Uncertainty Estimation Using Riemannian Model Dynamics for Offline Reinforcement Learning

Guy Tennenholtz*
Technion Institute of Technology

Shie Mannor
Technion Institute of Technology & Nvidia Research

Abstract

Model-based offline reinforcement learning approaches generally rely on bounds of model error. Estimating these bounds is usually achieved through uncertainty estimation methods. In this work, we combine parametric and nonparametric methods for uncertainty estimation through a novel latent space based metric. In particular, we build upon recent advances in Riemannian geometry of generative models to construct a pullback metric of an encoder-decoder based forward model. Our proposed metric measures both the quality of out-of-distribution samples as well as the discrepancy of examples in the data. We leverage our method for uncertainty estimation in a pessimistic model-based framework, showing a significant improvement upon contemporary model-based offline approaches on continuous control and autonomous driving benchmarks.

1 Introduction

Offline Reinforcement Learning (RL) [Levine et al., 2020], a.k.a. batch-mode RL [Ernst et al., 2005, Riedmiller, 2005, Fonteneau et al., 2013], involves learning a policy from data sampled by a potentially suboptimal policy. Offline RL seeks to *surpass* the average performance of the agents that generated the data. Traditional methodologies fall short in offline settings, causing overestimation of the return [Buckman et al., 2020, Wang et al., 2020, Zanette, 2020].

One approach to overcome this in model-based settings is to penalize the return in out of distribution (OOD) regions, as depicted in Figure 1. In this manner, the agent is constrained to stay “near" areas of low model error, thereby limiting possible overestimation. However, reliable estimates of model error are key to the success of such methods.

Estimating model error in OOD regions can be achieved through uncertainty estimation [Yu et al., 2020]. Methods of parametric uncertainty estimation such as bootstrap ensembles [Efron, 1982], Monte Carlo Dropout [Gal and Ghahramani, 2016], and randomized priors [Osband et al., 2018], may be susceptible to poor model specification and are most effective when dealing with large datasets. In contrast, nonparametric methods such as k-nearest neighbors (k-NN) [Villa Medina et al., 2013, Fathabadi et al., 2021] are beneficial in regions of limited data, yet require a proper metric to be used.

We propose to combine parametric and nonparametric methods for uncertainty estimation. Particularly, we define a novel Riemmannian metric which captures the epistemic and aleatoric uncertainty of a generative parametric forward model. This distance metric is then applied to measure the average geodesic distance to the k-nearest neighbors in the data. We derive analytical expressions for our metric and provide an efficient manner to estimate it. We then demonstrate the effectiveness of our metric for penalizing an offline RL agent compared to contemporary approaches on continuous control and autonomous driving benchmarks. As we empirically show, common approaches, including statistical bootstrap ensemble or Euclidean distances in latent space, do not necessarily capture the underlying degree of error needed for model-based offline RL.

*Correspondence to guytenn@gmail.com

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

2 Preliminaries

2.1 Offline Reinforcement Learning

We consider the standard Markov Decision Process (MDP) framework [Puterman, 2014] defined by the tuple $(\mathcal{S}, \mathcal{A}, r, P, \alpha)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $r : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ the reward function, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ the transition kernel, and $\alpha \in (0, 1)$ is the discount factor.

In the online setting of reinforcement learning (RL), the environment initiates at some state $s_0 \sim \rho_0$. At any time step the environment is in a state $s \in \mathcal{S}$, an agent takes an action $a \in \mathcal{A}$ and receives a reward $r(s, a)$ from the environment as a result of this action. The environment transitions to state s' according to the transition function $P(\cdot|s, a)$. The goal of online RL is to find a policy $\pi(a|s)$ that maximizes the expected discounted return $v^\pi = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \alpha^t r(s_t, a_t) | s_0 \sim \rho_0 \right]$.

Unlike the online setting, the offline setup considers a dataset $\mathcal{D}_n = \{s_i, a_i, r_i, s'_i\}_{i=1}^n$ of transitions generated by some unknown agents. The objective of offline RL is to find the best policy in the test environment (i.e., real MDP) given only access to the data generated by the unknown agents.

2.2 Riemannian Manifolds

We define the Riemannian pullback metric, a fundamental component of our proposed method. We refer the reader to Carmo [1992] for further details on Riemannian geometry.

We are interested in studying a smooth surface M with a Riemannian metric g. A Riemannian metric is a smooth function that assigns a symmetric positive definite matrix to any point in M. At each point $z \in M$ a tangent space $T_z M$ specifies the pointing direction of vectors "along" the surface.

Definition 1. *Let M be a smooth manifold. A Riemannian metric g on M changes smoothly and defines a real scalar product on the tangent space $T_z M$ for any $z \in M$ as*

$$g_z(x, y) = \langle x, y \rangle_z = \langle x, G(z)y \rangle, \quad x, y \in T_z M,$$

where $G(z) \in \mathbb{R}^{d_z \times d_z}$ is the corresponding metric tensor. (M, g) is called a Riemannian manifold.

The Riemannian metric enables us to easily define geodesic curves. Consider some differentiable mapping $\gamma : [0, 1] \mapsto M \subseteq \mathbb{R}^{d_z}$, such that $\gamma(0) = z_0, \gamma(1) = z_1$. The length of the curve γ measured on M is given by

$$L(\gamma) = \int_0^1 \sqrt{\left\langle \frac{\partial \gamma(t)}{\partial t}, G(\gamma(t)) \frac{\partial \gamma(t)}{\partial t} \right\rangle} dt. \quad (1)$$

The geodesic distance $d(z_1, z_2)$ between any two points $z_1, z_2 \in M$ is then the infimum length over all curves γ for which $\gamma(0) = z_0, \gamma(1) = z_1$. That is, $d(z_1, z_2) = \inf_\gamma L(\gamma) \quad \text{s.t. } \gamma(0) = z_0, \gamma(1) = z_1$. The geodesic distance can be found by solving a system of nonlinear ordinary differential equations (ODEs) defined in the intrinsic coordinates [Carmo, 1992].

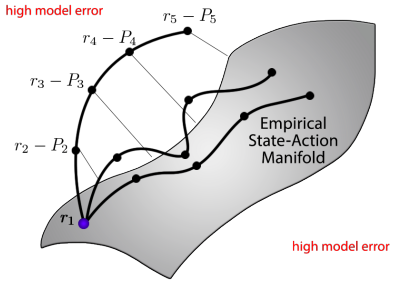


Figure 1: Illustration of a data manifold corresponding to model error. Curved lines represent trajectories at different stages of training. Agent incurs a penalty in areas of high model error.

Pullback Metric. Assume an ambient (observation) space $\mathcal{X}$ and its respective Riemannian manifold $(M_\mathcal{X}, g_\mathcal{X})$. Learning $g_\mathcal{X}$ can be hard (e.g., learning the distance metric between images). Still, it may be captured through a low dimensional submanifold. As such, it is many times convenient to parameterize the surface $M_\mathcal{X}$ by a latent space $\mathcal{Z} = \mathbb{R}^{d_\mathcal{Z}}$ and a smooth function $f : \mathcal{Z} \mapsto \mathcal{X}$, where $\mathcal{Z}$ is a low dimensional latent embedding space. As learning the manifold $M_\mathcal{X}$ can be hard, we turn to learning the immersed low dimensional submanifold $M_\mathcal{Z}$ (for which the chart maps are trivial, since $\mathcal{Z} = \mathbb{R}^{d_\mathcal{Z}}$). Given a curve $\gamma : [0, 1] \mapsto M_\mathcal{Z}$ we have that $\left\langle \frac{\partial f(\gamma(t))}{\partial t}, G_\mathcal{X}(f(\gamma(t))) \frac{\partial f(\gamma(t))}{\partial t} \right\rangle = \left\langle \frac{\partial \gamma(t)}{\partial t}, J_f^T(\gamma(t)) G_\mathcal{X}(f(\gamma(t))) J_f(\gamma(t)) \frac{\partial \gamma(t)}{\partial t} \right\rangle$, where the Jacobian matrix $J_f(z) = \frac{\partial f}{\partial z} \in \mathbb{R}^{d_\mathcal{X} \times d_\mathcal{Z}}$

maps tangent vectors in $TM_{\mathcal{Z}}$ to tangent vectors in $TM_{\mathcal{X}}$. The induced metric is thus given by

$$G_f(z) = J_f(z)^T G_{\mathcal{X}}(f(z)) J_f(z). \tag{2}$$

The metric G_f is known as the *pullback metric*, as it "pulls back" the metric $G_{\mathcal{X}}$ on $\mathcal{X}$ back to G_f via $f : \mathcal{Z} \mapsto \mathcal{X}$. The pullback metric captures the intrinsic geometry of the immersed submanifold while taking into account the ambient space $\mathcal{X}$. The geodesic distance in ambient space is captured by geodesics in the latent space $\mathcal{Z}$, reducing the problem to learning the latent embedding space $\mathcal{Z}$ and the observation function f. Indeed, learning the latent space and observation function f can be achieved through a encoder-decoder framework, such as a VAE [Arvanitidis et al., 2018].

3 Background: Penalty of Uncertainty for Offline Reinforcement Learning

A key element of model-based RL methods involves estimating a model $\hat{P}(s'|s,a)$ to construct a pessimistic MDP[2]. This work builds upon MOPO, a recently proposed model-based offline RL framework [Yu et al., 2020]). Particularly, we assume access to an approximate MDP $(\mathcal{S}, \mathcal{A}, \hat{r}, \hat{P}, \alpha)$ (e.g., trained by maximizing the likelihood of the data), and define a penalized MDP $(\mathcal{S}, \mathcal{A}, \tilde{r}, \hat{P}, \alpha)$, such that for all $s \in \mathcal{S}, a \in \mathcal{A}$, $\tilde{r}(s,a) = \hat{r}(s,a) - \lambda c(P(\cdot|s,a), \hat{P}(\cdot|s,a))$, where c penalizes the reward according to model error (e.g., the total variation distance) and $\lambda > 0$. The offline RL problem is then solved by executing an online algorithm in the reward-penalized MDP. Unfortunately, as $P(\cdot|s,a)$ is unknown, and can only be estimated from the data, $c(P(\cdot|s,a), \hat{P}(\cdot|s,a))$ cannot be calculated. Nevertheless, one can attempt to upper bound the distance, i.e., for some $U : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, $c(P(\cdot|s,a), \hat{P}(\cdot|s,a)) \leq U(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$. In this work we propose to use a naturally induced metric of a variational forward model, which we show can introduce an effective penalty for offline RL. In Section 4 we define this metric, and finally, we leverage it in Section 4.4.

4 Metrics of Uncertainty

As described in the previous section, our goal is to estimate model error in order to penalize the agent in out of distribution (OOD) regions. Yu et al. [2020] proposed to achieve this through bootstrap ensembles, an out of distribution uncertainty estimation technique. Alternatively, we propose to employ a well-known nonparametric approach for uncertainty estimation [Villa Medina et al., 2013, Fathabadi et al., 2021], namely k-nearest neighbors (k-NN). Specifically, for any $s, a \in \mathcal{S} \times \mathcal{A}$, we estimate model error by

$$U(s,a) = \frac{1}{k} \sum_{(s_i,a_i) \in \text{NN}_k(s,a)} d((s,a),(s_i,a_i)), \tag{3}$$

where $d : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}_+$ is a distance metric, and $\text{NN}_k(s,a)$ is the set of k-nearest neighbors of (s,a) in $\mathcal{D}_n$ according to the distance metric d.

A question arises: how to choose d? Using the Euclidean distance in ambient (state-action) space is usually a bad choice (e.g., the ℓ_2 distance between natural images is not necessarily meaningful). Moreover, to correctly measure the error, model dynamics should be somehow taken into consideration. We therefore consider an alternative approach which leverages the latent space of a variational forward model, as described next.

4.1 A Variational Latent Model of Dynamics

We begin by modeling $\hat{P}(s'|s,a)$ using a generative latent model. Specifically, we consider a latent model which consists of an encoder $E : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{B}(\mathcal{Z})$ and a decoder $f_D : \mathcal{Z} \mapsto \mathcal{B}(\mathcal{S})$, where $\mathcal{B}(\mathcal{X})$ is set of probability measures on the Borel sets of $\mathcal{X}$. While the encoder E learns a latent representation of s, a, the decoder f_D estimates the next state s' according to $P(\cdot|s,a)$. This model corresponds to the decomposition $P(\cdot|s,a) = f_D(\cdot|E(s,a))$. Such a model can be trained by maximizing the Evidence Lower BOund (ELBO, Kingma and Welling [2013]) over the data.

[2]Pessimism is a key element of offline RL algorithms [Jin et al., 2020], limiting overestimation of a trained policy due to the distribution shift between the data and the trained policy.

Algorithm 1 GELATO: Geometrically Enriched LATent model for Offline reinforcement learning

1: **Input:** Offline dataset $\mathcal{D}_n$, RL algorithm
2: Train variational latent forward model on dataset $\mathcal{D}_n$ by maximizing ELBO.
3: Construct approximate MDP $(\mathcal{S}, \mathcal{A}, \hat{r}, \hat{P}, \alpha)$
4: Use distance $d_{\mathcal{Z}}$ induced by pullback metric $G_{f_{D,U} \circ f_F}$ (Theorem 2) to penalize reward $\tilde{r}_d(s,a) = \hat{r}(s,a) - \lambda U(s,a)$ where $U(s,a) = \sum_{(s_i,a_i) \in \text{NN}_k(s,a)} d_{\mathcal{Z}}(E(s,a), E(s_i,a_i))$
5: Train RL algorithm over penalized MDP $(\mathcal{S}, \mathcal{A}, \tilde{r}_d, \hat{P}, \alpha)$

That is, given a prior $P(z)$, we model $E_\phi, f_{D,\theta}$ as parametric functions and maximize the ELBO, $\max_{\theta,\phi} \mathbb{E}_{E_\phi(z|s,a)} \Big[\log f_{D,\theta}(s'|z)\Big] - D_{KL}(E_\phi(z|s,a)||P(z))$ We refer the reader to the appendix for an exhaustive overview of training VAEs by maximum likelihood and the ELBO.

Recall that we wish to find a good metric for estimating model error. Having learned a latent model for $\hat{P}(s'|s,a)$, its latent space $\mathcal{Z}$ can be used to define the metric d in Equation (3), i.e., the Euclidean distance between latent representations of state-action pairs. Unfortunately, as was previously shown [Arvanitidis et al., 2018], latent codes in variational models contain sharp discontinuities, rendering Euclidean distances in latent space unreliable and inaccurate (as we will also demonstrate in our experiments). Instead, we propose to use the natural induced metric of our latent model, as described in the following subsection.

4.2 The Pullback Metric of Model Dynamics

In this part we define the metric d that will be used to approximate model error in Equation (3). Specifically, we consider a Riemannian submanifold defined by a latent space $\mathcal{Z}$ and observation function f, which induces minimum energy in the ambient space. We will later choose $\mathcal{Z}$ to be the latent space of our variational model (i.e., encoded state-action) and f to be the decoder function f_D of next state transitions. We define the distance metric formally below.

Definition 2. *We define a Riemannian submanifold $(\mathcal{M}_{\mathcal{Z}}, g_{\mathcal{Z}})$ by a differential function $f : \mathcal{Z} \mapsto \mathcal{S}$ and latent space $\mathcal{Z}$ such that*

$$d_{\mathcal{Z}}(z_1, z_2) = \inf_{\gamma} \int_0^1 \left\| \frac{\partial f(\gamma(t))}{\partial t} \right\| dt \quad \textit{s.t. } \gamma(0) = z_1, \gamma(1) = z_2.$$

A similar metric has been used in previous work on generative latent models [Chen et al., 2018, Arvanitidis et al., 2018]. By choosing f to be the decoder function f_D, latent codes that are close w.r.t. $d_{\mathcal{Z}}$ induce curves of minimal energy in the ambient observation space (i.e., next state). This metric is closely related to the pullback metric (see Section 2.2), as shown by the following proposition.

Proposition 1. *Let $(\mathcal{M}_{\mathcal{Z}}, g_{\mathcal{Z}})$ as defined above. Then $G_f(z) = J_f^T(z) J_f(z)$, for any $z \in \mathcal{Z}$.*

Indeed, Proposition 1 shows us that G_f is a pullback metric. Particularly $\mathcal{Z}$ and J_f define the structure of the ambient observation space $\mathcal{X}$ (in our case, next state transitions).

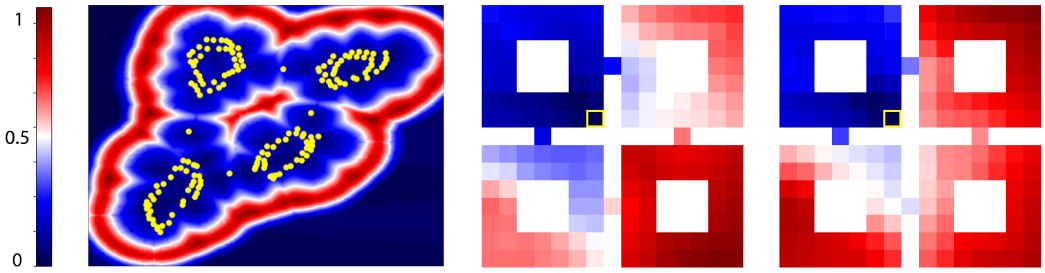
By choosing f to be the decoder function f_D, the metric G_{f_D} becomes stochastic, complicating analysis. Instead, as proposed and analyzed in Arvanitidis et al. [2018], we use the expected pullback metric $\mathbb{E}\left[G_{\mathcal{Z}}\right]$ as an approximation of the underlying stochastic metric. Similar to previous work on variational models, we use a normally distributed decoder to define the output. Using Proposition 1, we have the following result (see Appendix for proof).

Theorem 1. *[Arvanitidis et al. [2018]] Assume $f_D(\cdot|z) \sim \mathcal{N}(\mu(z), \sigma(z)I)$. Then*

$$\mathbb{E}_{f_D(\cdot|z)} \Big[G_{f_D}(z)\Big] = G_\mu(z) + G_\sigma(z), \tag{4}$$

where $G_\mu(z) = J_\mu^T(z) J_\mu(z)$ and $G_\sigma(z) = J_\sigma^T(z) J_\sigma(z)$.

Given an embedded latent space $\mathcal{Z}$, the expected metric in Equation (4) gives us a sense of the topology of the latent space manifold induced by f_D. The terms $G_\mu = J_\mu^T J_\mu$ and $G_\sigma = J_\sigma^T J_\sigma$ are in fact the induced pullback metrics of μ and σ, respectively.



(a) $\sqrt{\det(G_{f_D})}$ (b) Latent Geodesic Distance (c) Latent Euclidean Distance

Figure 2: **(a)** The latent space (yellow markers) of the grid world environment and the geometric volume measure of the decoder-induced metric (background). **(b, c)** The geodesic distance of the decoder-induced submanifold and the Euclidean distance of latent states, as viewed in ambient space. All distances are calculated w.r.t. the yellow marked state. **Note**: colors in (a), which measure magnitude, are unrelated to colors in (b,c), which measure distance to the yellow marked state.

4.3 Capturing Epistemic and Aleatoric Uncertainty

The previously proposed encoder-decoder model induces a metric which captures the structure of the learned dynamics. However, the decoder variance, $\sigma(z)$, does not differentiate between aleatoric uncertainty (environment dynamics) and epistemic uncertainty (missing data).

We propose two methods to enrich the metric in Equation (4) in order to achieve a better estimate of uncertainty. First, by using an ensemble of M decoder functions $\{f_{D,i}\}_{i=1}^{M}$ trained using standard bootstrap techniques [Efron, 1982], we capture the traditional epistemic uncertainty of the decoder parameters. Second, to correctly distinguish epistemic and aleatoric uncertainty, we add a latent forward function to our previously proposed variational model. Specifically, our latent model consists of an encoder $E : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{B}(\mathcal{Z})$, forward model $f_F : \mathcal{Z} \mapsto \mathcal{B}(\mathcal{X})$ and decoder functions $f_{D,i} : \mathcal{X} \mapsto \mathcal{B}(\mathcal{S})$ such that $P(\cdot|s,a) = f_{D,i}(\cdot|x)$, and $x \sim f_F(\cdot|E(s,a))$. This structure enables us to capture the aleatoric uncertainty under the forward transition model f_F, and the epistemic uncertainty using the decoders $f_{D,i}$. That is, for $f_F(\cdot|z) \sim \mathcal{N}(\mu_F(z), \sigma_F(z)I)$, the variance model $\sigma_F(z)$ captures the stochasticity in model dynamics. This decomposition is also helpful whenever one wants to train an agent in latent space (e.g., for planning Schrittwieser et al. [2020])

Next, we turn to analyze the pullback metric induced by the proposed forward transition model. As both f_F and $\{f_{D,i}\}_{i=1}^{m}$ are stochastic (capturing epistemic and aleatoric uncertainty), the result of Theorem 1 cannot be directly applied to their composition. The following proposition provides an analytical expression for the expected pullback metric of a sampled next state and a uniformly sampled decoder (the proof is given in the appendix).

Theorem 2. *Assume* $f_F(\cdot|z) \sim \mathcal{N}(\mu_F(z), \sigma_F(z)I)$, $f_{D,i}(\cdot|x) \sim \mathcal{N}(\mu_D^i(x), \sigma_D^i(x)I)$, $U \sim$ *Unif*$\{1, \dots, M\}$. *Then, the expected pullback metric of the composite function* $(f_{D,U} \circ f_F)$ *is given by*

$$\mathbb{E}_{P(f_{D,U}\circ f_F)}\Big[G_{f_{D,U}\circ f_F}(z)\Big] = J_{\mu_F}^T(z)\overline{G}_{f_D}(z)J_{\mu_F}(z) + J_{\sigma_F}^T(z)\mathit{diag}\big(\overline{G}_{f_D}(z)\big)J_{\sigma_F}(z),$$

where $\overline{G}_{f_D}(z) = \frac{1}{M}\sum_{i=1}^{M}\mathbb{E}_{x\sim F(\cdot|z)}\Big[J_{\mu_D^i}^T(x)J_{\mu_D^i}(x) + J_{\sigma_D^i}^T(x)J_{\sigma_D^i}(x)\Big]$.

Unlike the metric in Equation (4), the composite metric distorts the decoder metric with Jacobian matrices of the forward model statistics. It takes into account both the aleatoric and epistemic uncertainty through the forward model as well as ensemble of decoders. As a special case we note the metric for the case of deterministic model dynamics.

Corollary 1. *Assume deterministic model dynamics, i.e.,* $x = f_F(z)$, *and without loss of generality assume* $f_F \equiv I$. *Then, the expected pullback metric of Theorem 2 is given by* $\mathbb{E}\Big[G_{f_{D,U}\circ f_F}(z)\Big] = \frac{1}{M}\sum_{i=1}^{M} J_{\mu_D^i}^T(z)J_{\mu_D^i}(z) + J_{\sigma_D^i}^T(z)J_{\sigma_D^i}(z)$.

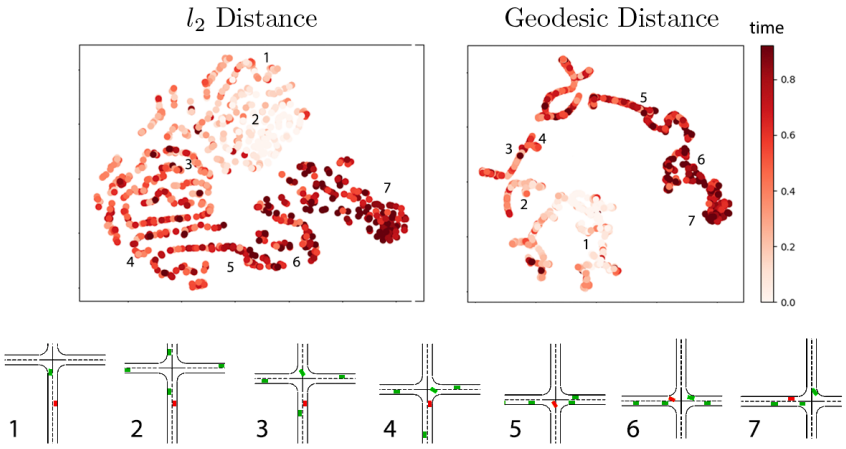


Figure 3: Plots show t-SNE embeddings generated in the intersection environment. Left plot depicts embeddings using Euclidean distances. Right plot depicts embeddings using geodesics distances which induce curves of minimum energy in ambient space. Colors correspond to the time in a trajectory (normalized w.r.t. longest trajectory). Numbering show mappings of a specific trajectory's states onto the embedded space as the controlled red car takes a left turn at an intersection (trajectory visualization shown under plots).

4.4 GELATO: Geometrically Enriched LATent model for Offline reinforcement learning

Having defined our metric, we are now ready to leverage it in a model based offline RL framework. Specifically, provided a dataset $\mathcal{D}_n = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^n$ we train the variational latent forward model described in the previous section.

Algorithm 1 presents GELATO, our proposed approach. In GELATO, we estimate model error by measuring the distance of a new sample to the data manifold. We construct the reward-penalized MDP for which the error acts as a pessimistic regularizer. Finally, we train an RL agent over the pessimistic MDP with transition $\hat{P}(\cdot|s,a)$ and reward $r(s,a) - \lambda U(s,a)$. By achieving an improved estimate for model error the model-based pessimistic approach can significantly improve performance, as shown in the following section.

5 Experiments

This section is dedicated to quantitatively and qualitatively understand the benefits of our proposed metric and method. We validate two principal claims: **(1) Our metric captures inherent characteristics of model dynamics.** We demonstrate this by visualizing state-action geodesics of a toy grid world problem and a multi-agent autonomous driving task. We show that curves of minimum energy in ambient space indeed capture intrinsic properties of the problem. **(2) Our metric provides an improved OOD uncertainty estimate for offline RL.** We compare the traditionally used bootstrap ensemble method to our approach, which leverages our pullback metric in a nonparametric nearest neighbors approach. We also compare our method to simple use of Euclidean distances in latent space. We run extensive experiments on continuous control and autonomous driving benchmarks. We show that our metric achieves significantly improved performance in tasks for which geodesics are non-euclidean.

5.1 Metric Visualization

Four Rooms. To better understand the inherent structure of our metric, we constructed a grid-world environment for visualizing our proposed latent representation and metric. The 15×15 environment, as depicted in Figure 2, consists of four rooms, with impassable obstacles in their centers. The agent, residing at some position $(x, y) \in [-1, 1]^2$ in the environment can take one of four actions: up, down, left, or right – moving the agent $1, 2$ or 3 steps (uniformly distributed) in that direction. We collected a dataset of 10000 samples, taking random actions at random initializations of the environment. The ambient state space was represented by the position of the agent – a vector of dimension 2, normalized to values in $[-1, 1]$. Finally, we trained a variational latent model with latent dimension $d_{\mathcal{Z}} = 2$.

Table 1: Performance of GELATO and its variants in comparison to contemporary model-based methods on D4RL datasets. Scores correspond to the return, averaged over 5 seeds (standard deviation removed due to space constraints and is given in the appendix). Results for MOPO, MBPO, SAC, and imitation are taken from Yu et al. [2020]. Mean score of dataset added for reference. Bold scores show an improved score w.r.t other methods.

	Hopper			**Walker2d**			**Halfcheetah**		
Method	**Rand**	**Med**	**Med-Expert**	**Rand**	**Med**	**Med-Expert**	**Rand**	**Med**	**Med-Expert**
Data Score	299	1021	1849	1	498	1062	-303	3945	8059
GELATO (metric)	**685**	**1676**	574	**412**	**1269**	**1515**	2560	**5168**	**7989**
GELATO (ℓ_2)	544	1320	815	388	312	1255	512	4096	7304
MOPO (bootstrap)	**677**	1202	1063	**396**	518	1296	**4114**	4974	7594
MBPO	444	457	2105	251	370	222	3527	3228	907
SAC	664	325	1850	120	27	-2	3502	-839	-78
Imitation	615	1234	**3907**	47	193	329	-41	4201	4164

We used a standard encoder $z \sim \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$ and decoder $s' \sim \mathcal{N}(\mu_\phi(z), \sigma_\phi(z))$ represented by neural networks trained end-to-end using the evidence lower bound. We refer the reader to the appendix for an exhaustive description of the training procedure.

The latent space output of our model is depicted by yellow markers in Figure 2a. Indeed, the latent embedding consists of four distinctive clusters, structured in a similar manner as our grid-world environment. Interestingly, the distortion of the latent space accurately depicts an intuitive notion of distance between states. As such, rooms are distinctively separated, with fair distance between each cluster. States of pathways between rooms clearly separate the room clusters, forming a topology with four discernible bottlenecks.

In addition to the latent embedding, Figure 2a depicts the geometric volume measure $\sqrt{\det(G_{f_D})}$ of the trained pullback metric induced by f_D. This quantity demonstrates the effective geodesic distances between states in the decoder-induced submanifold. Indeed geodesics between data points to points outside of the data manifold (i.e., outside of the red region), attain high values as integrals over areas of high magnitude. In contrast, geodesics near the data manifold would low values.

We visualize the decoder-induced geodesic distance and compare it to the latent Euclidean distance in Figures 2b and 2c, respectively. The plots depict the normalized distances of all states to the state marked by a yellow square. Evidently, the geodesic distance captures a better notion of distance in the said environment, correctly exposing the “land distance" in ambient space. As expected, states residing in the bottom-right room are farthest from the source state, as reaching them comprises of passing through at least two bottleneck states. In contrast, the latent Euclidean distance does not properly capture these geodesics, exhibiting a similar distribution of distances in other rooms. Nevertheless, both geodesic and Euclidean distances reveal the intrinsic topological structure of the environment, that of which is not captured by the extrinsic coordinates $(x, y) \in [-1, 1]^2$. Particularly, the state coordinates (x, y) would wrongly assign short distances to states across impassible walls or obstacles, i.e., measuring the “air distance".

Intersection. We visualized our metric in the intersection environment proposed in Leurent [2018]. Figure 3 compares the Euclidean and geodesic distances of a partially trained agent. Unlike the previous toy example, to visualize the inherent manifolds we used t-SNE [Van der Maaten and Hinton, 2008] projections computed with Euclidean distance and compared them to the projection computed with geodesic distance, i.e., curves of minimum energy in ambient space (Definition 2). Indeed, the geodesics captured the inherent structure of the environment, whereas Euclidean distances only managed to capture general clusters. These suggest that Euclidean distance might not be representative for measuring distance in latent space, as will also become evident by our experiments in the following subsections.

5.2 Datasets and Implementation Details

We used D4RL [Fu et al., 2020] and the autonomous vehicle environments highway-env [Leurent, 2018] as benchmarks for all of our experiments. We tested GELATO on three Mujoco [Todorov et al., 2012] environments (Hopper, Walker2d, Halfcheetah) on datasets generated by a single policy and a mixture of two policies. Specifically, we used datasets generated by a random agent (1M samples), a partially trained agent, i.e, medium agent (1M samples), and a mixture of partially trained and expert

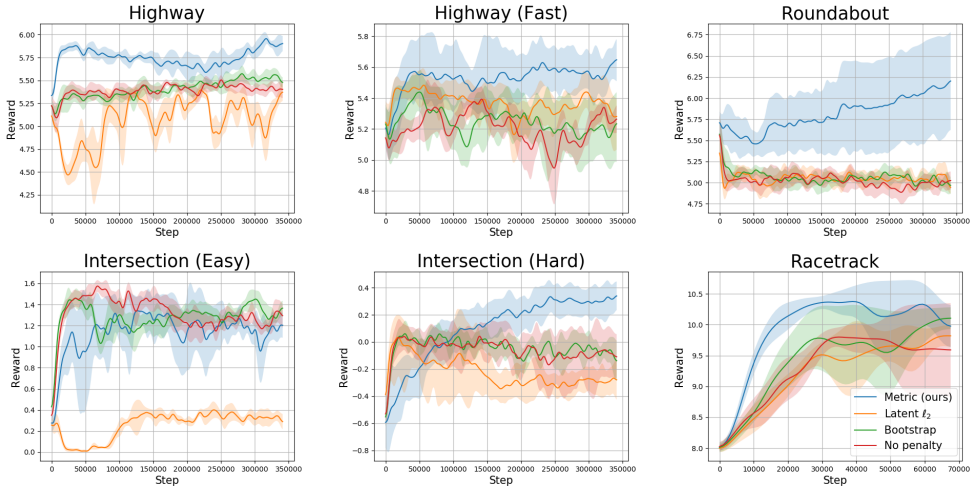


Figure 4: Performance of GELATO with different uncertainty estimation methods on the highway-env benchmarks. Results suggest our induced semiparametric distance is an effective penalty for offline RL.

agents (2M samples). For autonomous driving, we tested GELATO on four environments (Highway, Roundabout, Intersection, Racetrack), on datasets containing five episodes generated by a partially trained agent. We also tested a faster ($\times 15$ speedup) variant of the Highway environment, as well as a harder instantiation of the Intersection environment in which the number of cars was tripled (further details can be found in the appendix).

We trained our variational latent model in two phases. First, the model was fully trained using a calibrated Gaussian decoder [Rybkin et al., 2020]. Specifically, a maximum-likelihood estimate of the variance was used $\sigma^* = \text{MSE}(\mu, \hat{\mu}) \in \arg\max_\sigma \mathcal{N}(\hat{\mu}|\mu, \sigma^2 I)$. Then, in the second stage we fit the variance decoder networks. Hyperparameters for training are found in the appendix.

In order to practically estimate the geodesic distance in Algorithm 1, we defined a parametric curve in latent space and used gradient descent to minimize the curve's energy. The resulting curve and pullback metric were then used to calculate the geodesic distance by a numerical estimate of the curve length (See Appendix for an exhaustive overview of the estimation method).

We used FAISS [Johnson et al., 2019] for efficient GPU-based k-nearest neighbors calculation. We set $k = 5$ neighbors for the penalized reward (Equation (3)). Finally, we used a variant of Soft Learning, as proposed by Yu et al. [2020] as our RL algorithm for the continuous control benchmarks, and PPO [Schulman et al., 2017] for the autonomous driving tasks. All agents were trained for 1M steps (for continuous control benchmarks) and 350K steps (for the driving benchmarks), using a single GPU (RTX 2080), and averaged over 5 seeds (see Appendix for more details).

5.3 Results

D4RL. Results for the continuous control domains are shown in Table 1. We performed experiments to analyze GELATO on various continuous control datasets. We compared GELATO to contemporary model-based offline RL approaches; namely, MOPO [Yu et al., 2020] and MBPO [Janner et al., 2019], as well as the standard baselines of SAC [Haarnoja et al., 2018] and imitation (behavioral cloning, Bain and Sammut [1995], Ross et al. [2011]).

We found performance increase on most domains, and most significantly in the medium domain, i.e., partially trained agent. Since the medium dataset contained average behavior, combining the nonparametric nearest-neighbor uncertainty method with the bootstrap of decoders benefited the agent's overall performance. In addition, GELATO with the latent ℓ_2 distance metric performed well on many of the benchmarks. We conjecture this is due to the inherent Euclidean nature of the continuous control benchmarks. Unlike the embedding for the autonomous driving benchmarks (Figure 3), we found the D4RL data to project similarly when ℓ_2 and geodesic distances were used (we provide plots of these embeddings in the Appendix).

Highway-Env. Figure 4 shows results for the autonomous driving benchmarks in highway-env. In contrast to the continuous control benchmarks, we found a significant improvement of our metric on the autonomous driving benchmarks compared to standard uncertainty estimation methods as well as the latent Euclidean distance. We credit this improvement to the non-euclidean nature of the environments, as previously described in Figure 3. While Euclidean distances were useful in the Mujoco environments, they performed distinctly worse in the autonomous driving environments.

Our results emphasize the importance of OOD uncertainty estimations methods in reinforcement learning on various types of datasets. While robotic control tasks provided useful insights, they did not capture the non-euclidean nature inherent in alternative tasks, such as autonomous driving.

6 Related Work

Offline Reinforcement Learning. The field of offline RL has recently received much attention as several algorithmic approaches were able to surpass standard off-policy algorithms. Value-based online algorithms do not perform well under highly off-policy batch data [Fujimoto et al., 2019, Kumar et al., 2019, Fu et al., 2019, Fedus et al., 2020, Agarwal et al., 2020], much due to issues with bootstrapping from out-of-distribution (OOD) samples. These issues become more prominent in the offline setting, as new samples cannot be acquired.

Several works on offline RL have shown improved performance on standard continuous control benchmarks [Laroche et al., 2019, Kumar et al., 2019, Fujimoto et al., 2019, Chen et al., 2020b, Swazinna et al., 2020, Kidambi et al., 2020, Yu et al., 2020]. This work focused on model-based approaches [Yu et al., 2020, Kidambi et al., 2020], in which the agent is incentivized to remain close to areas of low uncertainty. Our work focused on controlling uncertainty estimation in high dimensional environments. Our methodology utilized recent success on the geometry of deep generative models [Arvanitidis et al., 2018, 2020], proposing an alternative approach to uncertainty estimation.

Representation Learning. Representation learning seeks to find an appropriate representation of data for performing a machine-learning task [Goodfellow et al., 2016]. Variational Auto Encoders [Kingma and Welling, 2013, Rezende et al., 2014] have been a popular representation learning technique, particularly in unsupervised learning regimes [Chen et al., 2016, Van Den Oord et al., 2017, Hsu et al., 2017, Serban et al., 2017, Engel et al., 2017, Bojanowski et al., 2018, Ding et al., 2020], though also in supervised learning and reinforcement learning [Hausman et al., 2018, Li et al., 2019, Petangoda et al., 2019, Hafner et al., 2019]. Particularly, variational models have been shown able to derive successful behaviors in high dimensional benchmarks [Hafner et al., 2020].

Various representation techniques in reinforcement learning have also proposed to disentangle representation of both states [Engel and Mannor, 2001, Littman and Sutton, 2002, Stooke et al., 2020, Zhu et al., 2020], and actions [Tennenholtz and Mannor, 2019, Chandak et al., 2019]. These allow for the abstraction of states and actions to significantly decrease computation requirements, by e.g., decreasing the effective dimensionality of the action space [Tennenholtz and Mannor, 2019]. Unlike previous work, GELATO is focused on a semiparametric approach for uncertainty estimation, enhancing offline reinforcement learning performance.

7 Discussion and Future Work

This work presented a metric for model dynamics and its application to offline reinforcement learning. While our metric showed supportive evidence of improvement in model-based offline RL we note that it was significantly slower – comparably, 5 times slower than using the decoder's variance for uncertainty estimation. The apparent slowdown in performance was mostly due to computation of the geodesic distance. Improvement in this area may utilize techniques for efficient geodesic estimation.

We conclude by noting possible future applications of our work. In Section 5.1 we demonstrated the inherent geometry our model had captured, its corresponding metric, and geodesics. Still, in this work we focused specifically on metrics related to the decoded state. In fact, a similar derivation to Theorem 2 could be applied to other modeled statistics, e.g., Q-values, rewards, future actions, and more. Each distinct statistic would induce its own unique metric w.r.t. its respective probability measure. Particularly, this concept may benefit a vast array of applications in continuous or large state and action spaces.

References

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: On the curvature of deep generative models. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

Georgios Arvanitidis, Søren Hauberg, and Bernhard Schölkopf. Geometrically enriched latent spaces. *arXiv preprint arXiv:2008.00565*, 2020.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. Optimizing the latent space of generative networks. In *International Conference on Machine Learning*, pages 600–609, 2018.

Jacob Buckman, Carles Gelada, and Marc G Bellemare. The importance of pessimism in fixed-dataset policy optimization. *arXiv preprint arXiv:2009.06799*, 2020.

Manfredo Perdigao do Carmo. *Riemannian geometry*. Birkhäuser, 1992.

Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip Thomas. Learning action representations for reinforcement learning. In *International Conference on Machine Learning*, pages 941–950, 2019.

Nutan Chen, Alexej Klushyn, Richard Kurle, Xueyan Jiang, Justin Bayer, and Patrick Smagt. Metrics for deep generative models. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1550. PMLR, 2018.

Nutan Chen, Francesco Ferroni, Alexej Klushyn, Alexandros Paraschos, Justin Bayer, and Patrick van der Smagt. Fast approximate geodesics for deep generative models. In *International Conference on Artificial Neural Networks*, pages 554–566. Springer, 2019.

Nutan Chen, Alexej Klushyn, Francesco Ferroni, Justin Bayer, and Patrick van der Smagt. Learning flat latent manifolds with vaes. 2020a.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, and Keith Ross. Bail: Best-action imitation learning for batch deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33, 2020b.

Zheng Ding, Yifan Xu, Weijian Xu, Gaurav Parmar, Yang Yang, Max Welling, and Zhuowen Tu. Guided variational autoencoder for disentanglement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7920–7929, 2020.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Bradley Efron. *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982.

Jesse Engel, Matthew Hoffman, and Adam Roberts. Latent constraints: Learning to generate conditionally from unconditional generative models. *arXiv preprint arXiv:1711.05772*, 2017.

Yaakov Engel and Shie Mannor. Learning embedded maps of markov processes. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 138–145, 2001.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

Aboalhasan Fathabadi, Seyed Morteza Seyedian, and Arash Malekian. Comparison of bayesian, k-nearest neighbor and gaussian process regression methods for quantifying uncertainty of suspended sediment concentration prediction. *Science of The Total Environment*, page 151760, 2021.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pages 3061–3071. PMLR, 2020.

Raphael Fonteneau, Susan A Murphy, Louis Wehenkel, and Damien Ernst. Batch mode reinforcement learning based on the synthesis of artificial trajectories. *Annals of operations research*, 208(1): 383–416, 2013.

Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. Diagnosing bottlenecks in deep q-learning algorithms. In *International Conference on Machine Learning*, pages 2021–2030, 2019.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in neural information processing systems*, pages 1878–1889, 2017.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*, 2019.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? *arXiv preprint arXiv:2012.15085*, 2020.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

Dimitris Kalatzis, David Eklund, Georgios Arvanitidis, and Søren Hauberg. Variational autoencoders with riemannian brownian motion priors. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*. PMLR, 2020.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Alexej Klushyn, Nutan Chen, Richard Kurle, Botond Cseke, and Patrick van der Smagt. Learning hierarchical priors in vaes. In *Advances in Neural Information Processing Systems*, pages 2870–2879, 2019.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11784–11794, 2019.

Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning*, pages 3652–3661. PMLR, 2019.

Edouard Leurent. An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env`, 2018.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Minne Li, Lisheng Wu, WANG Jun, and Haitham Bou Ammar. Multi-view reinforcement learning. In *Advances in neural information processing systems*, pages 1420–1431, 2019.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062. PMLR, 2018.

Michael L Littman and Richard S Sutton. Predictive representations of state. In *Advances in neural information processing systems*, pages 1555–1561, 2002.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Janith C Petangoda, Sergio Pascual-Diaz, Vincent Adam, Peter Vrancx, and Jordi Grau-Moya. Disentangled skill embeddings for reinforcement learning. *arXiv preprint arXiv:1906.09223*, 2019.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

Oleh Rybkin, Kostas Daniilidis, and Sergey Levine. Simple and effective vae training with calibrated decoders. *arXiv preprint arXiv:2006.13202*, 2020.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. *arXiv preprint arXiv:2009.08319*, 2020.

Phillip Swazinna, Steffen Udluft, and Thomas Runkler. Overcoming model bias for robust offline deep reinforcement learning. *arXiv preprint arXiv:2008.05533*, 2020.

Guy Tennenholtz and Shie Mannor. The natural language of actions. In *International Conference on Machine Learning*, pages 6196–6205, 2019.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Joe Luis Villa Medina et al. *Reliability of classification and prediction in k-nearest neighbours*. PhD thesis, Universitat Rovira i Virgili, 2013.

Ruosong Wang, Dean P Foster, and Sham M Kakade. What are the statistical limits of offline rl with linear function approximation? *arXiv preprint arXiv:2010.11895*, 2020.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

Andrea Zanette. Exponential lower bounds for batch reinforcement learning: Batch rl can be exponentially harder than online rl. *arXiv preprint arXiv:2012.08005*, 2020.

Jinhua Zhu, Yingce Xia, Lijun Wu, Jiajun Deng, Wengang Zhou, Tao Qin, and Houqiang Li. Masked contrastive representation learning for reinforcement learning. *arXiv preprint arXiv:2010.07470*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Table 2: Results of GELATO as presented in Table 1 with added std for each run, averaged over 5 seeds.

	Hopper			Walker2d			Halfcheetah		
Method	**Rand**	**Med**	**Med-Expert**	**Rand**	**Med**	**Med-Expert**	**Rand**	**Med**	**Med-Expert**
Data Score	299 ± 200	1021 ± 314	1849 ± 1560	1 ± 6	498 ± 807	1062 ± 1576	−303 ± 79	3945 ± 494	8059 ± 4204
GELATO (metric)	**685 ± 15**	**1676 ± 223**	574 ± 16	**412 ± 85**	**1269 ± 549**	**1515 ± 379**	2560 ± 240	**5168 ± 849**	**7989 ± 2790**
GELATO (ℓ_2)	544 ± 29	1320 ± 423	815 ± 153	388 ± 49	312 ± 189	1255 ± 310	512 ± 71	4096 ± 585	7304 ± 3780
MOPO	**677 ± 13**	1202 ± 400	1063 ± 193	**396 ± 76**	518 ± 560	1296 ± 374	**4114 ± 312**	**4974 ± 200**	**7594 ± 4741**
MBPO	444 ± 193	457 ± 106	2105 ± 1113	251 ± 235	370 ± 221	222 ± 99	3527 ± 487	3228 ± 2832	907 ± 1185
SAC	664	325	1850	120	27	-2	3502	-839	-78
Imitation	615	1234	**3907**	47	193	329	-41	4201	4164

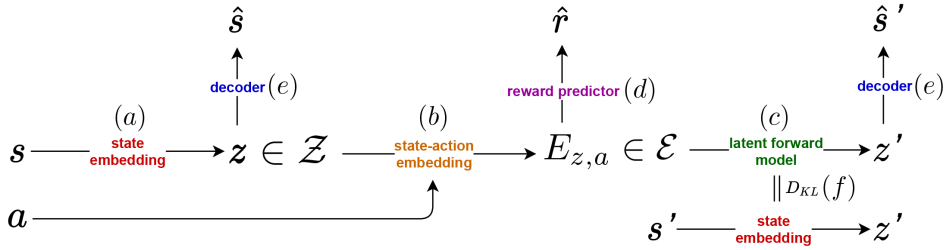


Figure 5: A graphical representation of our latent variable model. **(a)** The state s is embedded via the state embedding function (i.e., approximate posterior) $z \sim q(\cdot|s)$. **(b)** The action and embedded state pass through an invertible embedding function E to produce the state-action embedding $E_{z,a}$. **(c,d)** The state-action embedding is passed through a reward predictor and latent forward model, $\hat{r} \sim P(\cdot|E_{z,a})$ and $z' \sim P(\cdot|z,a)$, respectively. **(e)** The next latent state z' is decoded back to observation space to generate $\hat{s}' \sim P(\cdot|z')$. **(f)** Finally, during training, the target state s' is embedded and compared to z' (by the KL-divergence term in Equation (6)), preserving the consistency of the latent space $\mathcal{Z}$.

Appendix

8 Variational Latent Model

We begin by describing our variational forward model. The model, based on an encoder, latent forward function, and decoder framework assumes the underlying dynamics and reward are governed by a state-embedded latent space $\mathcal{Z} \subseteq \mathbb{R}^{d_\mathcal{Z}}$. The probability of a trajectory $\tau = (s_0, a_0, r_0, \ldots, s_h, a_h, r_h)$ is given by

$$P(\tau) = \int_{z_0,\ldots,z_h} P(z_0) \prod_{i=0}^{h} P(s_i|z_i)\pi(a_i|s_i)P(r_i|E_{z_i,a_i}) \prod_{j=1}^{h} P(z_j|E_{z_{j-1},a_{j-1}}) dz_0 \ldots dz_h, \quad (5)$$

where $E : \mathcal{Z} \times \mathcal{A} \mapsto \mathcal{E} \subseteq \mathbb{R}^{d_\mathcal{E}}$ is a deterministic, invertible embedding function which maps pairs (z,a) to a state-action-embedded latent space $\mathcal{E}$. $E_{z,a}$ is thus a sufficient statistic of (z,a). The proposed graphical model is depicted in Figure 5. We note that an extension to the partially observable setting replaces s_t with $h_t = (s_0, a_0, \ldots, s_t)$, a sufficient statistic of the unknown state.

Maximizing the log-likelihood $\log P(\tau)$ is hard due to intractability of the integral in Equation (5). We therefore introduce the approximate posterior $q(z|s)$ and maximize the evidence lower bound.

To clear notations we define $E_{z_{-1},a_{-1}} = 0$, so that we can rewrite the above expression as

$$P(s_0, a_0, r_0, \ldots, s_h, a_h, r_h) = \int_{z_0,\ldots,z_h} \prod_{i=0}^{h} P(s_i|z_i)\pi(a_i|s_i)P(r_i|E_{z_i,a_i})P(z_j|E_{z_{-1},a_{-1}})$$

Introducing $q(z_i|s_i)$ we can write

$$\begin{aligned}
&\log \int_{z_0,\ldots,z_h} \prod_{i=0}^{h} \frac{q(z_i|s_i)}{q(z_i|s_i)} \prod_{i=0}^{h} P(s_i|z_i)\pi(a_i|s_i)P(r_i|E_{z_i,a_i})P(z_j|E_{z_{-1},a_{-1}}) \\
&\geq \int_{z_0,\ldots,z_h} \prod_{i=0}^{h} q(z_i|s_i) \left(\sum_{i=0}^{h} \log \left(\frac{P(s_i|z_i)\pi(a_i|s_i)P(r_i|E_{z_i,a_i})P(z_j|E_{z_{-1},a_{-1}})}{q(z_i|s_i)} \right) \right) \\
&= \sum_{i=0}^{H} \mathbb{E}_{q(z_i|s_i)} \Big[\log(P(s_i \mid z_i)\pi(a_i \mid s_i)P(r_i|E_{z_i,a_i})) \Big] \\
&- \sum_{i=0}^{H-1} \mathbb{E}_{q(z_i|s_i)} \Big[D_{KL}(q(z_{i+1} \mid s_{i+1}) \mid\mid P(z_{i+1}|E_{z_i,a_i})) \Big] \\
&- D_{KL}(q(z_0 \mid s_0) \mid\mid P(z_0)).
\end{aligned}$$

Hence,

$$\begin{aligned}
&\sum_{i=0}^{h} \mathbb{E}_{z_i \sim q(z_i|s_i)} \Big[\log(P(s_i \mid z_i)\pi(a_i \mid s_i)P(r_i|E_{z_i,a_i})) \Big] \\
&- \sum_{i=0}^{h-1} \mathbb{E}_{z_i \sim q(z_i|s_i)} \Big[D_{KL}(q(z_{i+1} \mid s_{i+1}) \mid\mid P(z_{i+1}|E_{z_i,a_i})) \Big] \\
&- D_{KL}(q(z_0 \mid s_0) \mid\mid P(z_0)).
\end{aligned} \tag{6}$$

The distribution parameters of the approximate posterior $q(z|s)$, the likelihoods $P(s|z), \pi(a|s), P(r|E_{z,a})$, and the latent forward model $P(z'|E_{z,a})$ are represented by neural networks. The invertible embedding function E is represented by an invertible neural network, e.g., affine coupling, commonly used for normalizing flows [Dinh et al., 2014]. Though various latent distributions have been proposed [Klushyn et al., 2019, Kalatzis et al., 2020], we found Gaussian parametric distributions to suffice for all of our model's functions. Particularly, we used two outputs for every distribution, representing the expectation μ and variance σ. All networks were trained end-to-end to maximize the evidence lower bound in Equation (6).

9 Specific Implementation Details

D4RL. As a preprocessing step rewards were normalized to values between $[-1, 1]$. We trained our variational model with latent dimensions $\dim(\mathcal{Z}) = 32$ and $\dim(\mathcal{E}) = \dim(\mathcal{Z}) + \dim(\mathcal{A})$. All domains were trained with the same hyperparameters. Specifically, we used a 2-layer Multi Layer Perceptron (MLP) to encode $\mathcal{Z}$, after which a 2-layer Affine Coupling (AC) [Dinh et al., 2014] was used to encode $\mathcal{E}$. We also used a 2-layer MLP for the forward, reward, and decoder models. All layers contained 256 hidden layers.

The latent model was trained in two separate phases for 100k and 50k steps each by stochastic gradient descent and the ADAM optimizer [Kingma and Ba, 2014]. First, the model was fully trained using a calibrated Gaussian decoder [Rybkin et al., 2020]. Specifically, a maximum-likelihood estimate of the variance was used $\sigma^* = \text{MSE}(\mu, \hat{\mu}) \in \arg\max_\sigma \mathcal{N}(\hat{\mu}|\mu, \sigma^2 I)$. Finally, in the second stage we fit the variance decoder network. We found this process of to greatly improve convergence speed and accuracy, and mitigate posterior collapse. We used a minimum variance of 0.01 for all of our stochastic models.

To further stabilize training we used a momentum encoder. Specifically we updated a target encoder as a slowly moving average of the weights from the learned encoder as

$$\bar{\theta} \leftarrow (1 - \tau)\bar{\theta} + \tau\theta$$

Hyperparameters for our variational model are summarized in Table 3. The latent architecture is visualized in Figure 6.

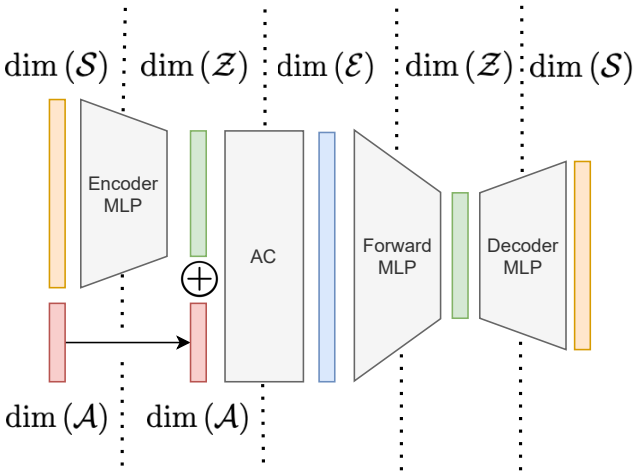


Figure 6: Latent model architecture (does not depict reward MLP).

Parameter	Value	Parameter	Value
$\dim(\mathcal{Z})$	32	LEARNING RATE	10^{-3}
$\dim(\mathcal{E})$	$32 + \dim(\mathcal{A})$	BATCH SIZE	128 (D4RL), 32 (HIGHWAY-ENV)
ENCODER MLP HIDDEN	$256, 256$	TARGET UPDATE τ	0.01
FORWARD MLP HIDDEN	$256, 256$	TARGET UPDATE INTERVAL	1
DECODER MLP HIDDEN	$256, 256$	PHASE 1 UPDATES	100000 (D4RL), 10000 (HIGHWAY-ENV)
REWARD MLP HIDDEN	$256, 256$	PHASE 2 UPDATES	50000 (D4RL), 2000 (HIGHWAY-ENV)

Table 3: Hyper parameters for variational model

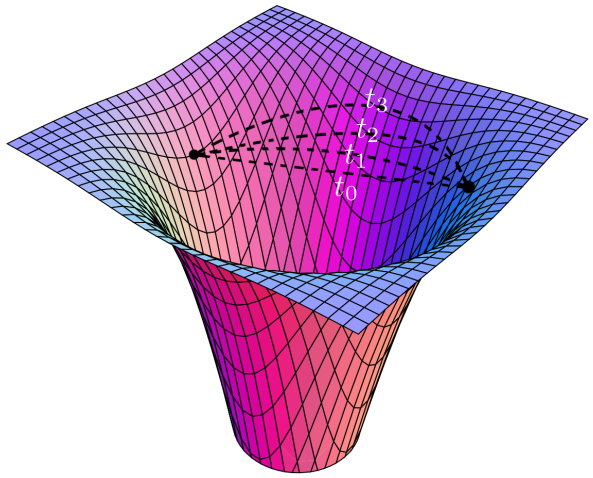


Figure 7: Illustration of geodesic curve optimization in Algorithm 2.

9.1 Geodesic Distance Estimation

In order to practically estimate the geodesic distance between two points $e_1, e_2 \in \mathcal{E}$ we defined a parametric curve in latent space and used gradient descent to minimize the curve's energy [3]. The resulting curve and pullback metric were then used to calculate the geodesic distance by a numerical estimate of the curve length.

[3] Other methods for computing the geodesic distance include solving a system of ODEs [Arvanitidis et al., 2018], using graph based geodesics [Chen et al., 2019], or using neural networks [Chen et al., 2018].

Algorithm 2 Geodesic Distance Estimation

Input: forward latent F, decoder f_D, learning rate λ, number of iterations T, grid size n, eval points e_0, e_1
Initialize: parametric curve $\gamma_\theta : \gamma_\theta(0) = e_0, \gamma_\theta(1) = e_1$
for $t = 1$ **to** T **do**
  $L_\mu(\theta) \leftarrow \sum_{i=1}^{n} \mu_D(\mu_F(\gamma_\theta(\frac{i}{n}))) - \mu_D(\mu_F(\gamma_\theta(\frac{i-1}{n})))$
  $L_\sigma(\theta) \leftarrow \sum_{i=1}^{n} \sigma_D(\mu_F(\gamma_\theta(\frac{i}{n}))) - \sigma_D(\mu_F(\gamma_\theta(\frac{i-1}{n})))$
  $L(\theta) \leftarrow L_\mu(\theta) + L_\sigma(\theta)$
  $\theta \leftarrow \theta - \lambda \nabla_\theta L(\theta)$
end for
$G_{D \circ F} = J_{\mu_F}^T \overline{G}_D J_{\mu_F} + J_{\sigma_F}^T \text{diag}(\overline{G}_D) J_{\sigma_F}$
$\forall i, \Delta_i \leftarrow \gamma_\theta(\frac{i}{n}) - \gamma_\theta(\frac{i-1}{n})$
$d(e_0, e_1) \leftarrow \sum_{i=1}^{n} \left(\frac{\partial \gamma_\theta}{\partial t} \big|_{\frac{i}{n}} \right)^T G_{D \circ F}(\gamma_\theta(\frac{i}{n})) \left(\frac{\partial \gamma_\theta}{\partial t} \big|_{\frac{i}{n}} \right) \Delta_i$
Return: $d(e_0, e_1)$

Pseudo code for Geodesic Distance Estimation is shown in Algorithm 2. Our curve was modeled as a cubic spline with 8 coefficients. We used SGD (momentum 0.99) to optimize the curve energy over 20 gradient iterations with a grid of 10 points and a learning rate of 10^{-3}. An illustration of the convergence of such a curve is illustrated in Figure 7

9.2 RL algorithm

D4RL. Our learning algorithm is based on the Soft Learning framework proposed in Algorithm 2 of Yu et al. [2020]. Pseudo code is shown in Algorithm 3. Specifically we used two replay buffers $\mathcal{R}_{\text{model}}, \mathcal{R}_{\text{data}}$, where $|R_{\text{model}}| = 50000$ and $\mathcal{R}_{\text{data}}$ contained the full offline dataset. In every epoch an initial state s_0 was sampled from the offline dataset and embedded using our latent model to generate $z_0 \in \mathcal{Z}$. During rollouts of π, embeddings $E_{z,a} \in \mathcal{E}$ were then generated from z and used to (1) sample next latent state z', (2) sample estimated rewards r, and (3) compute distances to $K = 20$ nearest neighbors in embedded the dataset.

Highway-Env. We used PPO [Schulman et al., 2017] implemented in RLlib [Liang et al., 2018] trained over the variational forward model. All environments (except Racetrack) were trained with horizon $h = 10$. The Racetrack environment was trained with horizon $h = 50$.

We used Algorithm 2 to compute the geodesic distances, and FAISS [Johnson et al., 2019] for efficient nearest neighbor computation on GPUs. To stabilize learning, we normalized the penalty $\frac{1}{K} \sum_{k=1}^{K} d_k$ according to the maximum penalty, ensuring penalty lies in $[0, 1]$ (recall that the latent reward predictor was trained over normalized rewards in $[-1, 1]$). For the non-skewed version of GELATO, we used $\lambda = 1$ as our reward penalty coefficient and $\lambda = 2$ for the skewed versions. We used rollout horizon of $h = 5$, and did not notice significant performance improvement for different values of h.

10 Visualization in Continuous Control Benchmarks

In Section 5.1 we showed visualizations of our metric and compared its geodesics to ℓ_2 distances in latent space. In Figure 8 we show similar visualization for the medium-agent Halfcheetah dataset in D4RL. Particually, it is evident that the latent state in this environment is similar under our proposed metric and the standard Euclidean distance, suggesting the problems' natural manifold is in fact flat [Chen et al., 2020a]. These results give evidence to performance results of GELATO in Table 1. Indeed, GELATO with ℓ_2 penalty achieved similar performance to other methods in the D4RL benchmarks. Nevertheless, as was shown in Figure 4, the Euclidean distance in latent space failed catastrophically in the autonomous driving environments, due to discontinuities in latent space (as depicted in Figure 3.

Algorithm 3 GELATO with Soft Learning

Input: Reward penalty coefficient λ, rollout horizon h, rollout batch size b, training epochs T, number of neighbors K.
Train variational latent forward model on dataset $\mathcal{D}_n$ by maximizing ELBO (Equation (6))
Construct embedded dataset $\mathcal{D}_{\text{embd}} = \{E_i\}_{i=1}^{n}$ using latent model to initialize KNN.
Initialize policy π and empty replay buffer $\mathcal{R}_{\text{model}} \leftarrow \emptyset$.
for epoch $= 1$ **to** T **do**
  for $i = 1$ **to** b (in parallel) **do**
    Sample state s_1 from $\mathcal{D}_n$ for the initialization of the rollout and embed using latent model to produce z_1.
    for $j = 1$ **to** h **do**
      Sample an action $a_j \sim \pi(\cdot|z_j)$.
      Embed $(z_j, a_j) \rightarrow E_{z_j,a_j}$ using latent model
      Sample z_{j+1} from latent forward model $F(E_{z_j,a_j})$.
      Sample r_j from latent reward model $R(E_{z_j,a_j})$.
      Use Algorithm 2 to compute K nearest neighbors $\left\{\text{NN}_{z_j,a_j}^{(k)}\right\}_{k=1}^{K}$ and their distances $\{d_k\}_{k=1}^{N}$ to E_{z_j,a_j}.
      Compute $\tilde{r}_j = r_j - \lambda\left(\frac{1}{K}\sum_{k=1}^{K} d_k\right)$
      Add sample $(z_j, a_j, \tilde{r}_j, z_{j+1})$ to $\mathcal{R}_{\text{model}}$.
    end for
  end for
  Drawing samples from $\mathcal{R}_{\text{data}} \cup \mathcal{R}_{\text{model}}$, use SAC to update π.
end for

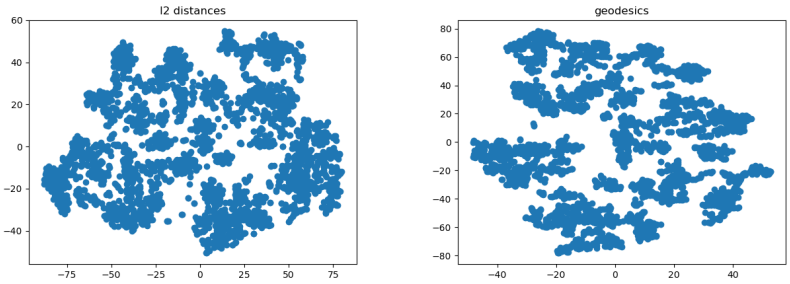


Figure 8: Plots show t-SNE embeddings generated using the HalfCheetah (medium) dataset. Left plot depicts embeddings using Euclidean distances. Right plot depicts embeddings using geodesics distances which induce curves of minimum energy in ambient space. Plots show first 3 episodes in the data.

11 Missing Proofs

11.1 Proof of Proposition 1

For any curve γ, we have that

$$\begin{aligned}\int_0^1 \left\| \frac{\partial f(\gamma(t))}{\partial t} \right\| dt &= \int_0^1 \left\| \frac{\partial f(\gamma(t))}{\partial \gamma(t)}^T \frac{\partial \gamma(t)}{\partial t} \right\| dt \\ &= \int_0^1 \left\| J_f(\gamma(t))^T \frac{\partial \gamma(t)}{\partial t} \right\| dt \\ &= \int_0^1 \sqrt{\left\langle \frac{\partial \gamma(t)}{\partial t}, J_f^T(\gamma(t)) J_f(\gamma(t)) \frac{\partial \gamma(t)}{\partial t} \right\rangle} dt \\ &= \int_0^1 \sqrt{\left\langle \frac{\partial \gamma(t)}{\partial t}, G_f(\gamma(t)) \frac{\partial \gamma(t)}{\partial t} \right\rangle} dt.\end{aligned}$$

This completes the proof.

11.2 Proof of Theorems 1 and 2

We begin by restating the theorems.

Theorem 1. *[Arvanitidis et al. [2018]] Assume $f_D(\cdot|z) \sim \mathcal{N}(\mu(z), \sigma(z)I)$. Then*

$$\mathbb{E}_{f_D(\cdot|z)}\Big[G_{f_D}(z)\Big] = G_\mu(z) + G_\sigma(z), \tag{4}$$

where $G_\mu(z) = J_\mu^T(z) J_\mu(z)$ and $G_\sigma(z) = J_\sigma^T(z) J_\sigma(z)$.

Theorem 2. *Assume $f_F(\cdot|z) \sim \mathcal{N}(\mu_F(z), \sigma_F(z)I)$, $f_{D,i}(\cdot|x) \sim \mathcal{N}(\mu_D^i(x), \sigma_D^i(x)I)$, $U \sim$ Unif$\{1, \dots, M\}$. Then, the expected pullback metric of the composite function $(f_{D,U} \circ f_F)$ is given by*

$$\mathbb{E}_{P(f_{D,U} \circ f_F)}\Big[G_{f_{D,U} \circ f_F}(z)\Big] = J_{\mu_F}^T(z) \overline{G}_{f_D}(z) J_{\mu_F}(z) + J_{\sigma_F}^T(z) \mathit{diag}\big(\overline{G}_{f_D}(z)\big) J_{\sigma_F}(z),$$

where $\overline{G}_{f_D}(z) = \frac{1}{M} \sum_{i=1}^M \mathbb{E}_{x \sim F(\cdot|z)}\Big[J_{\mu_D^i}^T(x) J_{\mu_D^i}(x) + J_{\sigma_D^i}^T(x) J_{\sigma_D^i}(x)\Big]$.

Notice that Theorem 1 is a special case of Theorem 2 with F being the trivial identity function. Additionally, a complete proof of Theorem 1 can be found in Arvanitidis et al. [2018]. We turn to prove Theorem 2.

We begin by proving the following auxilary lemma.

Lemma 1. *Let $\epsilon \sim \mathcal{N}(0, I_K)$, $f : \mathbb{R}^d \mapsto \mathbb{R}^K$, $A \in \mathbb{R}^{K \times K}$. Denote $S_i = \mathit{diag}\Big(\frac{\partial f^1}{\partial z_i}, \frac{\partial f^2}{\partial z_i}, \dots, \frac{\partial f^K}{\partial z_i}\Big)$ for $1 \leq i \leq d$ and*

$$B = \left[S_1\epsilon, S_2\epsilon, \dots, S_d\epsilon\right]_{K \times d}.$$

Then $\mathbb{E}\left[B^T A B\right] = J_f^T \mathit{diag}(A) J_f$.

Proof. We have that

$$\begin{aligned}\mathbb{E}\left[B^T A B\right] &= \mathbb{E}\left[\begin{bmatrix}\epsilon^T S_1^T \\ \epsilon^T S_2^T \\ \vdots \\ \epsilon^T S_d^T\end{bmatrix} A \left[S_1\epsilon, S_2\epsilon, \dots, S_d\epsilon\right]\right] \\ &= \begin{bmatrix}\mathbb{E}\left[\epsilon^T S_1^T A S_1 \epsilon\right], \mathbb{E}\left[\epsilon^T S_1^T A S_2 \epsilon\right], \dots, \mathbb{E}\left[\epsilon^T S_1^T A S_d \epsilon\right], \\ \mathbb{E}\left[\epsilon^T S_2^T A S_1 \epsilon\right], \mathbb{E}\left[\epsilon^T S_2^T A S_2 \epsilon\right], \dots, \mathbb{E}\left[\epsilon^T S_2^T A S_d \epsilon\right], \\ \ddots \\ \mathbb{E}\left[\epsilon^T S_d^T A S_1 \epsilon\right], \mathbb{E}\left[\epsilon^T S_d^T A S_2 \epsilon\right] \dots, \mathbb{E}\left[\epsilon^T S_d^T A S_d \epsilon\right]\end{bmatrix}.\end{aligned}$$

Finally notice that for any matrix M

$$\mathbb{E}\left[\epsilon^T M \epsilon\right] = \sum_{i=1}^{d}\sum_{j=1}^{d} M_{ij}\mathbb{E}\left[\epsilon_i \epsilon_j\right] = \sum_{i=1}^{d} M_{ii} = \text{trace}(M).$$

Then,

$$\mathbb{E}\left[B^T A B\right] = \begin{bmatrix} \text{trace}\big(S_1^T A S_1\big), \text{trace}\big(S_1^T A S_2\big), \ldots, \text{trace}\big(S_1^T A S_d\big) \\ \text{trace}\big(S_2^T A S_1\big), \text{trace}\big(S_2^T A S_2\big), \ldots, \text{trace}\big(S_2^T A S_d\big) \\ \ddots \\ \text{trace}\big(S_d^T A S_1\big), \text{trace}\big(S_d^T A S_2\big), \ldots, \text{trace}\big(S_d^T A S_d\big) \end{bmatrix}.$$

□

Next, note that

$$\text{trace}(S_i A S_j) = \sum_{k=1}^{K} \frac{\partial f^k}{\partial z_i}\frac{\partial f^k}{\partial z_j} A_{kk}.$$

Therefore,

$$\mathbb{E}\left[B^T A B\right] = J_f^T \text{diag}(A) J_f.$$

We are now ready to prove Theorem 2.

Proof of Theorem 2. We can write $z' = \mu_F(z) + \sigma_F(z) \odot \epsilon_F$ and $s' = \mu_D(z') + \sigma_D(z') \odot \epsilon_D$ where $\epsilon_F \sim \mathcal{N}(0, I_d), \epsilon_D \sim \mathcal{N}(0, I_K)$, $\mu_F : \mathbb{R}^d \mapsto \mathbb{R}^\ell$,$\mu_D : \mathbb{R}^\ell \mapsto \mathbb{R}^K$ and $\sigma_F : \mathbb{R}^d \mapsto \mathbb{R}^\ell, \sigma_D : \mathbb{R}^\ell \mapsto \mathbb{R}^K$.

Applying the chain rule we get

$$J_{D\circ F} = \frac{\partial (D \circ F)}{\partial z} = J_{\mu_D} J_{\mu_F} + J_{\mu_D} B_{\epsilon_F} + B_{\epsilon_D} J_{\mu_F} + B_{\epsilon_D} B_{\epsilon_F}$$

where

$$\begin{aligned} B_{\epsilon_F} &= \left(S_{F,1}\epsilon_F, S_{F,2}\epsilon_F, \ldots, S_{F,d}\epsilon_F\right)_{d\times d}, \\ S_{F,i} &= \text{diag}\left(\frac{\partial \sigma_F^1}{\partial z_i}, \frac{\partial \sigma_F^2}{\partial z_i}, \ldots, \frac{\partial \sigma_F^d}{\partial z_i}\right)_{d\times d}, \quad \text{and} \\ B_{\epsilon_D} &= \left(S_{D,1}\epsilon_D, S_{D,2}\epsilon_D, \ldots, S_{D,d}\epsilon_D\right)_{K\times d}, \\ S_{D,i} &= \text{diag}\left(\frac{\partial \sigma_D^1}{\partial z_i'}, \frac{\partial \sigma_D^2}{\partial z_i'}, \ldots, \frac{\partial \sigma_D^d}{\partial z_i'}\right)_{K\times K}. \end{aligned}$$

This yields

$$\begin{aligned} \mathbb{E}\left[J_{F\circ D}^T J_{F\circ D}\right] &= \mathbb{E}\left[\left(J_{\mu_D} J_{\mu_F} + J_{\mu_D} B_{\epsilon_F} + B_{\epsilon_D} J_{\mu_F} + B_{\epsilon_D} B_{\epsilon_F}\right)^T \left(J_{\mu_D} J_{\mu_F} + J_{\mu_D} B_{\epsilon_F} + B_{\epsilon_D} J_{\mu_F} + B_{\epsilon_D} B_{\epsilon_F}\right)\right] \\ &= J_{\mu_F}^T J_{\mu_D}^T J_{\mu_D} J_{\mu_F} + \mathbb{E}\left[B_{\epsilon_F}^T J_{\mu_D}^T J_{\mu_D} B_{\epsilon_F}\right] + \mathbb{E}\left[J_{\mu_F}^T B_{\epsilon_D}^T B_{\epsilon_D} J_{\mu_F}\right] + \mathbb{E}\left[B_{\epsilon_F}^T B_{\epsilon_D}^T B_{\epsilon_D} B_{\epsilon_F}\right] \\ &= J_{\mu_F}^T \left(J_{\mu_D}^T J_{\mu_D} + \mathbb{E}\left[B_{\epsilon_D}^T B_{\epsilon_D}\right]\right) J_{\mu_F} + \mathbb{E}\left[B_{\epsilon_F}^T \left(J_{\mu_D}^T J_{\mu_D} + B_{\epsilon_D}^T B_{\epsilon_D}\right) B_{\epsilon_F}\right] \end{aligned}$$

where in the second equality we used the fact that ϵ_D, ϵ_F are independent and $\mathbb{E}\left[B_\epsilon\right] = 0$. By Lemma 1 we have

$$\mathbb{E}\left[B_{\epsilon_D}^T B_{\epsilon_D}\right] = J_{\sigma_D}^T J_{\sigma_D}.$$

Similarly,

$$\mathbb{E}\left[B_{\epsilon_F}^T B_{\epsilon_D}^T B_{\epsilon_D} B_{\epsilon_F}\right] = \mathbb{E}\left[\mathbb{E}\left[B_{\epsilon_F}^T B_{\epsilon_D}^T B_{\epsilon_D} B_{\epsilon_F} | \epsilon_F\right]\right] = \mathbb{E}\left[B_{\epsilon_F}^T J_{\sigma_D}^T J_{\sigma_D} B_{\epsilon_F}\right] = J_{\sigma_F}^T \text{diag}\big(J_{\sigma_F}^T J_{\sigma_F}\big) J_{\sigma_F}$$

Finally,

$$\mathbb{E}\left[B_{\epsilon_F}^T J_{\mu_D}^T J_{\mu_D} B_{\epsilon_F}\right] = J_{\sigma_F}^T \text{diag}\big(J_{\mu_D}^T J_{\mu_D}\big) J_{\sigma_F}.$$

The proof is complete by taking expectation with respect to uniformly distributed set of decoders $\{D_i\}_{i=1}^M$. □