigma = 0.02$, $H = 400$, and $\Phi(\tau) = \sum_{t=0}^{H} r_t \left(\sum_{i=0}^{t} e_i \right)$ (reward-to-go encoding; see Pacchiano et al. (2019) for more details). Both tasks used 1000-dimensional random features and embeddings from the $n = 2$ previous policies to compute the WD.

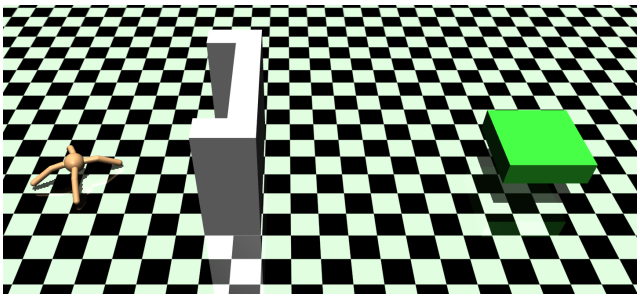

Figure 7: A visualization of the quadruped task. The agent receives receives more reward the closer it is to the goal (green). A naïve agent will get stuck in the local maximum at the wall if it attempts to move directly to the goal.