Meta Reinforcement Learning with Latent Variable Gaussian Processes

Steindór Sæmundsson
Department of Computing
Imperial College London
United Kingdom

Katja Hofmann
Microsoft Research
Cambridge
United Kingdom

Marc Peter Deisenroth
Department of Computing
Imperial College London
United Kingdom

Abstract

Learning from small data sets is critical in many practical applications where data collection is time consuming or expensive, e.g., robotics, animal experiments or drug design. Meta learning is one way to increase the data efficiency of learning algorithms by generalizing learned concepts from a set of training tasks to unseen, but related, tasks. Often, this relationship between tasks is hard coded or relies in some other way on human expertise. In this paper, we frame meta learning as a hierarchical latent variable model and infer the relationship between tasks automatically from data. We apply our framework in a model-based reinforcement learning setting and show that our meta-learning model effectively generalizes to novel tasks by identifying how new tasks relate to prior ones from minimal data. This results in up to a 60% reduction in the average interaction time needed to solve tasks compared to strong baselines.

1 INTRODUCTION

Reinforcement learning (RL) is a principled mathematical framework for learning optimal controllers from trial and error [28]. However, RL traditionally suffers from data inefficiency, i.e., many trials are needed to learn to solve a specific task. This can be a problem when learners operate in real-world environments where experiments can be time consuming (e.g., where experiments cannot run faster than real time) or expensive. For example, in a robot learning setting, it is impractical to conduct hundreds of thousands of experiments with a single robot because we will have to wait for a long time and the wear and tear on the hardware can cause damage.

There are various ways to address data-efficiency in RL. Model-based RL, where predictive models of the transition function are learned from data, can be used to reduce the number of experiments in the real world. The learned model serves as an emulator of the real world. A challenge with these learned models is the problem of model errors: If we learn a policy based on an incorrect model, the policy is unlikely to succeed on the real task. To mitigate the issue of these model errors it is recommended to use probabilistic models and to take model uncertainty explicitly into account during planning [27, 10]. This approach has been applied successfully to simulated and real-world RL problems [9], where a policy-search approach was used to learn optimal policy parameters. Robustness to model errors and, thereby, increased data efficiency, can be achieved by using model predictive control (MPC) instead of policy search since MPC allows for online updates of the model, whereas policy search would update the model only after a trial [17].

If we are interested in solving a set of related tasks we can use meta learning as an orthogonal approach to increase data efficiency. Generally, the aim of meta learning is to train a model on a set of training tasks and then generalize to new tasks using minimal additional data [12]. The strength of meta learning is to transfer learned knowledge to related situations. For example, we may want to control multiple robot arms with slightly different specifications (e.g., link weights or lengths) or different operating environments (e.g., underwater, in low gravity). Normally, learned controllers deal with a single task. In a robotics context, solutions for multiple related tasks are often desired, e.g., for grasping multiple objects [22] or in robot games, such as robot table tennis [24] or soccer [3]. Much of the literature on meta and transfer learning in RL has focused on multi-task learning, i.e., cases where the system/robot is the same, but the task changes [16, 29, 3, 5, 20, 21, 24, 8, 12]. Although meta learning given multiple or non-stationary dynamics has also been considered in [11, 18, 4, 1].

We adopt a meta learning [26, 32, 12] perspective on the problem of using knowledge from prior tasks for more efficient learning of new ones. We take a probabilistic view and propose to transfer knowledge within a model-based RL setting using a latent variable model. We focus on settings where system specifications differ, but where the task objective is identical. We treat system specifications as a latent variable, and infer these unobserved factors and their effects online. To address the issue of meta learning within the context of data-efficient RL, we propose to learn predictive dynamics models conditioned on the latent variable and to learn controllers using these models. We use Gaussian processes (GPs) [25] to model the dynamics, and MPC for policy learning. To obtain a posterior distribution on the latent variable, we use variational inference. The posterior can be updated online as we observe more and more data, e.g., during the execution of a control strategy. Hence, we systematically combine three orthogonal ideas (probabilistic models, MPC, meta learning) for increased data efficiency in settings where we need to solve different, but related tasks.

2 MODEL-BASED RL

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 $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{E})$, where $\boldsymbol{E} = \text{diag}(\sigma_1^2, \dots, \sigma_D^2)$. For model-based RL we first aim to learn the unknown transition function f. In this context, [27, 10] highlighted that probabilistic models of f are essential for data-efficient learning as they mitigate the effect of model errors. Therefore, we learn the dynamics of the system using a GP.

GP Dynamics A GP is a probabilistic, non-parametric model and can be interpreted as a distribution over functions. A GP is defined as an infinite collection of random variables $\{f_1, f_2, \dots\}$, any finite number of which are jointly Gaussian distributed [25]. A GP is fully specified by a mean function m and a covariance function (kernel) k, which allows us to encode high-level structural assumptions on the underlying function such as smoothness or periodicity. We denote an unknown function f that is modeled by a GP by $f \sim GP(m(\cdot), k(\cdot, \cdot))$. We use the squared exponential (RBF) covariance function

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\left(-\tfrac{1}{2}(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \mathbf{L}^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j)\right) \tag{2}$$

where σ_f^2 is the signal variance and $\boldsymbol{L}$ is a diagonal matrix of squared length-scales.

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

$$J = \mathbb{E}\left[\sum_{t=1}^{H} \ell(\boldsymbol{x}_t)\right], \tag{3}$$

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 $\boldsymbol{c}_0^*, \dots, \boldsymbol{c}_{H-1}^*$, we compute the expected long-term cost J in (3) using Gaussian approximations $p(\boldsymbol{x}_1), \dots, p(\boldsymbol{x}_H)$ for a given control sequence $\boldsymbol{c}_0, \dots, \boldsymbol{c}_{H-1}$. The computation of the expected long-term cost is detailed in the supplementary material. 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 $\boldsymbol{c}_0^*, \dots, \boldsymbol{c}_{H-1}^*$ 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 [17].

3 MODEL-BASED META RL

We assume a setting with a potentially infinite number of dynamical systems that are of the same type but with different specifications (e.g., multiple robotic arms with links of differing lengths and weight). More formally, we assume a distribution over dynamical systems with samples $f_p \sim p(f)$ indexed by $p = 1..P$. Each sample f_p is a dynamical system of the form (1) with states $\boldsymbol{x} \in \mathbb{R}^D$ and control signals $\boldsymbol{c} \in \mathbb{R}^K$. Instead of learning individual predictive models for each dynamical system from scratch, we look to meta learning as an approach to learning new dynamics more data efficiently by leveraging shared structure in the dynamics.

Meta Learning Generally, meta learning aims to learn new tasks with minimal data and/or computation using knowledge or inductive biases learned from prior tasks [12]. Here we require our model to accomplish two things simultaneously:

1. **Multi-Task Learning**: Disentangle global and task-specific properties of the different dynamics such that it can solve multiple tasks.
2. **Transfer Learning**: Use global properties to generalize predictive performance to novel dynamics.

We propose to address this meta-learning challenge in a probabilistic way: 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. Each sample f_p from the distribution is modeled as

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{c}_t, \boldsymbol{h}_p) + \boldsymbol{\epsilon}\,, \tag{4}$$

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.

Training and Evaluation Training corresponds only to the *multi-task learning* aspect of our meta learning approach. We aim to learn the global function f and the latent embeddings $\boldsymbol{h}_p$ given trajectory observations from a set of training systems. For evaluation at test time, we use inference to obtain a distribution over a set of latent variables $\boldsymbol{h}_*$ for each test system. Since our objective is to improve data efficiency in an RL setting, we consider two related but distinct measures of performance. One corresponds to the transfer learning aspect of our approach, where we infer only the latent test embeddings without updating the global model f. We refer to this as the *single-shot performance*. The other measure we use is the additional data required to successfully solve a RL task: *few-shot learning*. In this case, the global model f is updated with new additional data, thus combining both the multi-task and transfer learning aspects.

The meta RL procedures for training and testing are detailed in algorithms 1 and 2, respectively.

3.1 META-LEARNING MODEL

Our meta-learning model is a GP prior on the unknown transition function in (4) 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 $m(\tilde{\boldsymbol{x}}_t) = \mathbf{0}$, which encodes that a priori the state does not change [9]. Each dimension of the targets $\boldsymbol{y}$ is modeled by an independent GP. We use a Gaussian likelihood

$$p(\boldsymbol{y}_t|\tilde{\boldsymbol{x}}_t, \boldsymbol{f}(\cdot), \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{f}(\tilde{\boldsymbol{x}}_t), \boldsymbol{E}), \tag{5}$$

where $\boldsymbol{\theta} = \{\boldsymbol{E}, \boldsymbol{L}, \sigma_f^2, Q\}$ are the model hyperparameters and $\boldsymbol{f}(\cdot) = \left(f^1(\cdot), \ldots, f^D(\cdot)\right)$ denotes a multi-dimensional function. We place a standard-normal prior $\boldsymbol{h}_p \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ on the latent variables $\boldsymbol{h}_p$. The full

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 (3)
      —**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)

specification of the model is

$$p(\boldsymbol{Y}, \boldsymbol{H}, \boldsymbol{f}(\cdot)|\boldsymbol{X}, \boldsymbol{C}) \tag{6}$$
$$= \prod_{p=1}^{P} p(\boldsymbol{h}_p) \prod_{t=1}^{T_p} p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{c}_t, \quad \boldsymbol{h}_p, \boldsymbol{f}(\cdot)) p(\boldsymbol{f}(\cdot))$$

where we denote a collection of vectors in bold uppercase 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.

Model Properties Our meta-learning GP (ML-GP) model exhibits three important properties:

1. The latent variable encodes a distribution over plausible systems and is inferred from data

2. Conditioning the GP on the latent variable enables it to disentangle global and task specific variation in the dynamics. Generalization to new dynamics is done by inferring the latent variable of that system.

3. The latent variable is fixed within system trajectories so that inference can be performed online (e.g. while executing a controller).

Fig. 2 illustrates these properties on a toy example.

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 (3)
    —**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 (3)
      —**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)

3.2 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 (6). 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

$$Q(\boldsymbol{f}(\cdot), \boldsymbol{H}) = q(\boldsymbol{f}(\cdot))q(\boldsymbol{H}) \tag{7}$$

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

$$\mathcal{L} = \mathbb{E}_{Q(\boldsymbol{f}(\cdot),\boldsymbol{H})}\Big[\log \frac{p(\boldsymbol{Y}, \boldsymbol{H}, \boldsymbol{f}(\cdot)|\boldsymbol{X}, \boldsymbol{C})}{Q(\boldsymbol{f}(\cdot), \boldsymbol{H})}\Big], \tag{8}$$

which lower-bounds the log-marginal likelihood [15]. We parameterize our variational distribution such that we can compute the lower bound in (8). 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

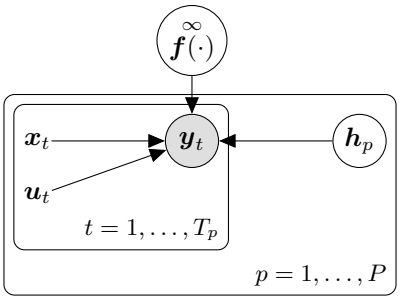

Figure 1: Graphical model for our ML-GP model.

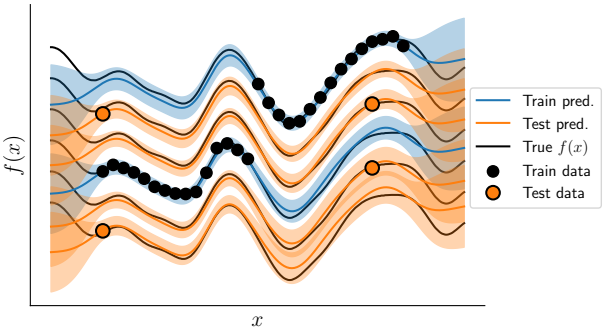

Figure 2: The figure shows six unknown tasks (toy examples) with a shared structure (the same function) and task specific variation (fixed offset). The ML-GP model is able to disentangle the two automatically given the training data (black discs) as demonstrated by the training prediction curves. It also infers a reasonable value for the offset given a single observations from unseen test tasks (orange discs) and can use the global structure to generalize predictive performance on those tasks.

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 [30] 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 [14] 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

$$q(f^d(\cdot)) = \int p(f^d(\cdot)|\boldsymbol{u}^d)q(\boldsymbol{u}^d)d\boldsymbol{u}^d. \tag{9}$$

where $q(\boldsymbol{u}^d) = \mathcal{N}(\boldsymbol{u}^d|\boldsymbol{m}^d, \boldsymbol{S}^d)$ is a full rank Gaussian. The integral in (9) can be computed in closed form since both terms are Gaussian, resulting in a GP with mean and

covariance functions given by

$$m_q(\cdot) = \boldsymbol{k}_Z^T(\cdot)\boldsymbol{K}_{ZZ}^{-1}\boldsymbol{m}^d \tag{10}$$

$$k_q(\cdot,\cdot) = k(\cdot,\cdot) - \boldsymbol{k}_Z^T(\cdot)\boldsymbol{K}_{ZZ}^{-1}(\boldsymbol{K}_{ZZ} - \boldsymbol{S}^d)\boldsymbol{K}_{ZZ}^{-1}\boldsymbol{k}_Z(\cdot) \tag{11}$$

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

$$q(\boldsymbol{H}) = \prod_{p=1}^{P} \mathcal{N}(\boldsymbol{h}_p|\boldsymbol{n}_p, \boldsymbol{T}_p) \tag{12}$$

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 (8).

Evidence Lower Bound (ELBO) The ELBO can be shown to decompose into (see supplementary material)

$$\begin{aligned}\mathcal{L} = &\sum_{p=1}^{P}\sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{f}_t|\boldsymbol{x}_t,\boldsymbol{c}_t)}\big[\log p(\boldsymbol{y}_t|\boldsymbol{f}_t)\big] \\ &- \mathrm{KL}\big[q(\boldsymbol{H})||p(\boldsymbol{H})\big] - \mathrm{KL}\big[q(\boldsymbol{U})||p(\boldsymbol{U})\big]\end{aligned} \tag{13}$$

where the expectation is taken with respect to

$$q(\boldsymbol{f}_t|\boldsymbol{x}_t, \boldsymbol{c}_t) = \int q(\boldsymbol{f}_t|\boldsymbol{x}_t, \boldsymbol{c}_t, \boldsymbol{h}_p) q(\boldsymbol{h}_p) d\boldsymbol{h}_p. \tag{14}$$

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 integral in (14) is intractable due to the non-linear dependence on $\boldsymbol{h}_p$ in (10) and (11). 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.

Training For the update steps in algorithms 1 and 2 we jointly optimize the GP hyperparameters $\boldsymbol{\theta}$ and the variational parameters $\boldsymbol{\phi} = \{\boldsymbol{Z}, M\,\{\boldsymbol{m}^d, \boldsymbol{S}^d\}_{d=1}^D, \{\boldsymbol{n}_p, \boldsymbol{T}_p\}_{p=1}^P\}$ 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. $\boldsymbol{\phi}_{\boldsymbol{h}} = \{\boldsymbol{n}_p, \boldsymbol{T}_p\}_{p=1}^P$.

In practice, we use a single sample $\boldsymbol{h}_p \sim q(\boldsymbol{h}_p)$ drawn from the variational distribution for each system. We use stochastic mini-batch training, sampling a small number of trajectories and their associated latent variable at a time. Empirically, we found standardizing the input states and controls $(\boldsymbol{x}, \boldsymbol{c})$ and outputs $(\boldsymbol{y})$ crucial for successful training of the model. For optimization we used Adam [19] with default hyperparameters: $\alpha = 1 \times 10^{-2}, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$.

4 EXPERIMENTS

Our experiments focus on evaluating our proposed model in terms of predictive performance, the nature of the latent embeddings and data efficiency. We address the following questions: "Does conditioning the GP on the latent variable allow us to disentangle system specific and global properties of the observations? Does this improve predictive performance in the transfer learning setting?" (Section 4.1). "Is the latent system embedding the model learns a sensible one? (Section 4.2) "Does the application of our ML-GP in model-based RL lead to data-efficient learning across tasks" (Section 4.3).

As a baseline model we use a sparse GP (SGP) [30] as described in Section 3.2 but without the latent variable that explicitly represents the task. For assessing the model quality (Section 4.1) we additionally evaluate the performance of a standard GP with no sparse approximation. We use the following nonlinear dynamical systems to perform our experiments:

Cart-pole swing-up The cart-pole system consists of a cart that moves horizontally on a track with a freely swinging pendulum attached to it. The state of this nonlinear system is the position x and velocity $\dot{x}$ of the cart and the angle θ and angular velocity $\dot{\theta}$ of the pendulum. The control signals act as a horizontal force on the cart limited to the range $c \in [-15, 15]\,\mathrm{N}$. The mean of the initial state distribution is the state where the pendulum is hanging downward. The task is to learn to swing up and balance the pendulum in the inverted position in the middle of the track.

Double-pendulum swing-up The double-pendulum system is a two-link robotic arm with two motors, one in the shoulder and one in the elbow. The state of the system comprises the angles θ_1, θ_2 and angular velocities $\dot{\theta}_1, \dot{\theta}_2$ of the inner and outer pendulums, respectively. The control signals are the torques $c_{1,2} \in [-4, 4]\,\mathrm{Nm}$ applied to the two motors. The mean of $p(\boldsymbol{x}_0)$ is the position where both pendulums are hanging downward. The goal is to find a control strategy that swings the double pendulum up and balances it in the inverted position.

4.1 QUALITY OF MODEL LEARNING

In the first set of experiments, we investigate if the latent variable of the ML-GP improves prediction performance on unseen systems compared to the SGP baseline. To assess the effect of the sparse approximation we also include a standard GP baseline (no sparse approximation) in this section. To test the prediction quality, we execute the same fixed control signals[1] on six settings of the cart-pole task to generate one 100-step (10 s) trajectory per training task. The specifications of the training tasks were all combinations (m, l) of $m \in \{0.4, 0.6, 0.8\}$, $l \in \{0.5, 0.7\}$ where m and l denote the mass and length of the pendulum, respectively. Thus, the total number of data points for our six training tasks is $T = 600$ amounting to 60 s of interaction time.

For evaluation we use the same sequence of control signals we used for training and compute the one-step prediction quality in terms of root mean squared error (RMSE) and negative log likelihood (NLL) on a set of test tasks. We use 14 held-out test tasks specified as $m \in \{0.4, 0.6, 0.7, 0.8, 0.9\}$, $l \in \{0.4, 0.5, 0.6, 0.7\}$, excluding the (m, l)-combinations of the training tasks.

During evaluation, we observe 10 time steps from an unseen trajectory based on which we infer the latent task $\boldsymbol{h}_p$ using variational inference for the ML-GP while leaving the model hyperparameters and other variational parameters fixed. We then predict the next 90 steps using the ML-GP, SGP and GP models. ML-GP also performs online inference of the latent variable after each step. We repeat this experiment with 10 different seeds that determine the initial state, and average the results.

Fig. 3 shows the RMSE and NLL for all 3 models. The ML-GP clearly outperforms both the SGP and GP baselines in terms of both the accuracy of its mean predictions (as evident by the RMSE) as well as capturing the data better under its predictive distribution as seen by the NLL. The NLL accounts for both the mean prediction as well as the uncertainty of the model about the prediction. Both baselines have comparable RMSEs to each other with enough inducing points but generalize poorly on new tasks with overconfident predictions. Fig. 4 illustrates this behavior.

The baselines fail to generalize since they have no observations from the system with this configuration. The ML-GP generalizes from training to new test tasks naturally because it explicitly incorporates the latent variables encoding the system configuration.

[1] the control signals were manually chosen as ones that solved a configuration not included in either the training or test set.

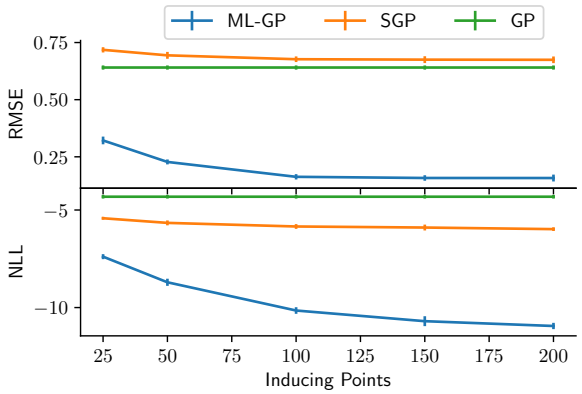

Figure 3: Mean and two standard deviation confidence error-bars of the RMSE and NLL for the ML-GP, SGP and the standard GP model as a function of the number of inducing points. The ML-GP significantly outperforms both baselines.

4.2 LATENT EMBEDDING

In order for our model to perform well in meta learning settings, the latent variables $\boldsymbol{h}_p$ need to reflect a *sensible* embedding. By sensible we mean it should take on a particular structure: a) locally similar values in the latent space should correspond to similar task specifications and b) moving in latent space should correspond to coherent transitions in task specifications.

Fig. 5 shows an example of an inferred latent embedding of both training and test tasks after the training procedure outlined above. The test-task latent variables are inferred from 10 observations from the held-out systems.

The different colors of the discs denote the four different settings of lengths whereas the colors of the dotted lines connecting the discs denote the five different settings of mass. The figure plots the mean of each $q(\boldsymbol{h}_p)$ with two standard deviation error bars in each dimension. The embedding displays an intuitive structure where changes in length or mass are disentangled (denoted by the black arrows) into a length-mass coordinate system with the expected transitive properties, e.g. the lengths are ordered as blue ($l = 0.4$), green ($l = 0.5$), red ($l = 0.6$) and orange ($l = 0.7$). The uncertainty estimates also exhibit qualitatively the intuitive property of being less uncertain about tasks which are similar to (closer to) the training tasks, e.g. comparing the red and blue tasks in fig. 5.

4.3 DATA-EFFICIENT RL

Our second set of experiments investigates the performance of the ML-GP model in terms of data efficiency in RL settings. Specifically, we look at whether our meta learning approach is a) at least as efficient at solving a

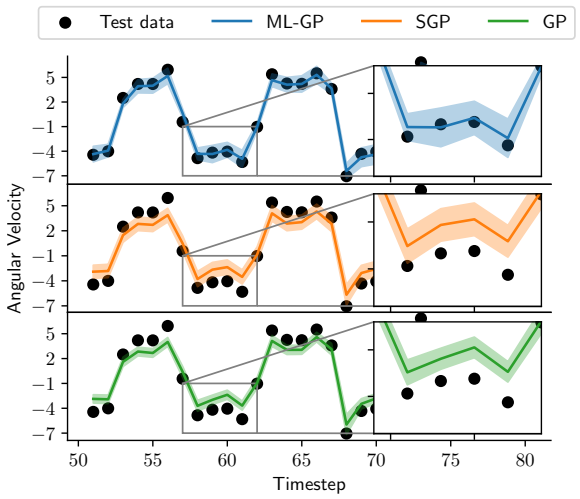

Figure 4: One-step predictions of the angular velocity in cart-pole. The figure shows the true data points (discs) and the predictive distributions with a two standard deviation confidence interval for the ML-GP, SGP and a standard GP. The ML-GP generalizes well to new tasks; both the SGP and GP baselines are overly confident.

set of training tasks, b) more efficient at solving subsequent test tasks, when compared to a non-meta learning baseline and c) whether the ML-GP model improves performance when compared to the SGP model trained with the meta learning approach.

We first learn a model of the dynamics (4), which we then use to learn a policy to control the system. For policy learning we use MPC, minimizing the cost in (3) with a moving horizon to learn an optimal sequence of control signals. We assume we have a set of training systems and evaluate the performance of the models using some held-out test systems with novel configurations (tasks).

We run experiments on both the cart-pole swing-up task and the double-pendulum swing-up task. In both scenarios, we use a sampling frequency of 10 Hz, episodes of 30 steps (3 s) and a planning horizon of 10 steps. For the cart-pole swing-up, solving the task means the pendulum is balanced closer than 8 cm from the goal position for at least the last 10 steps. For the double-pendulum swing-up, it means the outer pendulum is balanced closer than 22 cm for at least the last 10 steps.

At meta-training or test-time, a pass through the training-/test-set means executing the MPC policy learning algorithm on each of the unsolved task in that set. Each execution constitutes a trial for that task. The sets are traversed until all the tasks are solved or all unsolved tasks have executed 15 trials. The training and test procedures are detailed in algorithms 1 and 2 in section 3. All results are averaged over 20 independent random initializations.

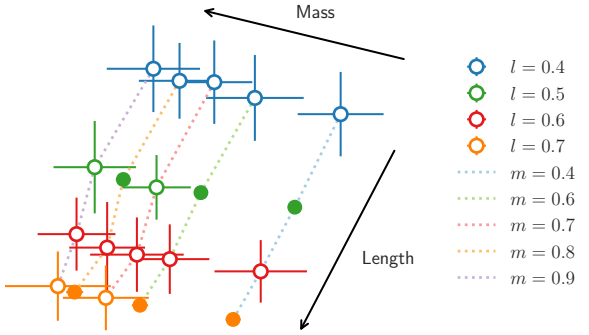

Figure 5: Latent space embedding of cart-pole configurations/tasks. The figure shows the mean (discs) of the inferred latent variables and two standard deviation error bars. Filled discs are training tasks and empty discs are held out test tasks. The colors of the discs represent the length and the colors of the dotted lines between discs represent the mass.

Note that we execute on all (unsolved) tasks before retraining the dynamics model as detailed in section 3. This means that the model is updated with 3 s worth of experience for every task in that pass at a time. On the other hand, the model does not take advantage of additional prior experience until it has completed a pass.

For comparison with the ML-GP model, we use the SGP model trained in two different ways. To establish a lower-bound baseline, we run the model-based RL approach where we train a separate model for each task on both the training and test sets. After each training task we additionally attempt to solve each of the test tasks to evaluate single-shot performance where we report the mean across the training tasks as the single shot success rate. We refer to this baseline as SGP-I which is a sparse variant of the approach in [17] that achieves state-of-the-art in data efficiency. Secondly, we train a single SGP model on all the training tasks simultaneously using the same training approach as we do for ML-GP. We refer to this baseline as SGP-ML.

Cart-pole swing-up We train the models on six specifications of the cart-pole dynamics, with $m \in \{0.4, 0.6, 0.8\}$, $l \in \{0.6, 0.8\}$ and evaluate its performance on a set of four test tasks chosen as $m \in \{0.7, 0.9\}$, $l = \{0.5, 0.7\}$. We choose these settings to examine the performance on both interpolation and extrapolation for differing lengths and masses. We choose the squared distance between the tip of the pendulum and goal position (with the pendulum balanced straight in the middle of the track) as the cost. Fig. 6 shows the mean success rate (over initializations and the four test tasks) of ML-GP, SGP-I and SGP-ML against the number of trials executed on the systems. We observe that

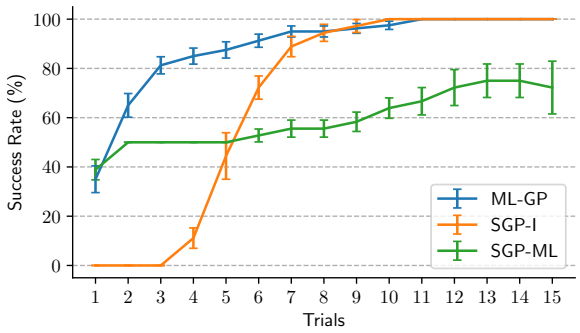

Figure 6: Mean success rate over initializations and the four test tasks for the cart-pole system after training on six tasks. The graph compares ML-GP with SGP-I (trained independently) and SGP-ML (trained on all tasks).

both the ML-GP model and the SGP-ML display generalization to new tasks as evident by the success rate in the first trial (see also Table 1). However, whereas the ML-GP quickly improves with more observations in subsequent trials, the SGP-ML model struggles to solve the remaining tasks. We attribute this failure to the inability of the SGP-ML model to explain variation in the dynamics caused by differences in system specifications.

When comparing with independent training of each system we see that the ML-GP compares favorably, reaching 80% success rate after only three trials and 90% after six trials compared to the SGP-I, which reaches 80% after 7 trials and 90% after 8 trials. We further analyze performance of ML-GP to identify the tasks that were additionally solved between trials 3 and 6. We find that this is due to a consistently challenging system with $m = 0.9, l = 0.5$, which requires the learner to extrapolate beyond the range of values seen during training. The mean number of trials required to solve this task is 4.3 ± 0.6, compared to the task mean of 2.7 ± 0.2 trials. Table 1 shows the mean total time required to solve

Table 1: Mean time spent solving the cart-pole system and the single-shot success rate.

MODEL	TRAIN (s)	TEST (s)	SINGLE SHOT
SGP-I	16.1 ± 0.4	17.5 ± 0.4	0.08 ± 0.01
SGP-ML	23.7 ± 1.4	20.8 ± 1.2	$\mathbf{0.38 \pm 0.04}$
ML-GP	$\mathbf{15.1 \pm 0.5}$	$\mathbf{8.1 \pm 0.6}$	0.35 ± 0.05

the training and test tasks. On average, ML-GP needs less than half the amount of time to solve the test tasks compared to individually training on the tasks (SGP-I). We also see an improvement in the total training time, which suggests that ML-GP derives some transfer benefit during training despite training on the systems on a concurrent trial basis, i.e. we do not update the model until all systems have executed a given trial. Compared to the SGP-ML, the ML-GP model can maintain an accurate model while learning multiple systems and quickly adapts to new dynamics, whereas the performance of SGP-ML stagnates as reflected in the interaction time on both the training and test systems.

Double-pendulum swing-up We repeat the same experimental set-up on the double-pendulum task. We trained on six systems with $m_1 \in \{0.5, 0.7\}$, $l_1 \in \{0.4, 0.5, 0.7\}$ and evaluate on a set of four test tasks chosen as $m_1 \in \{0.6, 0.8\}$, $l_1 = \{0.6, 0.8\}$, where m_1, l_1 are the mass and length of the inner pendulum. The cost is the squared distance between the tip of the outer pendulum and the goal position (with both pendulums standing straight up). Fig. 7 plots the mean success rate against the number of trials executed on the system. Comparing the ML-GP model to the SGP-ML we observe comparable single-shot performance and a qualitatively similar learning curve for the test tasks. However, the ML-GP reaches 90% success rate about four trials before the SGP-ML, around trial nine, i.e. meta learning achieves a significantly higher data efficiency. Compared to independent training of the tasks using SGP-I, the ML-GP leads to significantly less (new) training data needed to solve the tasks. Table 2 reports the mean total

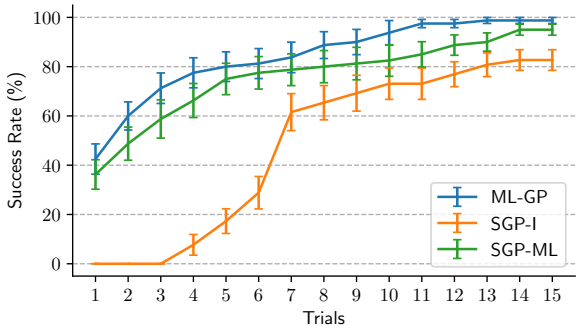

Figure 7: Mean success rate over initializations and the four test tasks for the double pendulum after training on six tasks. The graph compares the ML-GP against the SGP-I (trained independently on each task) and the SGP-ML (trained using the meta learning procedure).

time required to solve the tasks. Compared to the SGP-ML, the performance of the two is similar, although arguably the ML-GP compares favorably in terms of average time needed to solve the test tasks. Compared to the SGP-I, we see improvement during training as well as at test time. The average time needed for ML-GP to solve the test environments is reduced to around 40% to that of the SGP-I.

Table 2: Mean time spent solving the double-pendulum system and the single-shot success rate.

MODEL	TRAIN (s)	TEST (s)	SINGLE SHOT
SGP-I	18.9 ± 0.7	25.9 ± 1.5	0.07 ± 0.01
SGP-ML	17.9 ± 1.3	13.7 ± 2.2	0.36 ± 0.06
ML-GP	$\mathbf{16.6 \pm 1.1}$	$\mathbf{10.2 \pm 1.6}$	$\mathbf{0.43 \pm 0.06}$

5 RELATED WORK

Meta learning has long been proposed as a form of learning that would allow systems to systematically build up and re-use knowledge across different but related tasks [26, 32]. MAML is a recent promising model free meta learning approach that learns a set of model parameters that are used to rapidly learn novel tasks [12]. Another interpretation of MAML is formulated in [13], which shares our hierarchical Bayesian formulation of the meta learning problem. However, the model-free setting in which MAML has been applied so far typically require orders of magnitude more training data than the model-based approaches we build up on in the present work.

Our ML-GP model resembles the GP latent variable model (GPLVM), which is typically used in unsupervised settings [23]. In the GPLVM, the GP is used to map a low-dimensional latent embedding to higher-dimensional observations. A Bayesian extension (BGPLVM) was introduced in [31] where inference over the latent variable is performed using variational inference. To enable minibatch training, and unlike BGPLVM, we take the approach of [14] and do not marginalize out the inducing variables. The main difference of our model and the GPLVM is that we learn a mapping from both observed and latent inputs to observations.

The combination of observed and latent inputs was investigated in [33] where the authors use Metropolis sampling for inference which does not scale to larger datasets. A similar setup is found in [7] where the model is used for partially observed input data. The work also proposes uses in autoregressive settings similar to ours. Different from us, the distribution over inducing variables is analytically optimized, making minibatch training infeasible.

A related and complimentary line of research are multi-output GPs (MOGPs) [2]. Recently, [6] proposed a latent variable extension to MOGPs (LVMOGP) which is similar to our ML-GP, particularly in their missing data formulation of the model. The crucial difference from our work is that we augment the input space by concatenating the latent variable to the input space while the LVMOGP uses the Kronecker product of two separate kernels applied on the latent and input spaces respectively. Notably, the two models are equivalent for kernels that naturally decompose as a Kronecker product (e.g. the RBF) but depart from there.

A similar framework to ours is found in [11], called hidden parameter Markov decision processes (HiP-MDP), which parametrizes a family of related dynamics through a low dimensional latent embedding. The HiP-MDP assumes a fixed latent variable within trajectories. Different from us, the authors use an infinite mixture of GP basis functions where the task specific variation is obtained through the weights of the basis functions [11]. This work was extended in [18], replacing the GP basis functions with a Bayesian neural network. This enables non-linear interactions between the latent and addresses scalability. In this work, the interactions between latent and state variables are obtained through the non-linear RBF kernel, and the scalability is addressed through the variational sparse approach.

In [8], an RL setting is considered that is closely related to our meta-learning set-up. The authors use a parametric policy that depends on a known deterministic task variable and augment the policy function to include it as well. In [8], the authors consider the same dynamical system but solve different tasks by augmenting the policy with a task variable. In our work, we look at different settings of the dynamics but the task remains the same. We show how to generalize to the setting where task variables are latent and inferred from interaction data. This dramatically extends applicability in real-world settings.

6 CONCLUSION

We proposed a meta learning approach within the context of model-based RL that allows us to transfer knowledge from training configurations of robotic systems to unseen test configurations. The key idea behind our approach is to address the meta learning problem probabilistically using a latent variable model. We use online variational inference to obtain a posterior distribution over the latent variable, which describes the relatedness of tasks. This posterior is then used for long-term predictions of the state evolution and controller learning within a model-based RL setting. We demonstrated that our ML-GP approach is as efficient or better than a non-meta learning baseline when solving multiple tasks at once. The ML-GP further generalizes well to learning models and controllers for unseen tasks giving rise to substantial improvements in data-efficiency on novel tasks.

Acknowledgements

This work was supported by Microsoft Research through its PhD Scholarship Programme.

References

[1] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations (ICLR)*, 2018.

[2] M. A. Àlvarez, L. Rosasco, and N. D. Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning (FTML)*, 4(3):195, 2012.

[3] S. Barrett, M. Taylor, and P. Stone. Transfer learning for reinforcement learning on a physical robot. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2010.

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

[5] B. da Silva, G. Konidaris, and A. Barto. Learning parametrized skills. In *International Conference on Machine Learning (ICML)*, 2012.

[6] Z. Dai, M. A. Álvarez, and N. D. Lawrence. Efficient modeling of latent information in supervised learning using Gaussian processes. In *Neural Information Processing Systems (NIPS)*. 2017.

[7] A. Damianou and N. D. Lawrence. Semi-described and semi-supervised learning with Gaussian processes. *Uncertainty in Artificial Intelligence (UAI)*, 2015.

[8] M. P. Deisenroth, P. Englert, J. Peters, and D. Fox. Multi-task policy search for robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

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

[10] M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML)*, 2011.

[11] F. Doshi-Velez and G. Konidaris. Hidden parameter Markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

[12] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

[13] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths. Recasting gradient-based meta-learning as hierarchical Bayes. In *International Conference on Learning Representations (ICLR)*, 2018.

[14] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence (UAI)*, 2013.

[15] M. Hoffman, D. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research (JMLR)*, pages 1303–1347, 2013.

[16] A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *Neural Information Processing Systems (NIPS)*, 2002.

[17] S. Kamthe and M. P. Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

[18] T. Killian, S. Daulton, G. Konidaris, and F. Doshi-Velez. Robust and efficient transfer learning with hidden parameter Markov decision processes. In *Neural Information Processing Systems (NIPS)*, Long Beach, CA, 2017.

[19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[20] J. Kober, E. Otzop, and J. Peters. Reinforcement learning to adjust robot movements to new situations. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[21] G. Konidaris, I. Scheidwasser, and A. Barto. Transfer in reinforcement learning via shared features. *Journal of Machine Learning Research (JMLR)*, 13:1333–1371, 2012.

[22] O. Kroemer, R. Detry, J. Piater, and J. Peters. Combining active learning and reactive control for robot grasping. *Robotics and Autonomous Systems (RAS)*, 58:1105–1116, 2010.

[23] N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Neural Information Processing Systems (NIPS)*. 2004.

[24] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *International Journal of Robotics Research (IJRR)*, 2013.

[25] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[26] T. Schaul and J. Schmidhuber. Metalearning. *Scholarpedia*, 5(6):4650, 2010.

[27] J. G. Schneider. Exploiting model uncertainty estimates for safe dynamic control learning. In *Neural Information Processing Systems (NIPS)*. 1997.

[28] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[29] M. Taylor and P. Stone. Cross-domain transfer for reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2007.

[30] M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

[31] M. Titsias and N. D. Lawrence. Bayesian Gaussian process latent variable model. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[32] R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review (AI Review)*, 18(2):77–95, 2002.

[33] C. Wang and R. M. Neal. Gaussian Process Regression with Heteroscedastic or Non-Gaussian Residuals. *ArXiv e-prints*, Dec. 2012.