Efficient Gaussian Processes for Model-based Online Planning

Hu Hanyang
Supervisor: Jonathan Scarlett

Department of Mathematics
National University of Singapore
Mathematics Capstone Project for Semester 2, AY2024/2025

April 15, 2025

Outline

Introduction

In **model-free reinforcement learning**, 1 million time-steps is common for training, which might be infeasible for real-world applications.

Model-based reinforcement learning (MBRL), particularly online planning, may converge much earlier than 200k time-steps.[1]

[1] Tingwu Wang et al. *Benchmarking Model-Based Reinforcement Learning*. 2019. URL: https://arxiv.org/abs/1907.02057.

Introduction

In **model-free reinforcement learning**, 1 million time-steps is common for training, which might be infeasible for real-world applications.

Model-based reinforcement learning (MBRL), particularly online planning, may converge much earlier than 200k time-steps.[1]

Question: Can we make MBRL more **sample-efficient** by replacing the common MLP dynamics model with **Gaussian processes (GPs)**?

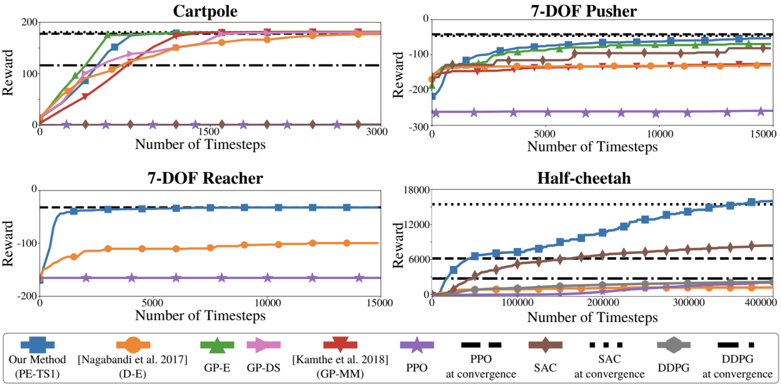
Challenges of GP dynamics for online planning:

- computational complexity (i.e., slow training and inference)
- curse of dimensionality (CoD)

[1] Tingwu Wang et al. *Benchmarking Model-Based Reinforcement Learning*. 2019. URL: https://arxiv.org/abs/1907.02057.

Existing Works on GP-based Planning

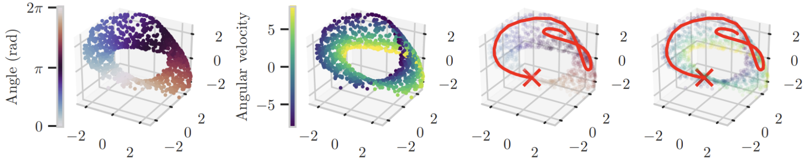
Results in the PE-TS[2] paper:



[2] Kurtland Chua et al. *Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models*. 2018.

Examples focusing on GP-based Planning:

1. (Hewing et al., 2020) Propagating uncertainty by first-order approximations (similar to extended Kalman filters) and selecting inducing points dynamically.
2. (Bosch et al., 2020) Using a neural network auto-encoder to alleviate the CoD, GP dynamics then plan in the latent space.



Note. These methods are typically tested only in simple environments.

Motivation

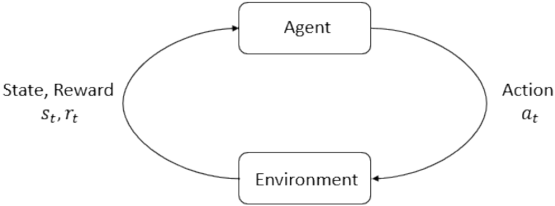
Overarching Goal: Extend GP-based planning to more diverse domains while maintaining real-time performance and advantage over NN models.

Outline

Markov Decision Processes

A **Markov decision process (MDP)** is defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, M, r, \rho_0)$

- state space $\mathcal{S}$
- action space $\mathcal{A}$
- transition probability distribution (dynamics) $s_{t+1} \sim M(\cdot|s_t, a_t)$
- reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- initial state distribution $s_0 \sim \rho_0(\cdot)$



Source: OpenAI Spinning Up (`spinningup.openai.com`).

Markov Decision Processes

Goal: Find a (deterministic) **policy** function $\pi : \mathcal{S} \to \mathcal{A}$ such that

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau \sim P^{\pi}(\cdot)}[R(\tau)]$$

where $R(\cdot)$ denotes the **infinite-horizon discounted return**

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

with discount factor $\gamma \in (0, 1)$.

Markov Decision Processes

Goal: Find a (deterministic) **policy** function $\pi : \mathcal{S} \to \mathcal{A}$ such that

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau \sim P^{\pi}(\cdot)}[R(\tau)]$$

where $R(\cdot)$ denotes the **infinite-horizon discounted return**

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

with discount factor $\gamma \in (0, 1)$.

Question: How to maximize this objective when we only have a set of sampled trajectories with finite length collected from a non-optimal policy?

Bellman Equation

Consider the **action-value function** $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim P^\pi(\cdot)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,\middle|\, s_0 = s, a_0 = a \right]$$

which satisfies the **Bellman equation**

$$Q^\pi(s, a) = r(s, a) + \gamma \cdot \mathbb{E}_{s' \sim M(\cdot|s,a)}[Q^\pi(s', \pi(s'))].$$

Bellman Equation

Consider the **action-value function** $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

$$Q^\pi(s,a) = \mathbb{E}_{\tau \sim P^\pi(\cdot)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,\middle|\, s_0 = s, a_0 = a \right]$$

which satisfies the **Bellman equation**

$$Q^\pi(s,a) = r(s,a) + \gamma \cdot \mathbb{E}_{s' \sim M(\cdot|s,a)}[Q^\pi(s', \pi(s'))].$$

The **optimal action-value function** Q^* should satisfy

$$Q^*(s,a) = r(s,a) + \gamma \cdot \mathbb{E}_{s' \sim M(\cdot|s,a)}[\max_{a \in \mathcal{A}} Q^*(s', a))]$$

and hence the optimal policy can be extracted by $\pi^*(s) = \max_a Q^*(s,a)$.

Deep Deterministic Policy Gradient

(1) Given a policy π, we can train a Q-network Q_θ by minimizing

$$L(\theta, \mathcal{D}) = \mathbb{E}_{(s_t, a_t, s_{t+1}, r_t) \sim \mathcal{D}}[(Q_\theta(s_t, a_t) - \underbrace{(r_t + \gamma Q_\theta(s_{t+1}, \pi(s_{t+1}))))}_{\text{temporal-difference (TD) target}})^2]$$

so that Q_θ approximates Q^π (referred to as **TD learning**).

(2) Find the policy network $\pi_\theta(s) \approx \arg\max_a Q^\pi(s, a)$ by maximizing

$$\mathbb{E}_{s \sim \mathcal{D}}[Q_{\theta_{\text{targ}}}(s, \pi_\theta(s))]$$

where θ_{targ} is a lagged target Q-network introduced to stabilize training.

The **deterministic policy gradient** theorem (Silver et al., 2014) states that this is approximately the same as the original objective of the MDP.

Model-based Online Planning

Intuition: RL is difficult, regression is easy.

With a learned dynamics model M_θ and reward model R_θ, the optimal action can be selected by **model predictive control (MPC)**

$$\pi_{\mathrm{MPC}}(s_t) = \arg\max_{a_t} \max_{a_{t+1:t+H}} \mathbb{E}\left[\sum_{i=0}^{H} \gamma^i R_\theta(s_{t+i}, a_{t+i})\right].$$

Model-based Online Planning

Algorithm 1 Model Predictive Control (MPC)

Input: Number of iterations J, population size N, number of elite samples K, roll-out horizon H, initial distribution parameters μ^0, Σ^0, (learned) dynamics model M_θ, (learned) reward model R_θ, current state s_t.
for each iteration $i = 1, 2, \ldots, J$ **do**
    Sample N action sequences of length H from $\mathcal{N}(\mu^{j-1}, \Sigma^{j-1})$.
    for all N sequences $\Gamma = (a_t, a_{t+1}, \ldots, a_{t+H})$ **do**
        for step $j = 0, 1, \ldots, H-1$ **do**   ▷ Estimate trajectory return ϕ_Γ
            Update $\phi_\Gamma = \phi_\Gamma + \gamma^t R_\theta(s_{t+j}, a_{t+j})$.   ▷ Initially setting $\phi_\Gamma = 0$
            Predict $s_{t+j+1} \sim M_\theta(s_{t+j}, a_{t+j})$.
        end for
    end for
    Select the elite samples $\{\Gamma_k^*\}_{k=1}^K$ corresponding to the top-K returns $\{\phi_{\Gamma_k^*}\}_{k=1}^K$.
    Update parameters μ^j, σ^j for the next iteration based on $\{\Gamma_k^*\}_{k=1}^K$ and $\{\phi_{\Gamma_k^*}\}_{k=1}^K$.
end for
Output: $(a_t^*, a_{t+1}^*, \ldots, a_{t+H}^*) \sim \mathcal{N}(\mu^J, \Sigma^J)$

Time Complexity: $O(JNH)$ model inference steps. J is the number of iterations, N is the population size, and H is the planning horizon.

TD Learning for MPC (TD-MPC)

Intuition: Use TD learning to model global optimality and use MPC to refine local behaviors, requiring a shorter planning horizon H.

Modified MPC Objective:

$$\pi_{\text{TD-MPC}}(s_t) = \arg\max_{a_t} \max_{a_{t+1:t+H}} \mathbb{E}\left[\sum_{i=0}^{H-1} \gamma^i R_\theta(s_{t+i}, a_{t+i}) + \gamma^H Q_\theta(s_H, a_H)\right].$$

Note. All components of TD-MPC are implemented using deterministic neural networks.

Model-based Value Error of TD-MPC

Motivation: The model-based value error partially determines the performance of model-based online planning.

Theorem (modified and extended from Xiao et al., Theorem 1)

The model-based value error is bounded by

$$\left|V^{\pi}(s) - \hat{V}^{\pi}(s)\right| \le \underbrace{K_{\mathcal{M}} \frac{\gamma - \gamma^{H+1}}{1-\gamma} \epsilon_m}_{\text{dynamics gap}} + \underbrace{\frac{1-\gamma^{H}}{1-\gamma} \epsilon_r + \gamma^{H} \epsilon_q}_{\text{return estimation gap}}$$

and $K_{\mathcal{M}} \le (L_R + 2\gamma V_{\max} L_M)\sqrt{1 + L_{\pi}^2}$.

Notations:

1. **Dynamics Model Error** $\max_{s,a} W(M(\cdot|s,a), \hat{M}(\cdot|s,a)) \le \epsilon_m$;
2. **Reward Model Error** $\max_{s,a} |r(s,a) - \hat{r}(s,a)| \le \epsilon_r$;
3. **Value Function Error** $\max_{s,a} |Q^{\pi}(s,a) - \hat{Q}^{\pi}(s,a)| \le \epsilon_q$.

Outline

Overview

Key Design Choices:

1. Integration with TD-MPC.
2. GP-based Correction of MLPs.
3. Decoupled Training and Inference.
4. Integration with Deep Kernel Learning (DKL).

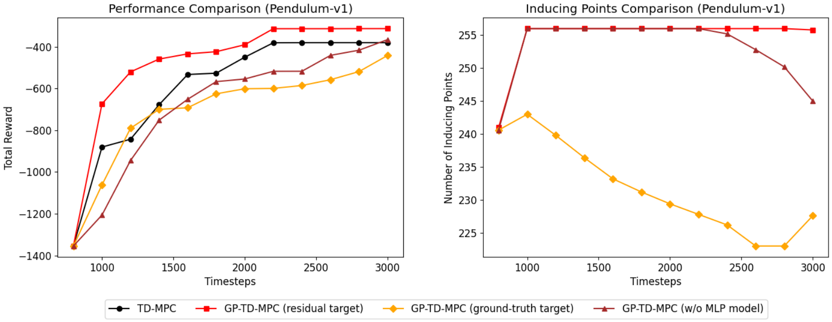
Note. We would refer to our method as GP-TD-MPC.

GP-based Correction

Instead of standalone GPs, we use an MLP model $f_\theta(\cdot)$ as the prior mean

$$\text{output}(\mathbf{x}^*) = f_\theta(\mathbf{x}^*) + \underbrace{\mathbf{k}_{X\mathbf{x}^*}^\top \hat{K}_{XX}^{-1}(\mathbf{y} - f_\theta(X))}_{\text{GP correction}}$$

and the GP training target becomes the residual $\mathbf{y} - f_\theta(X)$.



Computational Bottlenecks

Training a GP model involves maximizing the marginal log-likelihood (MLL) w.r.t. the kernel hyperparameters θ (e.g., the lengthscales)

$$\mathcal{L} = \log p(\mathbf{y} \mid X, \theta) \propto -\underbrace{(\mathbf{y} - m(X))^\top \hat{K}_{XX}^{-1}(\mathbf{y} - m(X))}_{\text{model fit}} - \underbrace{\log |\hat{K}_{XX}|}_{\text{complexity}}$$

which is computation-heavy.

Cached inference of the GP correction

$$\text{output}(\mathbf{x}^*) = f_\theta(\mathbf{x}^*) + \underbrace{\mathbf{k}_{X\mathbf{x}^*}^\top \hat{K}_{XX}^{-1}(\mathbf{y} - f_\theta(X))}_{\text{GP correction}}$$

takes $O(n)$ time for each independent GP (n is the size of the data).

Stochastic Variational Gaussian Process (SVGP)

Evidence Lower Bound (ELBO): Using a **variational distribution** $q(\mathbf{u}) \sim \mathcal{N}(\mathbf{m_u}, S_\mathbf{u})$ w.r.t. m inducing points Z to approximate $p(\mathbf{f}, \mathbf{u}) = p(\mathbf{f} \mid \mathbf{u})p(\mathbf{u}) \approx p(\mathbf{f} \mid \mathbf{u})q(\mathbf{u})$ so that

$$\log p(\mathbf{y}|X, \theta) \geq \mathbb{E}_{q(\mathbf{u})p(\mathbf{f}|\mathbf{u})}[\log p(\mathbf{y}|\mathbf{f})] - D_{\mathrm{KL}}(q(\mathbf{u}) \| p(\mathbf{u}))$$

where $p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{n} p(\mathbf{y}_i|\mathbf{f}_i)$ usually factorizes over data instances.

Stochastic Variational Gaussian Process (SVGP)

Evidence Lower Bound (ELBO): Using a **variational distribution** $q(\mathbf{u}) \sim \mathcal{N}(\mathbf{m}_\mathbf{u}, S_\mathbf{u})$ w.r.t. m inducing points Z to approximate $p(\mathbf{f}, \mathbf{u}) = p(\mathbf{f} \mid \mathbf{u})p(\mathbf{u}) \approx p(\mathbf{f} \mid \mathbf{u})q(\mathbf{u})$ so that

$$\log p(\mathbf{y}|X, \theta) \geq \mathbb{E}_{q(\mathbf{u})p(\mathbf{f}|\mathbf{u})}[\log p(\mathbf{y}|\mathbf{f})] - D_{\mathrm{KL}}(q(\mathbf{u}) \| p(\mathbf{u}))$$

where $p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{n} p(\mathbf{y}_i|\mathbf{f}_i)$ usually factorizes over data instances.

The inference can also be approximated by

$$\begin{aligned} \mathbb{E}[f(\mathbf{x}^*) \mid X, \mathbf{y}, \mathbf{x}^*] &\approx \mathbf{k}_{Z\mathbf{x}^*}^\top K_{ZZ}^{-1} \mathbf{m}_\mathbf{u} \\ \mathrm{Var}[f(\mathbf{x}^*) \mid X, \mathbf{y}, \mathbf{x}^*] &\approx k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}_{Z\mathbf{x}^*}^\top K_{ZZ}^{-1}(K_{ZZ} - S_\mathbf{u})K_{ZZ}^{-1}\mathbf{k}_{Z\mathbf{x}^*} \end{aligned}$$

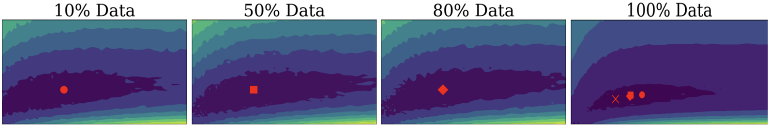
where the mean prediction only takes $O(m)$ time.

Note. We consider $m \equiv 0$ for simplicity.

Decoupled Training and Inference

Motivation: The SVGP training introduces $O(m^2)$ additional parameters, which we might want to avoid to reduce training time.

Training: Subsample a mini-batch of data for scalable training.[3]

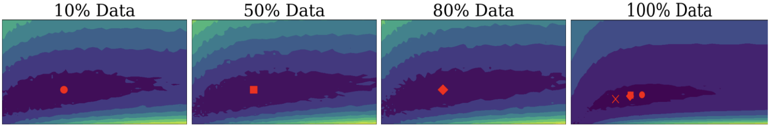


[3] Shifan Zhao et al. *Efficient Two-Stage Gaussian Process Regression Via Automatic Kernel Search and Subsampling*. 2024.

Decoupled Training and Inference

Motivation: The SVGP training introduces $O(m^2)$ additional parameters, which we might want to avoid to reduce training time.

Training: Subsample a mini-batch of data for scalable training.[3]



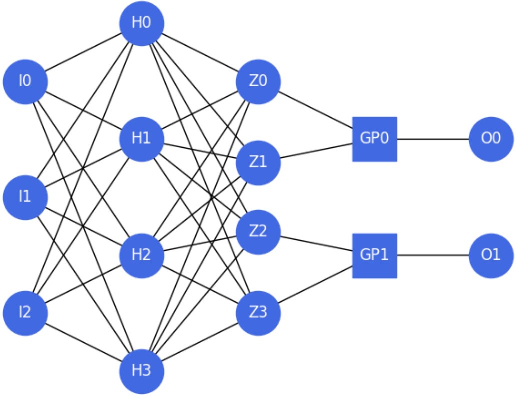
Inference: Choose Z by **farthest point sampling** or **pivoted Cholesky**, then obtain the optimal variational distribution in $O(nm^2)$ time

$$\mathbf{c} = K_{ZX}\Sigma_{\mathbf{y}}^{-1}(\mathbf{y} - f_\theta(X)), \qquad C = K_{ZX}\Sigma_{\mathbf{y}}^{-1}K_{XZ}$$
$$\mathbf{m}_{\mathbf{u}} = K_{ZZ}(K_{ZZ} + C)^{-1}\mathbf{c}, \qquad S_{\mathbf{u}} = K_{ZZ}(K_{ZZ} + C)^{-1}K_{ZZ}.$$

[3] Shifan Zhao et al. *Efficient Two-Stage Gaussian Process Regression Via Automatic Kernel Search and Subsampling*. 2024.

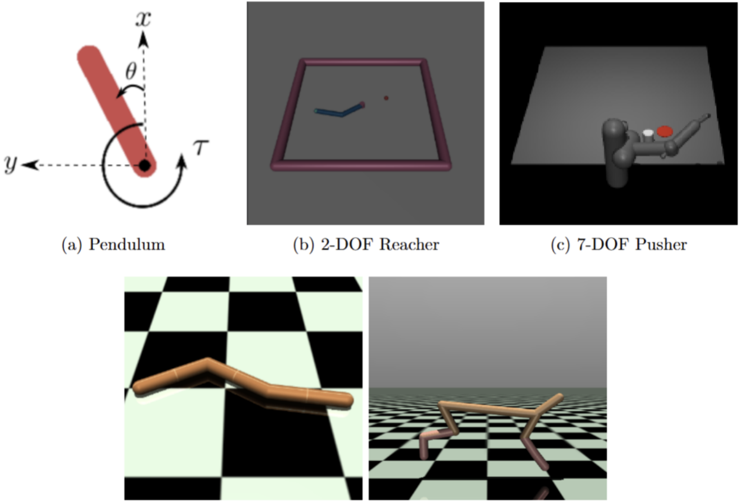
Deep Kernel Learning

In non-stationary and/or high-dimensional settings, we may introduce a neural network feature extractor as kernel hyperparameters, known as **deep kernel learning (DKL)**.



Outline

Overview

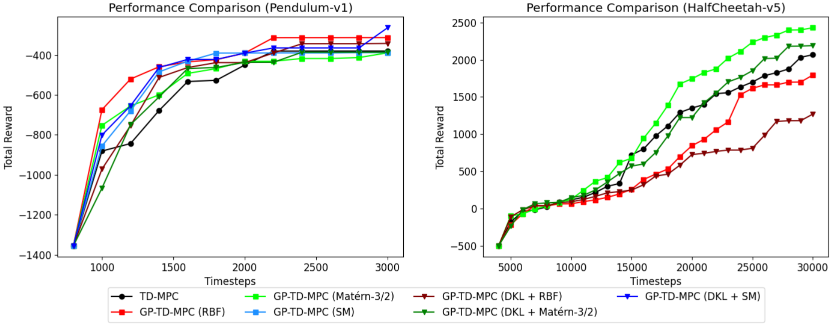


(a) Pendulum (b) 2-DOF Reacher (c) 7-DOF Pusher

(d) Swimmer (e) Half Cheetah

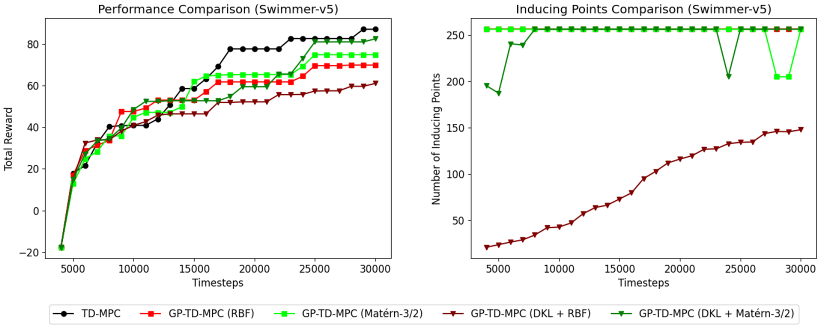
Successful Cases

Certain variants of GP-TD-MPC outperformed the baseline:



Failed Cases

For the Swimmer task, only GP-TD-MPC with the Matérn kernel and DKL delivered comparable performance in the end.



Key Findings

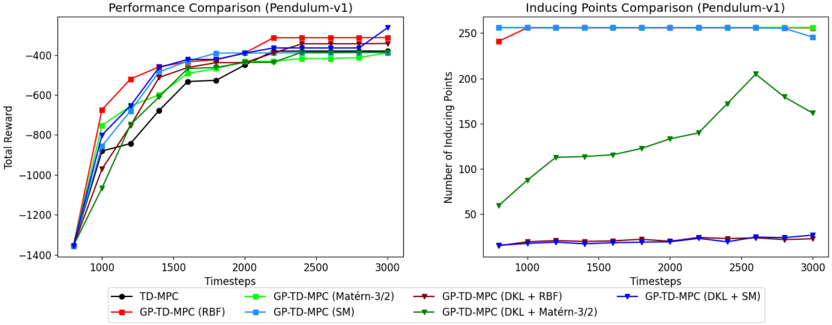
From the experimental results, we derive the following observations:

1. GP-TD-MPC with DKL typically requires fewer inducing points to achieve the error tolerance for the pivoted Cholesky method.
2. GP-TD-MPC with the Matérn-3/2 kernel consistently matches or even outperforms the TD-MPC baseline.

Number of Inducing Points

The **pivoted Cholesky method** selects up to m inducing points and may terminate early if the truncation error falls below the specified tolerance.

For $m \leq 256$, only DKL variants reached the default tolerance.



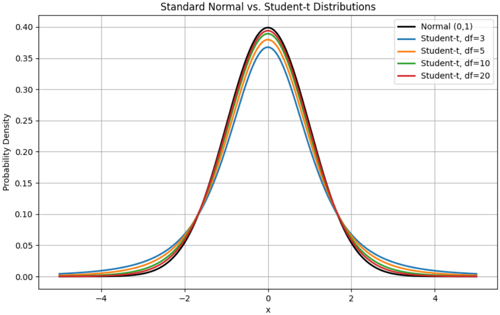
Modeling Contact Dynamics

The spectral densities of the RBF kernel and the Matérn kernel are

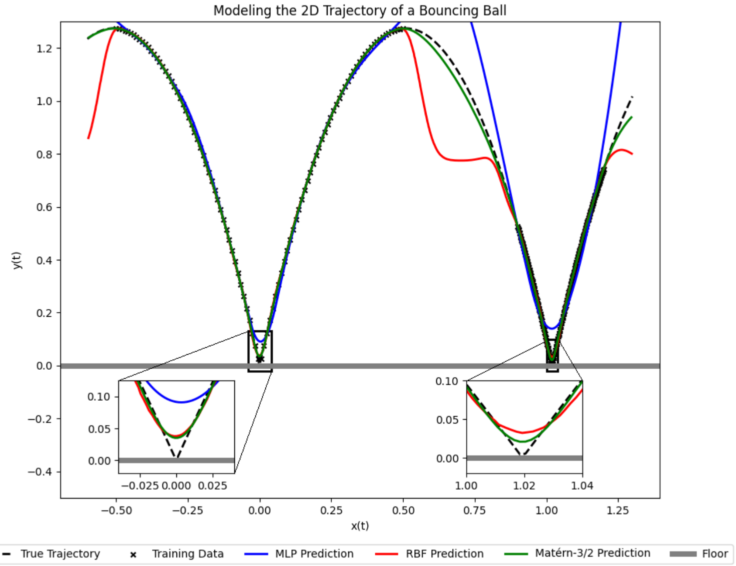
$$p_l^{\mathrm{RBF}}(s) = (2\pi l^2)^{D/2} \exp\left(-2\pi^2 l^2 s^2\right)$$

$$p_{l,\nu}^{\mathrm{Matérn}}(s) = \frac{2^D \pi^{D/2} \Gamma(\nu + D/2)(2\nu)^\nu}{\Gamma(\nu) l^{2\nu}} \left(\frac{2\nu}{l^2} + 4\pi^2 s^2\right)^{-(\nu + D/2)}$$

resembling the Gaussian distribution and the t-distribution.



Toy Example: Trajectory of a Bouncing Ball



Runtime and Efficiency

Task Name	TD-MPC	RBF		Matérn-3/2		Spectral Mixture	
		Standard	DKL	Standard	DKL	Standard	DKL
Pendulum	175.13	260.31	266.87	279.64	297.81	430.92	287.64
Reacher	127.96	172.53	173.18	173.31	174.48	856.31	182.29
Pusher	201.40	305.83	293.66	315.20	302.04	–	–
Swimmer ($m \leq 256$)	1049.06	1531.21	1603.92	1537.10	1635.02	–	–
Swimmer ($m \leq 1024$)	1049.06	1592.99	1603.02	1666.68	1732.88	–	–
Half Cheetah	1064.46	2019.56	1631.59	2794.53	2818.31	–	–

Table 1: Comparison of total runtime (in seconds) averaged across 5 trials.

Outline

Alternative Inference Methods

Motivation: For high-dimensional tasks that require a larger amount of data, more inducing points might be required. However, this may slow down inference. Instead, we consider alternative inference methods.

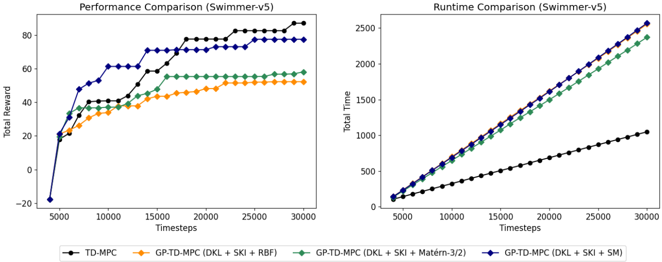
1. **Local Kernel Interpolation.** $O(1)$-time inference, scaling more effectively with a larger number of inducing points;
2. **Dynamical Local Projection.** Making more efficient use of the limited inducing points.

Local Kernel Interpolation

Use **cubic interpolation** to obtain sparse matrix W s.t. $WK_{XX} \approx K_{XZ}$ and hence $K_{XX} \approx K_{XZ}K_{ZZ}^{-1}K_{ZX} \approx W^\top K_{ZZ}W$.

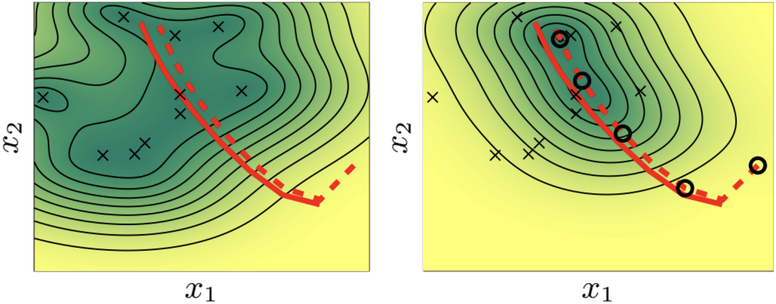
At any test location $\mathbf{x}^*$, compute the sparse interpolation vector $\mathbf{w}_{\mathbf{x}^*}$ s.t. $K_{ZZ}\mathbf{w}_{\mathbf{x}^*} \approx \mathbf{k}_{Z\mathbf{x}^*}$, we can approximate the original GP correction by

$$\text{output}(\mathbf{x}^*) = f_\theta(\mathbf{x}^*) + \underbrace{\mathbf{w}_{\mathbf{x}^*}^\top K_{ZZ}W(W^\top K_{ZZ}W + \sigma_\epsilon^2\mathbf{I})^{-1}(\mathbf{y} - f_\theta(X))}_{\text{LKI Correction}}.$$



Dynamical Local Projection

Inspired by **online variational conditioning (OVC)**[4] and **dynamic sparse GPs for MPC**[5].



[4] Wesley J. Maddox, Samuel Stanton, and Andrew Gordon Wilson. *Conditioning Sparse Variational Gaussian Processes for Online Decision-making*. 2021.

[5] Lukas Hewing, Juraj Kabzan, and Melanie N. Zeilinger. "Cautious Model Predictive Control Using Gaussian Process Regression". In: (2020).

Dynamical Local Projection

Given a set of M inducing points Z, we compute the corresponding $\mathbf{c}$ and C, they can thus be projected to a subset $Z' \subseteq Z$ of size m by

$$\mathbf{c}' = K_{Z'X}\Sigma_{\mathbf{y}}^{-1}\mathbf{y} \approx K_{Z'Z}(K_{ZZ}^{-1}\mathbf{c}),$$
$$C' = K_{Z'X}\Sigma_{\mathbf{y}}^{-1}K_{XZ'} \approx K_{Z'Z}(K_{ZZ}^{-1}CK_{ZZ}^{-1})K_{ZZ'}.$$

to compute the optimal variational distribution w.r.t. Z'. Each time, DLP takes $O(M^2m)$ time, independent of the dataset size.

Dynamical Local Projection

Given a set of M inducing points Z, we compute the corresponding $\mathbf{c}$ and C, they can thus be projected to a subset $Z' \subseteq Z$ of size m by

$$\mathbf{c}' = K_{Z'X}\Sigma_{\mathbf{y}}^{-1}\mathbf{y} \approx K_{Z'Z}(K_{ZZ}^{-1}\mathbf{c}),$$
$$C' = K_{Z'X}\Sigma_{\mathbf{y}}^{-1}K_{XZ'} \approx K_{Z'Z}(K_{ZZ}^{-1}CK_{ZZ}^{-1})K_{ZZ'}.$$

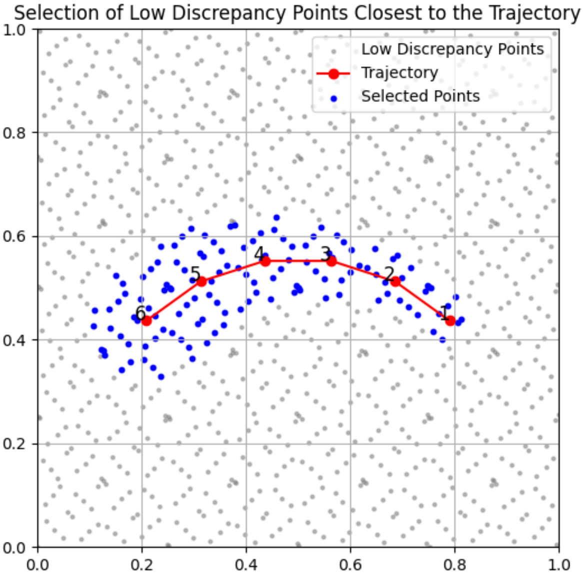
to compute the optimal variational distribution w.r.t. Z'. Each time, DLP takes $O(M^2m)$ time, independent of the dataset size.

Let $\mathbf{p}$ be the reference path, we can select Z' according to

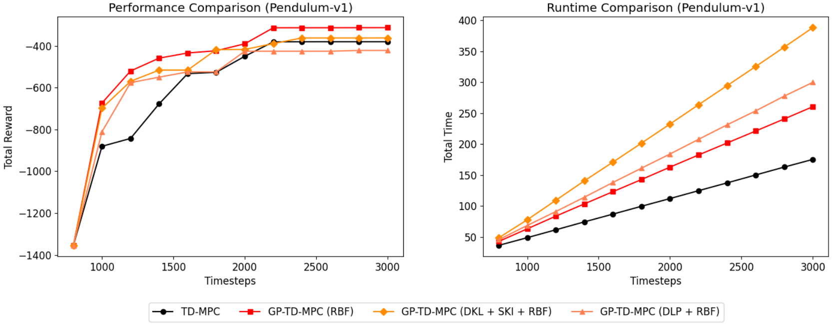
$$\mathrm{cost}(\mathbf{z}, \mathbf{p}) = \min_{h=1,\ldots,H-1} \eta^h \mathrm{dist}(\mathbf{z}, \overline{\mathbf{p}_h\mathbf{p}_{h+1}})$$

which encourages more points to be selected near line segments corresponding to larger timesteps.

Dynamical Local Projection



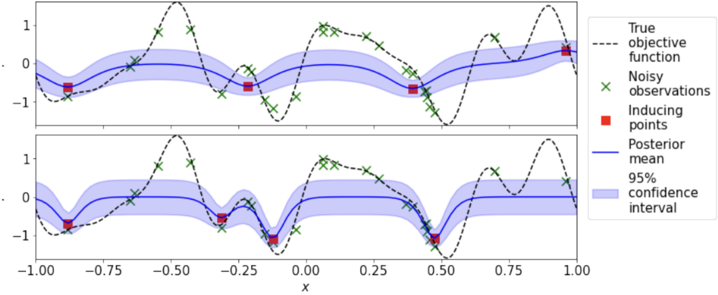
Dynamical Local Projection



Future Work

We recommend some directions for future explorations:

1. More informative **inducing point allocation (IPA)**.[6]
2. Uncertainty quantification using **pathwise conditioning**.[7]
3. **Kernel composition** for domain-specific applications.



[6] Henry B. Moss, Sebastian W. Ober, and Victor Picheny. *Inducing Point Allocation for Sparse Gaussian Processes in High-Throughput Bayesian Optimisation*. 2023.

[7] James T. Wilson et al. *Pathwise Conditioning of Gaussian Processes*. 2021.

Outline