Meta Reinforcement Learning with Latent Variable Gaussian Processes: Supplement

1 Model-Based Meta Reinforcement Learning

We consider stochastic systems of the form

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{c}_t) + \boldsymbol{\epsilon} \tag{1}$$

with state variables $\boldsymbol{x} \in \mathbb{R}^D$, control signals $\boldsymbol{c} \in \mathbb{R}^K$ and i.i.d. system noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{E})$, where $\boldsymbol{E} = \text{diag}(\sigma_1^2, \ldots, \sigma_D^2)$.

We model the distribution over systems using a latent embedding $\boldsymbol{h}$ and model the dynamics using a global function conditioned on the latent embedding. For a sample f_p from the distribution we modify the dynamics from (1) to

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{c}_t, \boldsymbol{h}_p) + \boldsymbol{\epsilon}\,, \tag{2}$$

such that the successor state depends on the latent system specification $\boldsymbol{h}_p$. This means, we explicitly model the global properties through a shared function f and the task-specific variation using a distribution over the latent variables $p(\boldsymbol{h}_p)$. Framing the meta learning problem as a hierarchical Bayesian model means that meta-training becomes inference in a meta-learning model. The meta RL procedures for training and testing are detailed in algorithms 1 and 2, respectively.

1.1 Meta-Learning Gaussian Process

Our meta-learning model is a GP prior on the unknown transition function in (2) with a concatenated state $\tilde{\boldsymbol{x}}_t = (\boldsymbol{x}_t, \boldsymbol{c}_t, \boldsymbol{h}_p) \in \mathbb{R}^{D+K+Q}$ as the input to the model. We define $\boldsymbol{y}_t = \boldsymbol{x}_{t+1} - \boldsymbol{x}_t$ as the targets of the GP and take the mean function to be $m(\tilde{\boldsymbol{x}}_t) = \boldsymbol{0}$, which encodes that a priori the state does not change [2]. Each dimension of the targets $\boldsymbol{y}$ is modeled by an independent GP. We use a Gaussian likelihood

$$p(\boldsymbol{y}_t | \tilde{\boldsymbol{x}}_t, \boldsymbol{f}(\cdot), \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_t | \boldsymbol{f}(\tilde{\boldsymbol{x}}_t), \boldsymbol{E}), \tag{3}$$

where $\boldsymbol{\theta} = \{\boldsymbol{E}, \boldsymbol{L}, \sigma_f^2, Q\}$ are the hyperparameters and $\boldsymbol{f}(\cdot) = \big(f^1(\cdot), \ldots, f^D(\cdot)\big)$ denotes a multi-dimensional function. We place a standard-normal prior $\boldsymbol{h}_p \sim$

Initialize dataset D and model M

▷ Initial random rollouts

forall *training tasks* **do**
  execute random policy
  add observations to D
end

▷ Meta training

while *training tasks not solved* **do**
  —**update**—: train M and infer $\boldsymbol{h}$ given D
  forall *unsolved training tasks* **do**
    for *each step in horizon* **do**
      —**plan**—: get control sequence using (13)
      —**execute**—: execute first control in sequence
    end
    add observations to D
    check if task solved
  end
end

Algorithm 1: Model-based Meta RL with MPC (Train)

$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ on the latent variables $\boldsymbol{h}_p$. The full specification of the model is

$$\begin{aligned} & p(\boldsymbol{Y}, \boldsymbol{H}, \boldsymbol{f}(\cdot)|\boldsymbol{X}, \boldsymbol{C}) \\ & = \prod_{p=1}^{P} p(\boldsymbol{h}_p) \prod_{t=1}^{T_p} p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{c}_t, \boldsymbol{h}_p, \boldsymbol{f}(\cdot)) p(\boldsymbol{f}(\cdot)) \end{aligned} \tag{4}$$

where we denote a collection of vectors in bold upper-case and we have dropped dependence on the hyperparameters for notation purposes. The corresponding graphical model is given in Fig. 1. The figure shows the dependence of individual system observations on the global GPs $\boldsymbol{f}(\cdot)$ modeling each dimension of the outputs, the system-specific latent embeddings $\boldsymbol{h}_p$ and the observed states and controls.

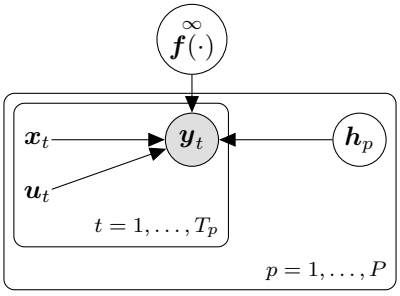

Figure 1: Graphical model for our ML-GP model.

Given dataset D and model M from training

▷ Single shot performance

forall *test tasks* **do**
  for *each step in horizon* **do**
    —**plan**—: get control sequence using (13)
    —**execute**—: execute first control in sequence
    —**inference**—: infer the value of $\boldsymbol{h}_*$ given observations so far
  end
  add observations to D
  check if task solved
end

▷ Meta test

while *test tasks not solved* **do**
  —**update**—: train M and infer $\boldsymbol{h}$ given D
  forall *unsolved test tasks* **do**
    for *each step in horizon* **do**
      —**plan**—: get control sequence using (13)
      —**execute**—: execute first control in sequence
    end
    add observations to D
    check if task solved
  end
end

Algorithm 2: Model-based Meta RL with MPC (Test)

1.2 Inference

To learn the dynamics model we seek to optimize the hyperparameters $\boldsymbol{\theta}$ w.r.t. the log-marginal likelihood, which involves marginalization of the latent variables in (4). For predictions of the evolution of a system we also need to infer the posterior GP and the posterior distribution of the latent variables $\boldsymbol{H} = (\boldsymbol{h}_1, ..., \boldsymbol{h}_P)$. We approach this problem with approximate variational inference. We posit a variational distribution that assumes independence between the latent functions of the GP and the latent task variables

$$Q(\boldsymbol{f}(\cdot), \boldsymbol{H}) = q(\boldsymbol{f}(\cdot))q(\boldsymbol{H}) \tag{5}$$

and minimize the Kullback-Leibler divergence between the approximate and true posterior distributions. Equivalently we can maximize the evidence lower bound

$$\mathcal{L} = \mathbb{E}_{Q(\boldsymbol{f}(\cdot),\boldsymbol{H})}\Big[\log \frac{p(\boldsymbol{Y}, \boldsymbol{H}, \boldsymbol{f}(\cdot)|\boldsymbol{X}, \boldsymbol{C})}{Q(\boldsymbol{f}(\cdot), \boldsymbol{H})}\Big], \tag{6}$$

which lower-bounds the log-marginal likelihood [5]. We parameterize our variational distribution such that we can compute the lower bound in (6). We then jointly optimize $\mathcal{L}$ with respect to the model hyperparameters and the variational parameters.

Sparse Gaussian Processes It is important to account for the fact that training a GP on a joint data set of P different systems quickly becomes infeasible due to the $\mathcal{O}(T^3)$ computational complexity for training and $\mathcal{O}(T^2)$ for predictions where T is the total number of observations. To address this we turn to the variational sparse GP approximation [9] and approximate the posterior GP with a variational distribution $q(\boldsymbol{f}(\cdot))$ that depends on a small set of $M \ll T$ *inducing points*. We introduce a set of M inducing inputs $\boldsymbol{Z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M) \in \mathbb{R}^{M \times (D+K+Q)}$, which live in the same space as $\tilde{\boldsymbol{x}}$, with corresponding GP function values $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_M) \in \mathbb{R}^{M \times D}$. We follow [4] and specify the variational approximation as a combination of the conditional GP prior and a variational distribution over the inducing function values, independent across output dimensions

$$q(f^d(\cdot)) = \int p(f^d(\cdot)|\boldsymbol{u}^d) q(\boldsymbol{u}^d) d\boldsymbol{u}^d. \tag{7}$$

where $q(\boldsymbol{u}^d) = \mathcal{N}(\boldsymbol{u}^d | \boldsymbol{m}^d, \boldsymbol{S}^d)$ is a full rank Gaussian. The integral in (7) can be computed in closed form since both terms are Gaussian, resulting in a GP with mean and covariance functions given by

$$m_q(\cdot) = \boldsymbol{k}_Z^T(\cdot) \boldsymbol{K}_{ZZ}^{-1} \boldsymbol{m}^d \tag{8}$$

$$k_q(\cdot, \cdot) = k(\cdot, \cdot) - \boldsymbol{k}_Z^T(\cdot) \boldsymbol{K}_{ZZ}^{-1} (\boldsymbol{K}_{ZZ} - \boldsymbol{S}^d) \boldsymbol{K}_{ZZ}^{-1} \boldsymbol{k}_Z(\cdot) \tag{9}$$

where $[\boldsymbol{k}_Z(\cdot)]_i = k(\cdot, \boldsymbol{z}_i)$ and $[\boldsymbol{K}_{ZZ}]_{ij} = k(\boldsymbol{z}_i, \boldsymbol{z}_j)$. Here, the variational approach has two main benefits: a) it reduces the complexity of training to $\mathcal{O}(TM^2)$ and predictions to $\mathcal{O}(TM)$, b) it enables mini-batch training for further improvement in computational efficiency.

Latent Variables For the latent variables $\boldsymbol{H}$ we assume a Gaussian variational posterior

$$q(\boldsymbol{H}) = \prod\nolimits_{p=1}^{P} \mathcal{N}(\boldsymbol{h}_p | \boldsymbol{n}_p, \boldsymbol{T}_p) \tag{10}$$

where $\boldsymbol{T}_p$ is in general a full rank covariance matrix. We use a diagonal covariance in practice for more efficient computation of the ELBO (6).

Evidence Lower Bound (ELBO) Plugging in the definition of the model in eq. (3) and the specified variational distributions in (7), (10) into the definition of the ELBO in (6) we find

$$\begin{aligned}
\mathcal{L} &= \mathbb{E}_{Q(\boldsymbol{f}(\cdot), \boldsymbol{H})} \Big[\log \frac{p(\boldsymbol{Y}, \boldsymbol{H}, \boldsymbol{f}(\cdot) | \boldsymbol{X}, \boldsymbol{C})}{Q(\boldsymbol{f}(\cdot), \boldsymbol{H})} \Big] \\
&= \mathbb{E}_{q(\boldsymbol{f}(\cdot)) q(\boldsymbol{H})} \Big[\log \frac{\prod_{p=1}^{P} p(\boldsymbol{h}_p) \prod_{t=1}^{T_p} p(\boldsymbol{y}_t | \boldsymbol{x}_t, \boldsymbol{c}_t, \boldsymbol{h}_p, \boldsymbol{f}(\cdot)) p(\boldsymbol{f}(\cdot))}{q(\boldsymbol{f}(\cdot)) q(\boldsymbol{H})} \Big] \\
&= \sum\nolimits_{p=1}^{P} \sum\nolimits_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{f}_t | \boldsymbol{x}_t, \boldsymbol{c}_t, \boldsymbol{h}_p) q(\boldsymbol{h}_p)} \big[\log p(\boldsymbol{y}_t | \boldsymbol{f}_t) \big] \\
&\quad - \mathrm{KL}\big[q(\boldsymbol{H}) || p(\boldsymbol{H}) \big] - \mathrm{KL}\big[q(\boldsymbol{f}(\cdot)) || p(\boldsymbol{f}(\cdot)) \big]
\end{aligned}$$

where we emphasize that $q(\boldsymbol{f}_t|\boldsymbol{x}_t, \boldsymbol{c}_t, \boldsymbol{h}_p) = q(\boldsymbol{f}(\tilde{\boldsymbol{x}}_t)|\boldsymbol{x}_t, \boldsymbol{c}_t, \boldsymbol{h}_p)$ is the marginal of the GP evaluated at the inputs $\tilde{\boldsymbol{x}}_t$. The KL term for the latent variables $\boldsymbol{H}$ is analytically tractable since both terms are Gaussian. The KL term between the GPs has been shown to simplify to $\mathrm{KL}\big[q(\boldsymbol{U})||p(\boldsymbol{U})\big]$ [7], which again is analytically tractable since both terms are Gaussian. Thus the ELBO can be written

$$\begin{aligned}\mathcal{L} = \sum\nolimits_{p=1}^{P} \sum\nolimits_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{f}_t|\boldsymbol{x}_t,\boldsymbol{c}_t)}\big[\log p(\boldsymbol{y}_t|\boldsymbol{f}_t)\big] \\ -\mathrm{KL}\big[q(\boldsymbol{H})||p(\boldsymbol{H})\big] - \mathrm{KL}\big[q(\boldsymbol{U})||p(\boldsymbol{U})\big]\end{aligned} \tag{11}$$

The expected log-likelihood term in (11) needs further consideration: we would like to integrate out the latent variable $\boldsymbol{h}_p$ to obtain

$$q(\boldsymbol{f}_t|\boldsymbol{x}_t, \boldsymbol{c}_t) = \int q(\boldsymbol{f}_t|\boldsymbol{x}_t, \boldsymbol{c}_t, \boldsymbol{h}_p) q(\boldsymbol{h}_p) d\boldsymbol{h}_p \tag{12}$$

The integral in (12) is intractable due to the non-linear dependence on $\boldsymbol{h}_p$ in (8) and (9). Given our choice of kernel (RBF) and Gaussian variational distribution $q(\boldsymbol{h}_p)$ the first and second moments can be computed in closed form. We could use these terms to compute the log-likelihood term in closed form since the likelihood is Gaussian but in practice this can be prohibitively expensive since it requires the evaluation of a TM^2D tensor. Instead we avoid computing the moments by approximately integrating out the latent variable using Monte Carlo sampling.

$$\begin{aligned}\text{Hyperparameters:} \quad & \boldsymbol{\theta} = \{\boldsymbol{E}, \boldsymbol{L}, \sigma_f^2, Q\} \\ \text{Variational parameters:} \quad & \boldsymbol{\phi} = \{\boldsymbol{Z}, M\, \{\boldsymbol{m}^d, \boldsymbol{S}^d\}_{d=1}^{D}, \{\boldsymbol{n}_p, \boldsymbol{T}_p\}_{p=1}^{P}\}\end{aligned}$$

Training For the update steps in algorithms 1 and 2 we jointly optimize the GP hyperparameters and the variational parameters w.r.t. the ELBO. For the inference step in algorithm 2, we optimize only the variational parameters for the latent variables $\boldsymbol{h}$, i.e. $\boldsymbol{\phi}_{\boldsymbol{h}} = \{\boldsymbol{n}_p, \boldsymbol{T}_p\}_{p=1}^{P}$.

2 Expected Long Term Cost

RL with MPC Our objective is to find a sequence of optimal controls $\boldsymbol{c}_0^*, \ldots, \boldsymbol{c}_{H-1}^*$ that minimizes the expected finite-horizon cost

$$J = \mathbb{E}\left[\sum\nolimits_{t=1}^{H} \ell(\boldsymbol{x}_t)\right], \tag{13}$$

where $\boldsymbol{x}_t$ is the state of the system at time t and ℓ is a known immediate/ instantaneous cost function that encodes the task objective. We consider an episodic setting. Initial states $\boldsymbol{x}_0$ are sampled from $p(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$.

To find the optimal open-loop sequence $\boldsymbol{c}_0^*, \ldots, \boldsymbol{c}_{H-1}^*$, we compute the expected long-term cost J in (13) using Gaussian approximations $p(\boldsymbol{x}_1), \ldots, p(\boldsymbol{x}_H)$ for a given control sequence $\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{H-1}$. Then, we find an open-loop control sequence that minimizes the expected long-term cost and apply the first control signal $\boldsymbol{c}_0^*$ to the system, which transitions into the next state. Next we re-plan, i.e., we determine the next open-loop control sequence $\boldsymbol{c}_0^*, \ldots, \boldsymbol{c}_{H-1}^*$ from the new state. This iterative MPC approach turns an open-loop controller into a closed-loop controller. Combining MPC with learned GP models for the underlying dynamics increases the robustness to model errors and has shown improved data efficiency in RL [6].

Computing the Expected Long-Term Cost To compute the expected long-term cost in (13), we sum up the expected immediate costs $\mathbb{E}[\ell(\boldsymbol{x}_t)] = \int \ell(\boldsymbol{x}_t) p(\boldsymbol{x}_t) d\boldsymbol{x}_t$ for $t = 0, \ldots, H-1$. We assume Gaussian approximations $p(\boldsymbol{x}_t)$ of the state distribution and choose ℓ so that this expectation can be computed analytically. To obtain the marginals $p(\boldsymbol{x}_t)$ we iteratively predict the state evolution using the GP dynamics model. Section 1.2 details the long-term predictions the Gaussian approximations.

Long-term Predictions Key to computing the expected long-term cost in (13) is to compute long-term predictions $p(\boldsymbol{x}_1), \ldots, p(\boldsymbol{x}_{H-1})$ of the state evolution. To accomplish this we iteratively predict

$$\begin{aligned} p(\boldsymbol{x}_{t+1}|\boldsymbol{c}_t) = \iiint & p(\boldsymbol{x}_{t+1}|\boldsymbol{f}_t, \boldsymbol{x}_t) q(\boldsymbol{f}_t|\boldsymbol{x}_t, \boldsymbol{c}_t, \boldsymbol{h}_p) \\ & \times p(\boldsymbol{x}_t) q(\boldsymbol{h}_p) d\boldsymbol{f}_t d\boldsymbol{x}_t d\boldsymbol{h}_p, \end{aligned} \tag{14}$$

which we approximate as a Gaussian using GP moment matching [8, 3].[1] We first define the distribution

$$q(\tilde{\boldsymbol{x}}_t|\boldsymbol{c}_t) = \mathcal{N}(\tilde{\boldsymbol{x}}_t|\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}_t}, \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_t}) \tag{15}$$

over the concatenated state $\tilde{\boldsymbol{x}}_t = (\boldsymbol{x}_t, \boldsymbol{c}_t, \boldsymbol{h}_p)$, where the Gaussianity of $q(\tilde{\boldsymbol{x}}_t|\boldsymbol{c}_t)$ follows from the fact that $p(\boldsymbol{x}_t)$ and $q(\boldsymbol{h}_p)$ are both multivariate Gaussians and independent of each other [1]. The mean and covariance of $q(\tilde{\boldsymbol{x}}_t|\boldsymbol{c}_t)$ are

$$\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}_t} = [\boldsymbol{\mu}_{\boldsymbol{x}_t}, \boldsymbol{c}_t, \boldsymbol{n}_p] \tag{16}$$

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_t} = \text{blockdiagonal}([\boldsymbol{\Sigma}_{\boldsymbol{x}_t}, \boldsymbol{0}, \boldsymbol{T}_p]). \tag{17}$$

Next we predict the change in state given the control $\boldsymbol{c}_t$

$$p(\boldsymbol{y}_t|\boldsymbol{c}_t) = \iint p(\boldsymbol{y}_t|\boldsymbol{f}_t) q(\boldsymbol{f}_t|\tilde{\boldsymbol{x}}_t) q(\tilde{\boldsymbol{x}}_t|\boldsymbol{c}_t) d\boldsymbol{f}_t d\tilde{\boldsymbol{x}}_t \tag{18}$$

[1] Note that we integrate out the distribution of the state, the GP dynamics model and the latent variable encoding the task. The control has no distribution as it is treated as a free parameter within an MPC context.

which is intractable due to the integral over the state $q(\tilde{\boldsymbol{x}}_t|\boldsymbol{c}_t)$ through the GP. However, given that the state is a Gaussian and the kernel is the RBF kernel, we can analytically compute the first and second moments of the distribution $p(\boldsymbol{y}_t|\boldsymbol{c}_t)$ [8, 3, 2]. We then make a Gaussian approximation using those moments

$$q(\boldsymbol{y}_t|\boldsymbol{c}_t) = \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{\mu}_{\boldsymbol{y}_t}, \boldsymbol{\Sigma}_{\boldsymbol{y}_t}) \tag{19}$$

We are really interested in

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{c}_t) = \iint p(\boldsymbol{x}_{t+1}|\boldsymbol{y}_t, \boldsymbol{x}_t) q(\boldsymbol{y}_t|\boldsymbol{c}_t) p(\boldsymbol{x}_t|\boldsymbol{c}_t) d\boldsymbol{y}_t d\boldsymbol{x}_t \tag{20}$$

which gives

$$\mathbb{E}[\boldsymbol{x}_{t+1}] = \mathbb{E}[\boldsymbol{y}_t] + \boldsymbol{\mu}_{\boldsymbol{x}_t} = \boldsymbol{\mu}_{\boldsymbol{y}_t} + \boldsymbol{\mu}_{\boldsymbol{x}_t} \tag{21}$$

$$\mathrm{var}[\boldsymbol{x}_{t+1}] = \mathrm{var}[\boldsymbol{x}_t] + \mathrm{var}[\boldsymbol{y}_t] + \mathrm{cov}[\boldsymbol{x}_t, \boldsymbol{y}_t] + \mathrm{cov}[\boldsymbol{y}_t, \boldsymbol{x}_t] \tag{22}$$

where the computation of the covariance terms in (22) is detailed in [2]. By iterating this procedure for $t = 1, \ldots, H-1$, we obtain the desired sequence of approximate Gaussian state distributions $p(\boldsymbol{x}_1), \ldots, p(\boldsymbol{x}_{H-1})$.

References

[1] J. Blitzstein and J. Hwang. *Introduction to Probability*. CRC Press, 2015.

[2] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *PAMI*, 37(2):408–423, 2015.

[3] M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *ICML*, 2011.

[4] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *UAI*, 2013.

[5] M. Hoffman, D. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *JMLR*, 2013.

[6] S. Kamthe and M. P. Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In *AISTATS*, 2018.

[7] A. Matthews, J. Hensman, R. Turner, and G. Zoubin. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. In *AISTATS*, 2016.

[8] J. Quiñonero-Candela, A. Girard, J. Larsen, and C. E. Rasmussen. Propagation of uncertainty in Bayesian kernel models—application to multiple-step ahead forecasting. In *ICASSP*, 2003.

[9] M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *AISTATS*, 2009.