Mode-Constrained Model-Based Reinforcement Learning via Gaussian Processes

Anonymous Author
Anonymous Institution

Abstract

Model-Based Reinforcement Learning (MBRL) algorithms do not typically consider environments – subject to multiple dynamics modes – where it is beneficial to avoid *inoperable or undesirable* dynamics modes. We present a MBRL algorithm that avoids entering such *inoperable or undesirable* dynamics modes, by constraining the controlled system to remain in a single dynamics mode with high probability. This is a particularly difficult problem because the mode constraint is *unknown a priori*. We propose to jointly infer the mode constraint, along with the underlying dynamics modes. Importantly, our method infers latent structure that our planning scheme leverages to 1) enforce the mode constraint with high probability, and to 2) target exploration where the mode constraint's *epistemic uncertainty* is high. We validate our method by showing that it can navigate a simulated quadcopter – subject to a turbulent dynamics mode – to a target state, whilst remaining in the desired dynamics mode with high probability.

1 INTRODUCTION

Over the last decade, Reinforcement Learning (RL) has become a popular paradigm for controlling dynamical systems (Hewing et al., 2020; Sutton and Barto, 2018). However, RL algorithms do not typically prevent agents from entering *inoperable or undesirable* dynamics modes. This would be desirable when flying a quadcopter to a target state whilst avoiding turbulent dynamics modes, or driving a car whilst avoiding dangerous road surfaces. In these examples, we seek agents that can learn sample-efficiently, whilst avoiding these *inoperable or undesirable* dynamics modes.

Preliminary work. Under review by AISTATS 2023. Do not distribute.

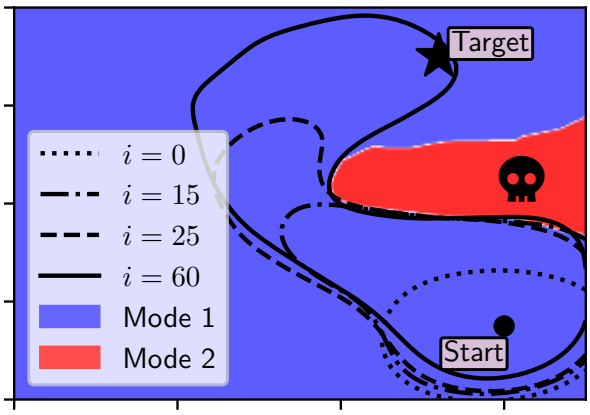

Figure 1: **Mode-constrained quadcopter navigation**. Diagram shows a top-down view of a quadcopter subject to 1) an *operable* dynamics mode (blue) and 2) an *inoperable*, turbulent dynamics mode induced by a strong wind field (red). The goal is to navigate to the target whilst remaining in the *operable* dynamics mode (blue). We achieve this by gradually expanding the δ-mode-constrained region (Definition 3.2) at each episode i, by updating our learned dynamics model.

One approach to solving this problem is to constrain the agent to a single dynamics mode during learning. However, in this mode-constrained setting, the agent does not directly observe the constraint. Instead, the mode constraint depends on the dynamics modes, which are *unknown a priori*. As a result, the mode constraint must be simultaneously inferred and enforced during learning.

When simultaneously enforcing and learning a constraint in RL, it is impossible to guarantee constraint satisfaction. This emphasises the need to learn sample efficiently, as each interaction with the environment could result in a constraint violation. Further to this, constraints can prevent an agent from solving the main task (Roy et al., 2022). This is because they can introduce local optima that make the space of feasible policies hard to navigate.

In this paper, we present ModeRL, a Bayesian Model-Based Reinforcement Learning (MBRL) algorithm, which simultaneously learns and enforces the mode constraint using well-calibrated uncertainty estimates from a learned

dynamics model. Our main contributions are as follows:

1. A method for jointly inferring the mode constraint alongside the underlying dynamics modes.
2. A planning algorithm that leverages the dynamics model's well-calibrated uncertainty estimates to 1) enforce the mode constraint up to a given probability, 2) combat local optima induced by the mode constraint, and 3) improve sample efficiency via a non-myopic exploration strategy.
3. We validate ModeRL in a simulated quadcopter navigation task by showing that it can solve tasks where a constrained greedy strategy fails.

2 RELATED WORK

To the best of our knowledge, there is no prior work addressing mode-constrained RL when both the dynamics modes and the mode constraint are *unknown a priori*.

Constrained Markov decision processes (CMDPs) A common paradigm when considering constraints in RL is to consider CMDPs (Altman, 1999). In this setting, the agent must satisfy a set of constraints defined by additional cost functions. In contrast, in the mode-constrained setting, the agent does not directly observe the constraint in the form of a cost from the environment.

Mode remaining planning Scannell et al., 2021 present a mode remaining trajectory optimisation algorithm that leverages a learned dynamics model. However, they do not consider the MBRL setting as they assume access to the environment *a priori*. Further to this, their approach attempts to remain in a single dynamics mode, without enforcing any constraints.

Safe active learning Our work has similarities to Schreiter et al., 2015 as they use a Gaussian Process (GP) classifier to identify safe and unsafe regions when learning a dynamics models in an active learning setting. However, similarly to the CMDP setting (Altman, 1999) they assume that they can directly observe whether a particular state transition is safe or not. In contrast, we cannot directly observe the mode constraint and must infer it using a dynamics model.

Safe model-based RL In low-dimensional continuous-control problems, Berkenkamp et al., 2017 propose to construct confidence intervals around Lyapunov functions by first learning a GP dynamics model. They then use their confidence intervals to optimise a policy such that it remains in a Lyapunov stable region of attraction. Their method encodes safety via stability guarantees but is limited to systems where the dynamics are Lipschitz continuous. Finally Koller et al., 2018 and Hewing et al., 2020 consider safe Model Predictive Control (MPC) schemes which use GP dynamics models to certify the safety of actions.

3 PROBLEM STATEMENT

We consider *multimodal*, *stochastic* environments with states $\mathbf{s}_t \in \mathcal{S} \subseteq \mathbb{R}^{D_x}$, actions $\mathbf{a}_t \in \mathcal{A} \subseteq \mathbb{R}^{D_u}$ and transition dynamics given by

$$\mathbf{s}_{t+1} = f_k(\mathbf{s}_t, \mathbf{a}_t) + \boldsymbol{\epsilon}_{k,t}, \quad \text{if } \alpha(\mathbf{s}_t) = k, \tag{1}$$

where the discrete mode indicator function $\alpha : \mathcal{S} \to \{1, \dots, K\}$ indicates which of the K underlying dynamics modes $\{f_k : \mathcal{S}_k \times \mathcal{A} \to \mathcal{S}\}_{k=1}^K$ and associated i.i.d. noise models $\boldsymbol{\epsilon}_{k,t}$ governs the environment at a given time step t. We refer to the output of the mode indicator function as the mode indicator variable $\alpha_t = \alpha(\mathbf{s}_t) \in \{1, \dots, K\}$.

We consider systems where the K underlying dynamics modes have pair-wise disjoint state domains. That is, each mode's state domain is given by $\mathcal{S}_k = \{\mathbf{s} \in \mathcal{S} \mid \alpha(\mathbf{s}) = k\}$, with $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ for distinct $i, j \in \{1, \dots K\}$. Notice that the dynamics modes $\{f_k\}_{k=1}^K$ are free to leave their state spaces $\mathcal{S}_k$ and enter other dynamics modes $\mathcal{S}$.

Problem statement We consider controlling the stochastic system in Eq. (1) in an episodic setting, over a horizon T. We assume that after each episode the system is reset to a known initial state $\mathbf{s}_0$. We consider general deterministic policies $\pi \in \Pi$, which encapsulates both closed-loop policies $\pi(\mathbf{s}_t)$ and open-loop policies $\pi(t)$. For a known transition dynamics model $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, the performance of a policy π is the sum of rewards over the horizon, in expectation over the transition noise,

$$J(\pi, f) = \mathbb{E}_{\boldsymbol{\epsilon}_{0:T}} \left[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \mid \mathbf{s}_0 = \mathbf{s}_0 \right]. \tag{2}$$

The goal of our work is to find the optimal policy π^* whilst remaining in a desired dynamics mode k^*,

$$\max_{\pi \in \Pi} J(\pi, f) \quad \text{s.t.} \quad \alpha(\mathbf{s}_t) = k^* \quad \forall t \in \{0, \dots, T\}. \tag{3}$$

Formally, a mode-constrained system is defined as follows.

Definition 3.1 (mode-constrained) *Let $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ denote a multimodal dynamical system and $\mathcal{S}_{k^*} = \{\mathbf{s} \in \mathcal{S} \mid \alpha(\mathbf{s}) = k^*\}$ denote the state domain of the desired dynamics mode k^*. Given an initial state $\mathbf{s}_0 \in \mathcal{S}_{k^*}$ and a policy $\pi \in \Pi$, the controlled system is said to be mode-constrained under the policy π iff:*

$$f(\mathbf{s}_t, \pi(\mathbf{s}_t, t)) \in \mathcal{S}_{k^*} \quad \forall t \in \{0, \dots, T\} \tag{4}$$

Given that neither the underlying dynamics modes $\{f_k\}_{k=1}^K$, nor how the system switches between them α, are *known a priori*, it is not possible to solve Eq. (3) with the mode constraint in Definition 3.1. Therefore, we relax the requirement to finding a mode-constrained policy with high probability. We formally define a $\delta -$ mode-constrained policy as follows.

Definition 3.2 (δ-mode-constrained) *Let $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ denote a multimodal dynamical system and k^* a desired dynamics mode defined by its state domain $\mathcal{S}_{k^*} = \{\mathbf{s} \in \mathcal{S} \mid \alpha(\mathbf{s}) = k^*\}$. Given an initial state $\mathbf{s}_0 \in \mathcal{S}_{k^*}$ and $\delta \in (0, 1]$, a controlled system is said to be δ-mode-constrained under the policy π iff:*

$$\Pr(\forall t \in \{0, \dots, T\} : f(\mathbf{s}_t, \pi(\mathbf{s}_t, t)) \in \mathcal{S}_{k^*}) \geq 1 - \delta. \tag{5}$$

Policies satisfying this $\delta-$ mode-constrained definition should remain in the desired dynamics mode with probability up to $1 - \delta$. It is worth noting that the agent would not be able to explore the environment without relaxing the mode constraint from Definition 3.1. Increasing δ promotes exploration but also increases the chance of violating the mode constraint. As such, δ can be viewed as setting the conservativeness of the agent's exploration.

Initial mode remaining controller In robotics applications, an initial set of poor performing controllers can normally be obtained via simulation or domain knowledge. We assume access to an initial data set of state transitions $\mathcal{D}_0 = \{(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1}\}_{t=1}^{TN}$ from N episodes of length T. We use it to learn a predictive dynamics model $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t, \mathcal{D}_0)$ which is locally accurate around the start state $\mathbf{s}_0$.

Assumption 3.1 *A state transition data set has been collected $\mathcal{D}_0 = \{(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1}\}_{t=1}^{TN}$ from an initial region of the state space $\mathcal{S}_0 \subseteq \mathcal{S}_{k^*}$, which belongs to the desired dynamics mode k^* and contains the start state $\mathbf{s}_0 \in \mathcal{S}_0$.*

Although such a model can be used to learn an initial policy, it will not work outside of the initial state domain $\mathcal{S}_0$ and may not be able to find a $\delta-$ mode-constrained policy, due to the model having high *epistemic uncertainty*. For this reason, we adopt a MBRL strategy which incrementally explores the environment subject to a $\delta-$ mode constraint. Each iteration collects data from the environment, reducing the dynamics model's *epistemic uncertainty*, in turn expanding the $\delta-$ mode-constrained region.

4 MODE-CONSTRAINED MODEL-BASED RL

We propose to solve the mode-constrained RL problem in Eq. (3) by synergising model learning and planning in a MBRL algorithm we name Mode Constrained Model-Based Reinforcement Learning (ModeRL). Our approach is detailed in Alg. 1.

4.1 DYNAMICS MODEL

ModeRL learns a single-step dynamics model and we adopt the delta state formulation to regularise the predictive distribution. We denote a state difference output as $\Delta\mathbf{s}_{t+1} = \mathbf{s}_{t+1} - \mathbf{s}_t$ and the set of all state difference outputs as $\Delta\mathbf{S}$. We further denote a state-action input as $\hat{\mathbf{s}}_t = (\mathbf{s}_t, \mathbf{a}_t)$, the set of all state-action inputs as $\hat{\mathbf{S}}$, the set of all state inputs as $\mathbf{S}$ and the state transition data set at episode i as $\mathcal{D}_{0:i}$.

Algorithm 1 ModeRL

Require: Start state $\mathbf{s}_0$, desired dynamics mode k^*, initial data set $\mathcal{D}_0$, policy π_0, dynamics model $p(\mathbf{s}_{t+1} \mid \hat{\mathbf{s}}_t, \mathcal{D}_0)$

```
for i = 0, 1, ..., num episodes do
    while not converged do
        Sample N_b state transitions B ~ D_{0:i} uniformly
        Update dynamics using Eq. (11) with B
    end while
    Optimise policy π_i using Eq. (13) or Eq. (12)
    Collect data D_{i+1} using π_{i+1}
    Update agent's data set D_{0:i+1} = D_{i+1} ∪ D_{0:i}
end for
```

The main goal of our dynamics model is to jointly infer the mode constraint along with the underlying dynamics modes. In particular, we would like our model to,

1. Formulate a prior over the mode constraint where we can encode prior knowledge, potentially enabling ModeRL to find a policy without ever violating the mode constraint. In practice, encoding prior knowledge allows us to exploit Bayesian interpolation (MacKay, 1992) to reduce the number of constraint violations during learning.

2. Disentangle the sources of uncertainty in the mode constraint so that ModeRL can target exploration where the agent has high *epistemic uncertainty* in the mode constraint.

Marginal likelihood Mixtures of Gaussian Process Experts (MoGPE) models are a natural choice for modelling multimodal systems as they automatically infer the assignment of observations to dynamics modes (experts). Let us start by introducing the MoGPE marginal likelihood,

$$p\left(\Delta\mathbf{S} \mid \hat{\mathbf{S}}\right) = \sum_{\boldsymbol{\alpha}} \underbrace{p\left(\boldsymbol{\alpha} \mid \mathbf{S}\right)}_{\text{gating network}} \left[\prod_{k=1}^{K} \underbrace{p\left(\Delta\mathbf{S}_k \mid \hat{\mathbf{S}}_k, \boldsymbol{\theta}_k\right)}_{\text{dynamics mode } k}\right] \tag{6}$$

where $\hat{\mathbf{S}}_k$ denotes the set of N_k inputs assigned to dynamics mode k, i.e. $\hat{\mathbf{S}}_k = \{\hat{\mathbf{s}}_t \in \hat{\mathbf{S}} \mid \alpha(\mathbf{s}_t) = k\}$. Similarly for the outputs we have $\Delta\mathbf{S}_k = \{\Delta\mathbf{s}_{t+1} \in \Delta\mathbf{S} \mid \alpha(\mathbf{s}_t) = k\}$. Note that there is a joint distribution corresponding to every possible combination of assignments of observations to dynamics modes. Hence, Eq. (6) is a sum over exponentially many (K^N) sets of assignments, where $\boldsymbol{\alpha} = \{\alpha_1, \dots, \alpha_N\}$

represents a set of assignments for all observations. This distribution factors into the product over modes, where each mode models the joint Gaussian distribution over the observations assigned to it.

Learning the mode constraint The gating network indicates which dynamics mode governs the system at a given state input. It is of particular importance in our work as we use it to represent our mode constraint. Motivated by synergising model learning and planning, we formulate our gating network using input-dependent functions – known as gating functions – and place GP priors over them. In Sec. 4.3 we exploit the disentangled *epistemic uncertainty* represented in our GP-based gating network to help ModeRL escape local optima induced by the mode constraint. Similar to Tresp, 2000, our gating network resembles a GP classification model,

$$p\left(\boldsymbol{\alpha} \mid \hat{\mathbf{S}}\right) = \mathbb{E}_{\underbrace{p(\mathbf{h}(\mathbf{S}))}_{\text{GP prior(s)}}}\left[\prod_{n=1}^{N} \underbrace{p\left(\alpha_t \mid \mathbf{h}(\mathbf{s}_t)\right)}_{\text{classification likelihood}}\right] \tag{7}$$

where $p\left(\alpha_t \mid \mathbf{h}(\mathbf{s}_t)\right)$ represents a classification likelihood parameterised by K gating functions $\mathbf{h} : \mathcal{X} \to \mathbb{R}^K$. We use a Bernoulli likelihood when $K = 2$ and a softmax when $K > 2$. We place GP priors on each of the gating functions $p(\mathbf{h}(\hat{\mathbf{S}})) = \prod_{k=1}^{K} \mathcal{N}\left(p(\mathbf{h}(\hat{\mathbf{S}}) \mid \hat{\mu}_k(\hat{\mathbf{S}}), \hat{k}_k(\hat{\mathbf{S}}, \hat{\mathbf{S}})\right)$, where $\hat{\mu}_k(\cdot)$ and $\hat{k}_k(\cdot, \cdot)$ represent the mean and covariance functions associated with the k^{th} gating function. In our gating network formulation the GP posteriors represent the mode constraint's *epistemic uncertainty*.

Dynamics modes We model the underlying dynamics modes $\{f_k\}_{k=1}^{K}$ as independent GP regression models,

$$\underbrace{p\left(\Delta\mathbf{S}_k \mid \hat{\mathbf{S}}_k\right)}_{\text{dyanmics mode } k} = \underbrace{\mathbb{E}_{p(f_k(\hat{\mathbf{S}}_k))}}_{\text{GP prior}}\left[\prod_{n=1}^{N_k} \underbrace{p\left(\Delta\mathbf{s}_{t+1} \mid f_k(\hat{\mathbf{s}}_t)\right)}_{\text{Gaussian likelihood}}\right],$$

where each mode's GP prior is given by $p(f_k(\hat{\mathbf{S}}_k)) = \mathcal{N}\left(f_k(\hat{\mathbf{S}}_k) \mid \mu_k(\hat{\mathbf{S}}_k), k_k(\hat{\mathbf{S}}_k, \hat{\mathbf{S}}_k)\right)$ with $\mu_k(\cdot)$ and $k_k(\cdot, \cdot)$ representing the mean and covariance functions associated with the k^{th} mode's GP prior respectively. Note that as the assignment of observations to modes is *not known a priori*, we must infer the assignments from observations. In our model, each dynamics mode's Gaussian likelihood represents the mode's *aleatoric uncertaity* (the transition noise in this case) whilst each mode's GP posterior represents the mode's *epistemic uncertainty*.

4.2 Dynamics Learning

Performing Bayesian inference in our dynamics model involves finding the posterior over the latent variables $p(\{f(\hat{\mathbf{S}})\}_{k=1}^{K}, \mathbf{h}(\mathbf{S}) \mid \mathcal{D}_{0:i})$, which requires calculating the marginal likelihood in Eq. (6). As such, exact inference in our model is intractable due to the marginalisation over the set of mode indicator variables. For this reason, we resort to a variational approximation.

Following the approach by Titsias, 2009, we augment the probability space with a set of M inducing variables for each GP. However, instead of collapsing these inducing variables, we represent them as variational distributions and use them to lower bound the marginal likelihood, similar to Hensman et al., 2013, 2015. Fig. 6 shows the graphical model of the augmented joint probability space.

Augmented dynamics modes We sidestep the hard assignment of observations to modes by augmenting each dynamics GP with a set of separate independent inducing points,

$$p(f_k(\boldsymbol{\zeta}_k)) = \mathcal{N}\left(f_k(\boldsymbol{\zeta}_k) \mid \mu_k(\boldsymbol{\zeta}_k), k_k(\boldsymbol{\zeta}_k, \boldsymbol{\zeta}_k)\right). \tag{8}$$

Introducing separate inducing points from each mode's GP can loosely be seen as "partitioning" the observations between modes. However, as the assignment of observations to modes is *not known a priori*, the inducing inputs $\boldsymbol{\zeta}_k$ and variables $f_k(\boldsymbol{\zeta}_k)$, must be inferred from observations.

Augmented gating network We follow a similar approach for the gating network and augment each gating function GP with a set of M inducing points,

$$p(h_k(\boldsymbol{\xi})) = \mathcal{N}\left(h_k(\boldsymbol{\xi}) \mid \hat{\mu}_k(\boldsymbol{\xi}), \hat{k}_k(\boldsymbol{\xi}, \boldsymbol{\xi})\right). \tag{9}$$

The distribution over all gating functions is denoted $p(\mathbf{h}(\boldsymbol{\xi})) = \prod_{k=1}^{K} p(h_k(\boldsymbol{\xi}))$. In contrast to the dynamics modes, the gating function GPs share inducing inputs $\boldsymbol{\xi}$.

Marginal likelihood We use these inducing points to approximate the true marginal likelihood with,

$$p\left(\Delta\mathbf{S} \mid \hat{\mathbf{S}}\right) \approx \mathbb{E}_{p(\mathbf{h}(\boldsymbol{\xi}))p(\mathbf{f}(\boldsymbol{\zeta}))}\Bigg[\prod_{n=1}^{N}\sum_{k=1}^{K} \Pr(\alpha_t = k \mid \mathbf{h}(\boldsymbol{\xi}))p(\Delta\mathbf{s}_{t+1} \mid f_k(\boldsymbol{\zeta}_k))\Bigg], \tag{10}$$

where the conditional distributions $p(\Delta\mathbf{s}_{t+1} \mid f_k(\boldsymbol{\zeta}_k))$ and $\Pr(\alpha_t = k \mid \mathbf{h}(\boldsymbol{\xi}))$ follow from standard sparse GP methodologies. See Eq. (14). Importantly, the factorisation over observations is outside of the marginalisation over the mode indicator variable, i.e. the mode indicator variable can be marginalised for each data point separately. This is not usually the case for MoGPE methods. Our approximation assumes that the inducing variables, $\{f_k(\boldsymbol{\zeta}_k)\}_{k=1}^{K}$, are a sufficient statistic for their associated latent function values, $\{f_k(\hat{\mathbf{S}}_k)\}_{k=1}^{K}$ and the set of assignments $\boldsymbol{\alpha}$. It becomes exact when each mode's inducing points represent the true data partition $\{\boldsymbol{\zeta}_k, f_k(\boldsymbol{\zeta}_k)\}_{k=1}^{K} = \{\hat{\mathbf{S}}_k, f_k(\hat{\mathbf{S}}_k)\}_{k=1}^{K}$.

Evidence Lower Bound Following a similar approach to

Hensman et al., 2013, 2015, we lower bound Eq. (10),

$$\mathcal{L}(\{\mathbf{m}_k,\mathbf{L}_k,\hat{\mathbf{m}}_k,\hat{\mathbf{L}}_k,\boldsymbol{\zeta}_k\}_{k=1}^K,\boldsymbol{\xi}) = \sum_{t=1}^{N}\mathbb{E}_{q(\mathbf{h}(\hat{\mathbf{s}}_t))q(\mathbf{f}(\boldsymbol{\zeta}))}\Bigg[\log\sum_{k=1}^{K}\Pr\left(\alpha_t=k\mid\mathbf{h}(\hat{\mathbf{s}}_t)\right)p(\Delta\mathbf{s}_{t+1}\mid f_k(\boldsymbol{\zeta}_k))\Bigg] - \sum_{k=1}^{K}\mathrm{KL}\left(q(f_k(\boldsymbol{\zeta}_k))\mid\mid p(f_k(\boldsymbol{\zeta}_k))\right) - \sum_{k=1}^{K}\mathrm{KL}\left(q(h_k(\boldsymbol{\xi}))\mid\mid p(h_k(\boldsymbol{\xi}))\right) \tag{11}$$

where the dynamics mode's variational posterior is given by $q(\mathbf{f}(\boldsymbol{\zeta})) = \prod_{k=1}^{K}\mathcal{N}\left(f_k(\boldsymbol{\zeta}_k)\mid\mathbf{m}_k,\mathbf{L}_k\mathbf{L}_k^T\right)$ and the gating network's variational posterior is given by $q(\mathbf{h}(\hat{\mathbf{s}}_t)) = \prod_{k=1}^{K}\int p(h_k(\hat{\mathbf{s}}_t)\mid h_k(\boldsymbol{\xi}))\mathcal{N}\left(h_k(\boldsymbol{\xi})\mid\hat{\mathbf{m}}_k,\hat{\mathbf{L}}_k\hat{\mathbf{L}}_k^T\right)\mathrm{d}\mathbf{h}(\boldsymbol{\xi})$.

Optimisation The bound in Eq. (11) induces a local factorisation over observations and has a set of global variables – the necessary conditions to perform Stochastic Variational Inference (SVI) (Hoffman et al., 2013) on $q(\mathbf{f}(\boldsymbol{\zeta}))$ and $q(\mathbf{h}(\boldsymbol{\xi}))$, i.e. optimise $\{\mathbf{m}_k,\mathbf{L}_k,\hat{\mathbf{m}}_k,\hat{\mathbf{L}}_k\}_{k=1}^K$. We treat the inducing inputs $\boldsymbol{\xi},\{\boldsymbol{\zeta}_k\}_{k=1}^K$, kernel hyperparameters and noise variances, as variational hyperparameters and optimise them alongside the variational parameters, using Adam (Kingma and Ba, 2017). We use mini-batches and approximate the expectations over the log-likelihood using Monte Carlo samples.

Our approach can loosely be viewed as parameterising the nonparametric model in Eq. (6) to obtain a desirable factorisation for 1) constructing a GP-based gating network and 2) deriving an ELBO that can be optimised with stochastic gradient methods. Imortantly, our approach still captures the complex dependencies between the gating network and dynamics modes.

Predictions Given our variational approximation, we make predictions at a new input $\hat{\mathbf{s}}_t$ with,

$$p(f_k(\hat{\mathbf{s}}_t)\mid\hat{\mathbf{s}}_t,\mathcal{D}_{0:i}) \approx \int p(f_k(\hat{\mathbf{s}}_t)\mid f_k(\boldsymbol{\zeta}_k))q(f_k(\boldsymbol{\zeta}_k))\mathrm{d}f_k(\boldsymbol{\zeta}_k),$$
$$p(h_k(\mathbf{s}_t)\mid\mathbf{s}_t,\mathcal{D}_{0:i}) \approx \int p(h_k(\mathbf{s}_t)\mid h_k(\boldsymbol{\xi}))q(h_k(\boldsymbol{\xi}))\mathrm{d}h_k(\boldsymbol{\xi}).$$

4.3 Planning

We now detail our planning algorithm which leverages the latent structure of our dynamics model to:

1. enforce the mode constraint with high probability,
2. target exploration where the agent is not confident in it's belief of the mode constraint, i.e. where the gating network's *epistemic uncertainty* is high.

Multi-step dynamics predictions Let us first observe that if the controlled system satisfies the mode constraint, the system will be fully governed by the desired dynamics mode f_{k^*}. We leverage this observation and simplify making multi-step dynamics predictions to using only the desired mode's dynamics f_{k^*}. This enables us to approximate the GP dynamics integration in closed from using moment matching (Deisenroth and Rasmussen, 2011; Girard et al., 2003; Kamthe and Deisenroth, 2018). This approximation has been shown to work well in RL contexts (Cutler and How, 2015; Deisenroth et al., 2015; Deisenroth and Rasmussen, 2011; Pan and Theodorou, 2014; Pan et al., 2015).

Objective Given this approach for making multi-step dynamics predictions, we formulate a relaxed version of the mode-constrained problem in Eq. (3),

$$\pi_{\text{greedy}} = \max_{\pi\in\Pi}\underbrace{\mathbb{E}_{f_{k^*}\sim p(f_{k^*}\mid\mathcal{D}_{0:i})}\left[J(\pi,f_{k^*})\right]}_{\text{greedy exploitation}}, \tag{12a}$$

$$\text{s.t. }\underbrace{\mathbb{E}_{f_{k^*}\sim p(f_{k^*}\mid\mathcal{D}_{0:i})}\left[\Pr(\alpha_t=k^*\mid\mathbf{s}_t)\right]\geq 1-\delta}_{\delta\text{ mode constraint}},\quad\forall t, \tag{12b}$$

The expectations are taken over the desired mode's f_{k^*} GP dynamics. This approach has the added benefit that we can calculate the expected mode constraint in closed form and given a quadratic reward function, we can also calculate the objective in closed form. Importantly, the expected mode constraint considers the *epistemic uncertainty* in both the desired dynamics mode $p(f_{k^*}(\hat{\mathbf{s}}_t)\mid\hat{\mathbf{s}}_t,\mathcal{D}_{0:i})$ and in the gating network $p(\mathbf{h}(\mathbf{s}_t)\mid\mathbf{s}_t,\mathcal{D}_{0:i})$. This ensures that the controlled system is $\delta-$mode-constrained (Definition 3.2) under the uncertainty of the learned dynamics model. Intuitively, it will remain where the dynamics model's *epistemic uncertainty* is low because the expectation results in lower proababilities for more uncertain states.

Exploration In our experiments, the constraint in Eq. (12b) hinders exploration and prevents the agent solving the task. This can be seen in the second from left plot in Fig. 3, where the mode constraint has induced a local optimum and stopped the agent reaching the target state. We name the approach in Eq. (12) the greedy constrained strategy π_{greedy}. We propose to overcome the issue of local optima induced by the mode constraint, by targeting exploration where the agent has high *epistemic uncertainty* in the mode constraint. We do this by augmenting our objective with an intrinsic exploration term that encourages the agent to explore where its gating network is *uncertain*. We use the entropy of the desired mode's gating function h_{k^*} over a trajectory $\bar{\mathbf{s}} = \mathbf{s}_0,\ldots,\mathbf{s}_T$ as our intrinsic exploration term.

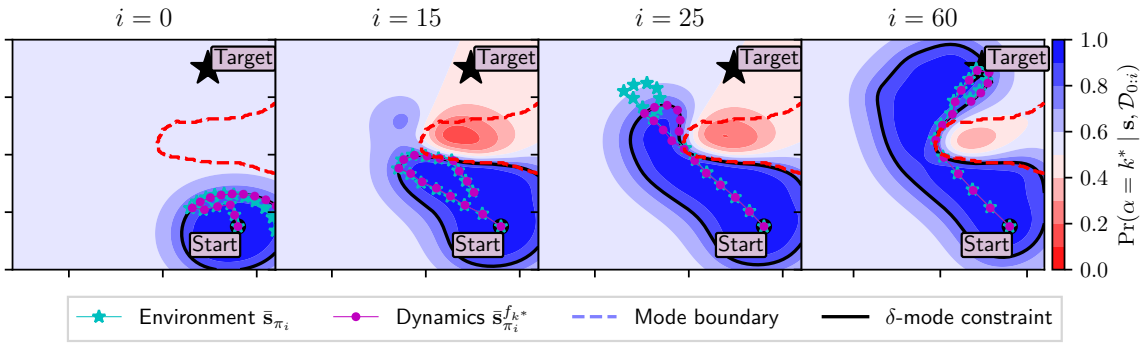

Figure 2: Visualisation of four episodes i of ModeRL in the quadcopter navigation task from Fig. 1, where the goal is to navigate to the black star, whilst avoiding the turbulent dynamics mode (dashed red line). The contour plots indicate the agent's belief of being in the desired dynamics mode $\Pr(\alpha = k^* \mid \mathbf{s}, \mathcal{D}_{0:i})$ at each iteration, i.e. after training on $\mathcal{D}_{0:i}$. The black lines show the $\delta-$mode constraint ($\Pr(\alpha = k^* \mid \mathbf{s}, \mathcal{D}_{0:i}) = 0.8$) expanding during learning. At each episode i we roll out the policy in the desired mode's GP dynamics (magenta) as well as the in the environment (cyan).

This leads to our strategy taking the form,

$$\pi = \max_{\pi \in \Pi} \underbrace{\mathbb{E}_{f_{k^*} \sim p(f_{k^*} \mid \mathcal{D}_{0:i})} \left[J(\pi, f_{k^*}) \right]}_{\text{greedy exploitation}} + \underbrace{\mathcal{H}\left[h_{k^*}(\bar{\mathbf{s}}) \mid \bar{\mathbf{s}}, \mathcal{D}_{0:i} \right]}_{\text{exploration}} \tag{13a}$$

$$\text{s.t.} \underbrace{\mathbb{E}_{f_{k^*} \sim p(f_{k^*} \mid \mathcal{D}_{0:i})} \left[\Pr(\alpha_t = k^* \mid \mathbf{s}_t) \right] \geq 1 - \delta}_{\delta \text{ mode constraint}} \quad \forall t. \tag{13b}$$

Intuitively, the entropy term should enable the agent to escape local optima induced by the mode constraint as it encourages the agent to explore away from regions of the mode constraint that it has already observed. This is because the gating network's *epistemic uncertainty* will be low where it has observed the environment.

Epistemic vs aleatoric uncertainty When adopting this approach, it is extremely important to disentangle the gating network's *epistemic uncertainty* from its *aleatoric uncertainty*. For example, what is the meaning of the agent's belief in the mode indicator variable tending to a uniform distribution i.e. $\Pr(\alpha = k \mid \mathbf{s}, \mathcal{D}_{0:i}) = \frac{1}{K}$? Does it mean that the agent has not observed the environment near $\mathbf{s}$ (high *epistemic uncertainty*), or that it has observed the environment near $\mathbf{s}$ but is still uncertain which mode governs the dynamics (high *aleatoric uncertainty*)? We show the importance of disentangling these sources of uncertainty in our experiments. This motivated our GP-based gating network in the MoGPE dynamics model, as it principally disentangles the sources of uncertainty over α, by representing the *epistemic uncertainty* in the GPs over the gating functions.

Non-myopic exploration Further to providing a principled approach for disentangling sources of uncertainty, our GP-based gating network enables us to deploy a non-myopic exploration strategy. That is, we are able to steer the agent along a trajectory that will maximise the uncertainty reduction over the entire trajectory, by considering the joint entropy over a trajectory $\mathcal{H}[h_{k^*}(\bar{\mathbf{s}}) \mid \bar{\mathbf{s}}, \mathcal{D}_{0:2}]$. In contrast, myopic exploration considers uncertainty reduction of each state independently without considering the influence of other states in the trajectory. e.g. $\frac{1}{T}\mathcal{H}[h_{k^*}(\mathbf{s}_t) \mid \mathbf{s}_t, \mathcal{D}_{0:2}]$

Trajectory optimisation We solve the constrained optimisation problem in Eq. (13) using Sequential Least Squares Quadratic Programming (SLSQP) in SciPy (Virtanen et al., 2020). In particular, we use the TensorFlow (Abadi et al., 2015) wrapper provided by GPflow (Matthews et al., 2017).

5 EXPERIMENTAL RESULTS

We test our method on a 2D quadcopter navigation example, where the goal is to navigate from an initial state $\mathbf{s}_0$, to a target state $\mathbf{s}_f$, whilst avoiding a turbulent dynamics mode. See App. B for a schematic of the environment and details of the problem. Our expriments seek to answer the following questions:

1. Why does the greedy exploitation strategy in Eq. (12) fail to solve the mode-constrained problem in Eq. (3)?
2. Does our strategy in Eq. (13), which targets exploration where the mode constraint's *epistemic uncertainty* is high, solve the mode-constrained problem in Eq. (3)?
3. Is it important to disentangle sources of uncertainty in the mode constraint?
4. Does our non-myopic exploration strategy provide benefits over a myopic exploration strategy?

We do not compare to a baseline from the literature because 1) there is no prior work considering mode-constrained RL

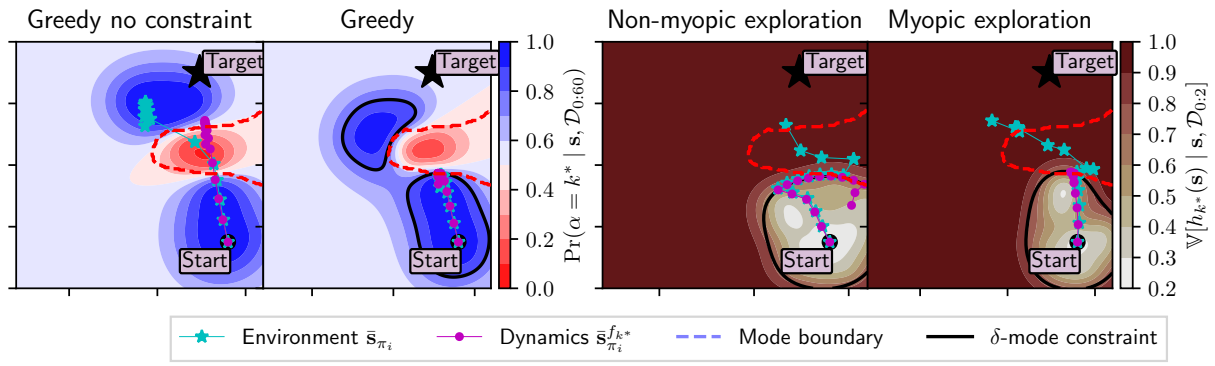

Figure 3: (Left) show trajectories found by the greedy exploitation strategy in Eq. (12), with and without the $\delta-$mode constraint (black line). (Right) shows trajectories found by our strategy in Eq. (13) when using our non-myopic exploration term $\mathcal{H}[h_{k^*}(\bar{\mathbf{s}}) \mid \bar{\mathbf{s}}, \mathcal{D}_{0:2}]$, compared to using the myopic exploration term $\frac{1}{T}\sum_{t=1}^{T}\mathcal{H}[h_{k^*}(\mathbf{s}_t) \mid \mathbf{s}_t, \mathcal{D}_{0:2}]$. We overlay the trajectories on the GP posterior variance associated with desired mode's gating function $\mathbb{V}[h_{k^*}(\mathbf{s}) \mid \mathbf{s}, \mathcal{D}_{0:2}]$ at episode $i = 2$.

and 2) our mode constraint is *unknown a priori* and highly coupled to our dynamics model. As a result, any constrained RL method would still require our dynamics model to infer the mode constraint. However, we are able to use the greedy strategy in Eq. (12), both with and without the mode constraints, as baselines to evaluate our method. Our experiment configuration is detailed in App. C.

Why does the greedy exploitation fail? Solving the greedy strategy without the mode constraint results in the optimisation finding trajectories that leave the desired dynamics mode. This is illustrated in Fig. 3 (left), which shows the unconstrained greedy strategy converged to a solution navigating straight to the target state. As a result, the trajectory leaves the desired dynamics mode and passes through the turbulent dynamics mode. This is to be expected as the strategy is not aware of the turbulent dynamics mode. In contrast, the constrained greedy strategy, shown in Fig. 3 (second left), does not leave the desired dynamics mode. However, the optimisation does not converge to the global optimum, i.e. the trajectory does not navigate to the target state $\mathbf{s}_f$. Instead, it gets stuck at the mode boundary. Fig. 3 (second left) shows the local optimum induced by the mode constraint.

Does our uncertainty guided exploration work? This motivated our strategy in Eq. (13), which favours searching regions of the state space where the mode constraint's *epistemic uncertainty* is high. Our hypothesis is that the mode constraint's *epistemic uncertainty* can be used to escape local optima induced by the mode constraint, such as the one in the second left plot of Fig. 3. Fig. 2 shows four episodes i of ModeRL in the quadcopter navtigation task. Reading from left to right, the contours show how the agent's belief of being in the desired dynamics mode $\Pr(\alpha = k^* \mid \mathbf{s}, \mathcal{D}_{0:i})$ changes as the agent interacts with the environment, collects data $\mathcal{D}_{0:i}$ and updates its dynamics model, i.e. trains on $\mathcal{D}_{0:i}$. It shows that the $\delta-$mode-constrained region (black line) expands as the agent reduces its *epistemic uncertainty* by training on the new observations. The middle two plots show that the agent was able to escape the local optimum. This supports our hypothesis that the mode constraint's *epistemic uncertainty* can be used to escape local optima induced by the mode constraint. As a result, the agent has been able to successfully navigate to the target state, as visualised in the right hand plot.

Is it important to disentangle sources of uncertainty? We now evaluate the importance of disentangling the sources of uncertainty in the gating network. In the left plot of Fig. 4 we visualise the entropy of the mode indicator variable $\mathcal{H}[\alpha \mid \mathbf{s}, \mathcal{D}_{0:20}]$ (*aleatoric uncertainty*) at a region of the mode boundary that the agent has observed (black crosses). The entropy is high (red) even though we have observed the environment at these states. As such, the agent would keep exploring the mode boundary even though it has been observed. In contrast, the middle plot shows the entropy of the gating function $\mathcal{H}[h_{k^*}(\mathbf{s}) \mid \mathbf{s}, \mathcal{D}_{0:20}]$. The entropy is low (white) around the observations (black crosses), which indicates that our GP-based gating network disentangles the *epistemic uncertainty* into the gating network's GPs. Intuitively, the entropy of the desired mode's gating function is a much better objective for exploration because it encourages the exploration away from the mode boundary once it has been observed. The right plot of Fig. 4 shows that the agent is not able to escape the local optimum induced by the mode constraint, when using the entropy of mode indicator variable for exploration.

Non-myopic vs myopic exploration? We now test the importance of our non myopic exploration term, i.e. using the joint entropy of the desired mode's gating function over a trajectory $\mathcal{H}[h_{k^*}(\bar{\mathbf{s}}) \mid \bar{\mathbf{s}}, \mathcal{D}_{0:2}]$, instead of taking the mean of the gating function's entropy at each time step $\frac{1}{T}\sum_{t=0}^{T}\mathcal{H}\left[h_{k^*}(\mathbf{s}_t) \mid \mathbf{s}_t, \mathcal{D}_{0:i}\right]$. In the right plots of

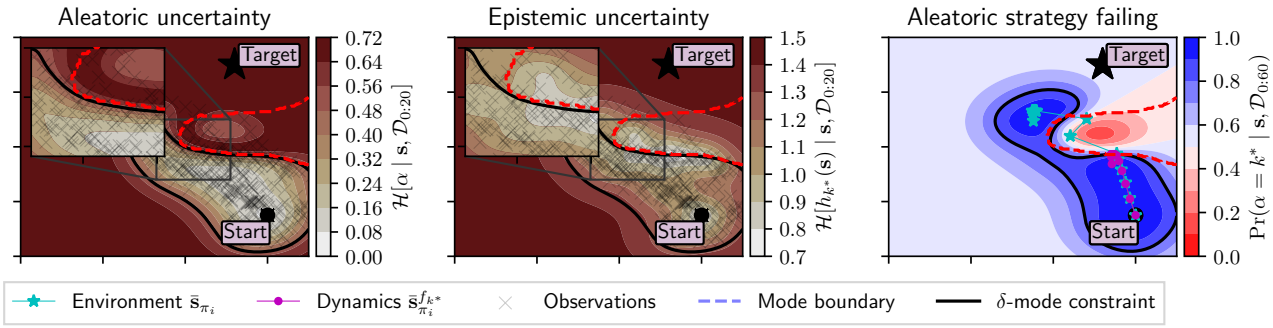

Figure 4: **Epistemic vs aleatoric uncertainty** Illustration of the shortcomings of using the entropy of the mode indicator variable (*aleatoric uncertainty*) for exploration. (Left) shows that the entropy of the mode indicator variable $\mathcal{H}[\alpha \mid \mathbf{s}, \mathcal{D}_{0:20}]$ (*aleatoric uncertainty*) at a region of the mode boundary which has been observed (black crosses), is high (red), even though we have observed the environment at these states. In contrast, (middle) shows that the entropy of the desired mode's gating function $\mathcal{H}[h_{k^*}(\mathbf{s}) \mid \mathbf{s}, \mathcal{D}_{0:20}]$ (*epistemic uncertainty*) is low (white). (Right) visualises our method convering to a local optimum when we replace the exploration term in Eq. (13) with the entropy of the mode indicator variable

Fig. 3, we overlay the trajectories found with the non-myopic and myopic exploration terms, over the GP posterior variance associated with desired mode's gating function $\mathbb{V}[h_{k^*}(\mathbf{s}) \mid \mathbf{s}, \mathcal{D}_{0:2}]$ at episode $i = 2$. The right hand plot of Fig. 3 shows that using the myopic exploration term (i.e. the entropy over a horizon without conditioning on the rest of the trajectory), results in the trajectory navigating to a single state of high entropy and remaining in that state for the rest of the trajectory. This is an undesirable behaviour. In contrast, the second from right plot in Fig. 3 shows our non-myopic exploration term $\mathcal{H}[h_{k^*}(\bar{\mathbf{s}}) \mid \bar{\mathbf{s}}, \mathcal{D}_{0:2}]$ spreading out. This is a desirable behaviour because it reduces the number of environment interactions that are required to solve the task, i.e. it improves sample efficiency.

5.1 Practical Considerations

Warm start trajectory optimisation In practice, the constrained optimisation in Eq. (13) fails if the initial trajectory does not satisfy the $\delta-$mode constraint. We overcome this issue by warm starting our trajectory optimiser by solving an unconstrained optimisation to a "fake" target state in the initial state domain $\mathcal{S}_0$.

Fixing model parameters during training During its initial iterations, ModeRL only explores the desired dynamics mode and does not observe any state transitions belonging to another mode. This results in the $\delta-$mode-constrained region expanding significantly further than the observed data because the gating function's GP learns a long lengthscale. For this reason, we fix the kernel's hyperparameters (e.g. lengthscale and signal variance). We also fix the noise variance in the non desired dynamics modes.

Inducing points We initialise each of the sparse GPs with a fixed number of inducing points uniformly sampled for $\mathcal{D}_0$. Although this approach worked well in the experiments, it is unlikely to scale to larger problems. As such, an interesting direction for future work is to study methods for dynamically adding new inducing points to each GP.

6 CONCLUSION

We introduced ModeRL, a Bayesian MBRL algorithm for low-dimensional continuous control problems, that constrains exploration to a single dynamics mode up to a given probability. It uses a MoGPE method to infer the mode constraint alongside the underlying dynamics modes. Importantly, the GP-based gating network disentangles the sources of uncertainty in the learned mode constraint, enabling ModeRL to escape local optima induced by the mode constraint by targeting exploration where the mode constraint's *epistemic uncertainty* is high. Further to this, the GPs ability to model joint distributions over simulated trajectories enabled ModeRL to construct a non-myopic exploration strategy. For planning, ModeRL optimises its closed-form objective, subject to its closed-form constraints, using Sequential Least Squares Quadratic Programming (SLSQP). Our experiments showed that targeting exploration where the mode constraint's *epistemic uncertainty* is high, enables ModeRL's planning algorithm to escape local optima induced by the mode constraint.

Limitations The main limitation of ModeRL is that it is mostly restricted to lower dimensional problems, due to the difficulties of defining GP priors in high dimensions. A potentially unavoidable downside to ModeRL is that it must leave the desired dynamics mode in order to learn about the mode constraint. This was because we used internal sensing to obtain information on the robot, however, in some applications it may be possible to infer the mode constraint using external sensors. For example, in autonomous driving, it may be possible to infer the friction coefficients associated with different road surfaces using cameras without ever leaving the desired dynamics mode.

References

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (2015). *TensorFlow: Large-scale Machine Learning on Heterogeneous Systems*.

Altman, Eitan (1999). *Constrained Markov Decision Processes: Stochastic Modeling*. Routledge.

Berkenkamp, Felix, Matteo Turchetta, Angela Schoellig, and Andreas Krause (2017). "Safe Model-based Reinforcement Learning with Stability Guarantees". In: *Advances in Neural Information Processing Systems*. Vol. 30, pp. 908–918.

Cutler, M. and J. P. How (May 2015). "Efficient Reinforcement Learning for Robots Using Informative Simulated Priors". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 2605–2612.

Deisenroth, M. P., D. Fox, and C. E. Rasmussen (Feb. 2015). "Gaussian Processes for Data-Efficient Learning in Robotics and Control". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.2, pp. 408–423.

Deisenroth, Marc and Carl Rasmussen (Jan. 2011). "PILCO: A Model-Based and Data-Efficient Approach to Policy Search." In: *International Conference on Machine Learning*. Vol. 28, pp. 465–472.

Girard, Agathe, Carl Edward Rasmussen, Joaquin Quinonero-Candela, and Roderick Murray-Smith (2003). "Gaussian Process Priors with Uncertain Inputs? Application to Multiple-Step Ahead Time Series Forecasting". In: *Becker, S*. Vancouver, Canada: MIT Press.

Hensman, James, Nicolo Fusi, and Neil D Lawrence (2013). "Gaussian Processes for Big Data". In: *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*. Vol. 29, pp. 282–290.

Hensman, James, Alexander Matthews, and Zoubin Ghahramani (Feb. 2015). "Scalable Variational Gaussian Process Classification". In: *Artificial Intelligence and Statistics*. PMLR, pp. 351–360.

Hewing, Lukas, Kim P. Wabersich, Marcel Menner, and Melanie N. Zeilinger (2020). "Learning-Based Model Predictive Control: Toward Safe Learning in Control". In: *Annual Review of Control, Robotics, and Autonomous Systems* 3.1, pp. 269–296.

Hoffman, Matthew D., David M. Blei, Chong Wang, and John Paisley (2013). "Stochastic Variational Inference". In: *Journal of Machine Learning Research* 14.4, pp. 1303–1347.

Kamthe, Sanket and Marc Deisenroth (Mar. 2018). "Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1701–1710.

Kingma, Diederik P. and Jimmy Ba (Jan. 2017). "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]*.

Koller, T., F. Berkenkamp, M. Turchetta, and A. Krause (Dec. 2018). "Learning-Based Model Predictive Control for Safe Exploration". In: *2018 IEEE Conference on Decision and Control (CDC)*, pp. 6059–6066.

MacKay, David J. C. (May 1992). "Bayesian Interpolation". In: *Neural Computation* 4.3, pp. 415–447.

Matthews, Alexander G. de G., Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman (Apr. 2017). "GPflow: A Gaussian Process Library Using TensorFlow". In: *Journal of Machine Learning Research* 18.40, pp. 1–6.

Naish-guzman, Andrew and Sean Holden (2008). "The Generalized FITC Approximation". In: *Advances in Neural Information Processing Systems*. Vol. 20. Curran Associates, Inc.

Pan, Yunpeng and Evangelos Theodorou (2014). "Probabilistic Differential Dynamic Programming". In: *Advances in Neural Information Processing Systems*. Vol. 27, pp. 1907–1915.

Pan, Yunpeng, Evangelos Theodorou, and Michail Kontitsis (2015). "Sample Efficient Path Integral Control under Uncertainty". In: *Advances in Neural Information Processing Systems*. Vol. 28.

Quiñonero-Candela, Joaquin and Carl Edward Rasmussen (2005). "A Unifying View of Sparse Approximate Gaussian Process Regression". In: *Journal of Machine Learning Research* 6.65, pp. 1939–1959.

Roy, Julien, Roger Girgis, Joshua Romoff, Pierre-Luc Bacon, and Chris J. Pal (June 2022). "Direct Behavior Specification via Constrained Reinforcement Learning". In: *Proceedings of the 39th International Conference on Machine Learning*. PMLR, pp. 18828–18843.

Scannell, Aidan, Carl Henrik Ek, and Arthur Richards (2021). "Trajectory Optimisation in Learned Multimodal Dynamical Systems Via Latent-ODE Collocation". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE.

Schreiter, Jens, Duy Nguyen-Tuong, Mona Eberts, Bastian Bischoff, Heiner Markert, and Marc Toussaint (2015). "Safe Exploration for Active Learning with Gaussian Processes". In: *Machine Learning and Knowledge Dis-*

covery in Databases. Cham: Springer International Publishing, pp. 133–149.

Sutton, R.S. and A.G. Barto (2018). *Reinforcement Learning, Second Edition: An Introduction*. Adaptive Computation and Machine Learning Series. MIT Press.

Titsias, Michalis (Apr. 2009). "Variational Learning of Inducing Variables in Sparse Gaussian Processes". In: *Artificial Intelligence and Statistics*. PMLR, pp. 567–574.

Tresp, Volker (2000). "Mixtures of Gaussian Processes". In: *Advances in Neural Information Processing Systems*. Vol. 13, pp. 654–660.

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17, pp. 261–272.

A Dynamics Learning

$$p(\Delta \mathbf{s}_{t+1} \mid f_k(\boldsymbol{\zeta}_k)) = \mathbb{E}_{p(f_k(\hat{\mathbf{s}}_t)|f_k(\boldsymbol{\zeta}_k))}\left[p(\Delta \mathbf{s}_{t+1} \mid f_k(\hat{\mathbf{s}}_t))\right] \tag{14a}$$

$$\Pr(\alpha_t = k \mid \mathbf{h}(\boldsymbol{\xi})) = \mathbb{E}_{p(\mathbf{h}(\hat{\mathbf{s}}_t)|\mathbf{h}(\boldsymbol{\xi}))}\left[\Pr\left(\alpha_t = k \mid \mathbf{h}(\hat{\mathbf{s}}_t)\right)\right] \tag{14b}$$

$$p(f_k(\hat{\mathbf{s}}_t) \mid f_k(\boldsymbol{\zeta}_k)) = \mathcal{N}\left(f_k(\hat{\mathbf{s}}_t) \mid \mathbf{k}_{knM}\mathbf{K}_{kMM}^{-1} f_k(\boldsymbol{\zeta}_k), k_{knn} - \mathbf{k}_{knM}\mathbf{K}_{kMM}^{-1}\mathbf{k}_{kMn}\right), \tag{14c}$$

$$p(\mathbf{h}(\hat{\mathbf{s}}_t) \mid \mathbf{h}(\boldsymbol{\xi})) = \prod_{k=1}^{K} \mathcal{N}\left(h_k(\hat{\mathbf{s}}_t) \mid \hat{\mathbf{k}}_{knM}\hat{\mathbf{K}}_{kMM}^{-1} h_k(\boldsymbol{\xi}), \hat{k}_{knn} - \hat{\mathbf{k}}_{knM}\hat{\mathbf{K}}_{kMM}^{-1}\hat{\mathbf{k}}_{kMn}\right), \tag{14d}$$

where $\mathbf{K}_{kMM} = k_k(\boldsymbol{\zeta}_k, \boldsymbol{\zeta}_k)$ represents the k^{th} expert's kernel evaluated between its inducing inputs, $k_{knn} = k_k(\hat{\mathbf{s}}_t, \hat{\mathbf{s}}_t)$ represents it evaluated between the n^{th} training input and $\mathbf{k}_{knM} = k_k(\hat{\mathbf{s}}_t, \boldsymbol{\zeta}_k)$ between the n^{th} training input and its inducing inputs. Similarly for the gating network.

A.1 Predictive posteriors

$= \int \mathcal{N}\left(f_k(\mathbf{s}_t) \mid f_k(\mathbf{s}_t), \boldsymbol{\Sigma}^2_{k^*}(\bar{\mathbf{s}}, \bar{\mathbf{s}})\right)$

$= \mathrm{N}\left(h_{k^*}(\bar{\mathbf{s}}) \mid \boldsymbol{\mu}_{k^*}(\bar{\mathbf{s}}), \boldsymbol{\Sigma}^2_{k^*}(\bar{\mathbf{s}}, \bar{\mathbf{s}})\right)$

The joint distribution over the gating function values $h_{k^*}(\bar{\mathbf{s}})$ is then given by,

$$p(h_{k^*}(\bar{\mathbf{s}}) \mid \bar{\mathbf{s}}, \mathcal{D}_{0:i-1}) \approx q(h_{k^*}(\bar{\mathbf{s}})) = \mathcal{N}\left(h_{k^*}(\bar{\mathbf{s}}) \mid \boldsymbol{\mu}_{k^*}(\bar{\mathbf{s}}), \boldsymbol{\Sigma}^2_{k^*}(\bar{\mathbf{s}}, \bar{\mathbf{s}})\right) \tag{15}$$

where $\boldsymbol{\mu}_{k^*}(\cdot)$ and $\boldsymbol{\Sigma}^2_{k^*}(\cdot, \cdot)$ are sparse GP mean and covariance functions, given by,

$$\boldsymbol{\mu}_{k^*}(\bar{\mathbf{s}}) = \hat{k}_{k^*}(\bar{\mathbf{s}}, \boldsymbol{\xi})\hat{k}_{k^*}(\boldsymbol{\xi}, \boldsymbol{\xi})^{-1}\hat{\mathbf{m}}_{k^*} \tag{16}$$

$$\boldsymbol{\Sigma}^2_{k^*}(\bar{\mathbf{s}}, \bar{\mathbf{s}}) = \hat{k}_{k^*}(\bar{\mathbf{s}}, \bar{\mathbf{s}}) + \hat{k}_{k^*}(\bar{\mathbf{s}}, \boldsymbol{\xi})\hat{k}_{k^*}(\boldsymbol{\xi}, \boldsymbol{\xi})^{-1}\left(\hat{\mathbf{S}}_{k^*} - \hat{k}_{k^*}(\boldsymbol{\xi}, \boldsymbol{\xi})\right)\hat{k}_{k^*}(\boldsymbol{\xi}, \boldsymbol{\xi})^{-1}\hat{k}_{k^*}(\boldsymbol{\xi}, \bar{\mathbf{s}}), \tag{17}$$

where $\hat{k}_{k^*}$ and $\boldsymbol{\xi}$ are the kernel and inducing inputs associated with the desired mode's gating function respectively. This sparse approximation arises because the MoSVGPE model uses sparse GPs and approximates the posterior with,

$$q(h_{k^*}(\bar{\mathbf{s}})) = \int p(h_{k^*}(\bar{\mathbf{s}}) \mid h_{k^*}(\boldsymbol{\xi}))q(h_{k^*}(\boldsymbol{\xi}))\mathrm{d}h_{k^*}(\boldsymbol{\xi}), \tag{18}$$

where $q(h_{k^*}(\boldsymbol{\xi})) = \mathcal{N}\left(h_{k^*}(\boldsymbol{\xi} \mid \hat{\mathbf{m}}_{k^*}, \hat{\mathbf{S}}_{k^*}\right)$.

B Illustrative Example

We test our method on a 2D quadcopter navigation example. See Fig. 5 for a schematic of the environment and details of the problem. The goal is to fly the quadcopter from an initial state $\mathbf{s}_0$, to a target state $\mathbf{s}_f$. However, it considers a quadcopter operating in an environment subject to spatially varying wind – induced by a fan – where two dynamics modes can represent the system,

Mode 1 is an *operable* dynamics mode away from the fan,

Mode 2 is an *inoperable*, turbulent dynamics mode in front of the fan.

The turbulent dynamics mode is subject to higher drift (in the negative x direction) and to higher diffusion (process noise). It is hard to know the exact turbulent dynamics due to complex and uncertain interactions between the quadcopter and the wind field. Further to this, controlling the system in the turbulent dynamics mode may be infeasible. This is because the

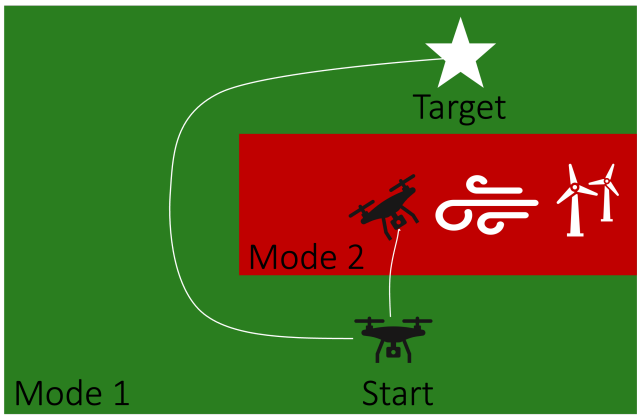

Figure 5: **mode-constrained quadcopter navigation** - Diagram showing a top-down view of a quadcopter subject to two dynamics modes: 1) an *operable* dynamics mode (green) and 2) an *inoperable*, turbulent dynamics mode induced by a strong wind field (red). The goal is to navigate to the target state $\mathbf{s}_f$ (white star), whilst avoiding the turbulent dynamics mode (red).

unpredictability of the turbulence may cause catastrophic failure. Therefore, when flying the quadcopter from an initial state $\mathbf{s}_0$, to a target state $\mathbf{s}_f$, it is desirable to find trajectories that avoid entering this turbulent dynamics mode.

The state-space of the velocity controlled quadcopter example consists of the 2D Cartesian coordinates $\mathbf{s} = (x, y)$ and the controls consist of the speed in each direction, given by $\mathbf{a} = (v_x, v_y)$.

The reward function is given by,

$$r(\mathbf{s}_t, \mathbf{a}_t) = -\left(\mathbf{s}_t - \mathbf{s}_f\right)^T \mathbf{Q} \left(\mathbf{s}_t - \mathbf{s}_f\right) - \mathbf{a}_t^T \mathbf{R} \mathbf{a}_t \tag{19}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are user-defined, real symmetric positive semi definite and positive definite weight matrices respectively. In our experiments we set both $\mathbf{Q}$ and $\mathbf{R}$ to be the identify matrix.

C Experiment Configuration

This section details how the experiments were configured.

Initial data set $\mathcal{D}_0$ The initial data set was collected by simulating 15 random trajectories from the start state $\mathbf{s}_0$ and terminating them when they left the initial state domain $\mathcal{S}_0$. This resulted in an initial data set of 118 state transitions, which is visualised as the blue crosses in **??**.

Model learning Following the experiments in **??** this section instantiated the MoSVGPE dynamics model with $K = 2$ experts, one to represent each of the dynamics modes. Each mode's dynamics GP used a Matern 5/2 kernel with ARD and a constant mean function. The gating network used a single gating function with a Matern 5/2 kernel with ARD and a zero mean function. At each ModeRL iteration, an early stopping callback was used to terminate the dynamics model training. The early stopping callback used a min delta of 0 and a patience of 1200. This meant that training terminated after 1200 epochs of no improvement.

Explorative controller The explorative controller π_{explore} was initialised with a horizon of $T = 15$. At each ModeRL iteration, the previous solution was used as the initial solution for the trajectory optimiser. In all experiments, ModeRL was configured with $\delta = 0.2$, which corresponds to mode chance constraints with a satisfaction probability of 0.8. That is, the mode chance constraints were given by,

$$\Pr(\alpha_t = k^* \mid \mathbf{s}_0, \mathbf{a}_{0:t}, \boldsymbol{\alpha}_{0:t-1} = k^*) \geq 0.8 \quad \forall \in \{0, \ldots, T-1\}. \tag{20}$$

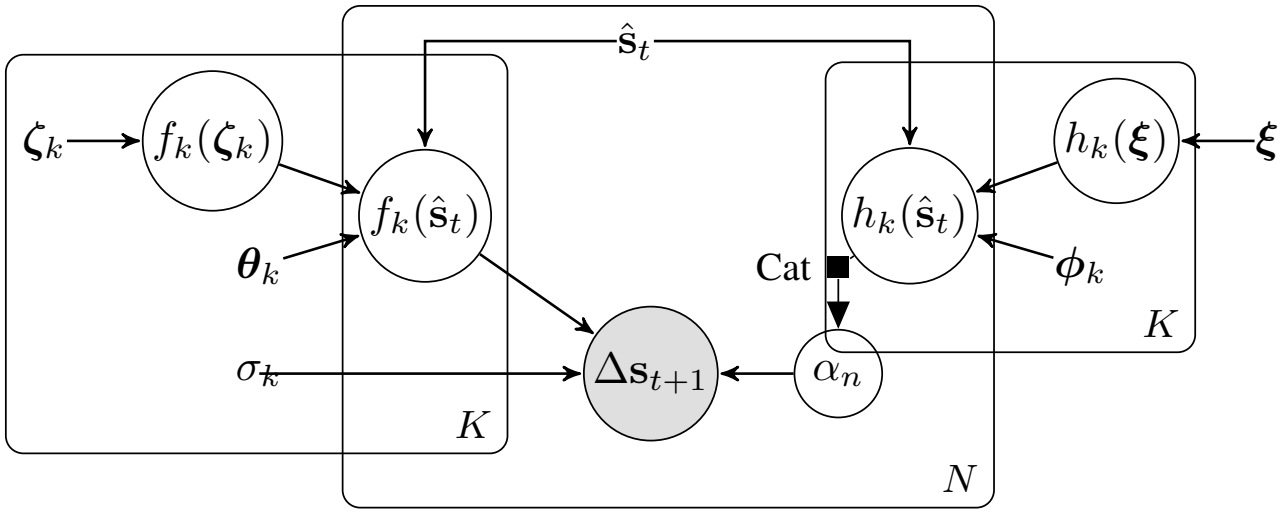

Figure 6: Graphical models where the outputs $\Delta\mathbf{S} = \{\Delta\mathbf{S}_k\}_{k=1}^K$ are generated by mapping the inputs $\hat{\mathbf{S}} = \{\hat{\mathbf{S}}_k\}_{k=1}^K$ through the latent processes. An input assigned to expert k is denoted $\mathbf{x}_{n,k}$ and the sets of all N_k inputs and outputs assigned to expert k are denoted $\hat{\mathbf{S}}_k$ and $\Delta\mathbf{S}_k$ respectively. The experts are shown on the left and the gating network on the right. The generative processes involve evaluating the gating network and sampling an expert mode indicator variable α_t. The indicated mode's latent function f_k and noise model $\mathcal{N}(0, \sigma_k)$ are then evaluated to generate the output $\Delta\mathbf{s}_{t+1}$.

D Dynamics Learning

$$\mathbf{s}_{t+1} \mid f_k(\hat{\mathbf{s}}_t) \sim \mathcal{N}\left(f_k(\hat{\mathbf{s}}_t), \sigma_k^2\right) \tag{21}$$

$$f_k(\hat{\mathbf{S}}) \sim \mathcal{GP}\left(\mu_k(\hat{\mathbf{S}}), k_k(\hat{\mathbf{S}}, \hat{\mathbf{S}})\right) \tag{22}$$

$$\alpha_t \mid \mathbf{h}(\hat{\mathbf{s}}_t) \sim \text{Categorical}\left(K \mid \text{softmax}(\mathbf{h}(\hat{\mathbf{s}}_t))\right) \tag{23}$$

$$\mathbf{h}(\hat{\mathbf{S}}) \sim \mathcal{GP}\left(\hat{\mu}_k(\hat{\mathbf{S}}, \hat{k}_k(\hat{\mathbf{S}}, \hat{\mathbf{S}})\right) \tag{24}$$

The first stage of our method formulates the single-step dynamics in Eq. (1) as a Mixtures of Gaussian Process Experts (MoGPE) model and casts the learning problem to probabilistic inference. We leverage the delta state formulation to regularise the predictive distribution and we simplify notation by denoting a state-control input as $\hat{\mathbf{s}}_t = (\mathbf{s}_t, \mathbf{a}_t)$ and a state difference output as $\Delta\mathbf{s}_{t+1} = \Delta\mathbf{s}_{t+1}$.

D.1 Generative Model

We model each dynamics mode (referred to as an expert in MoGPE terminology) as a Gaussian process regression model,

$$\underbrace{p\left(\Delta\mathbf{S}_k \mid \hat{\mathbf{S}}_k\right)}_{\text{expert } k} = \underbrace{\mathbb{E}_{p(f_k(\hat{\mathbf{S}}_k))}}_{\text{GP prior}}\left[\prod_{n=1}^{N_k} \underbrace{p\left(\Delta\mathbf{s}_{t+1} \mid f_k(\hat{\mathbf{s}}_t)\right)}_{\text{Gaussian likelihood}}\right] \tag{25}$$

$$p(f_k(\hat{\mathbf{S}}_k)) = \mathcal{N}\left(f_k(\hat{\mathbf{S}}_k) \mid \mu_k(\hat{\mathbf{S}}_k), k_k(\hat{\mathbf{S}}_k, \hat{\mathbf{S}}_k)\right), \tag{26}$$

where $\mu_k(\cdot)$ and $k_k(\cdot, \cdot)$ represent the mean and covariance functions associated with the k^{th} mode's GP prior respectively. Note that $\hat{\mathbf{S}}_k$ denotes the set of N_k inputs assigned to mode k, i.e. $\hat{\mathbf{S}}_k = \{\hat{\mathbf{s}}_t \in \hat{\mathbf{S}} \mid \alpha(\hat{\mathbf{s}}_t) = k\}$. Similarly for the outputs we have $\Delta\mathbf{S}_k = \{\Delta\mathbf{s}_{t+1} \in \Delta\mathbf{S} \mid \alpha(\hat{\mathbf{s}}_t) = k\}$. However, as the assignment of observations to modes is *not known a priori*, we must infer the assignments from observations.

MoGPE models are a natural choice for modelling multimodal systems as they automatically infer the assignments from

observations. The MoGPE marginal likelihood is given by,

$$p\left(\Delta\mathbf{S} \mid \hat{\mathbf{S}}\right) = \sum_{\boldsymbol{\alpha}} \underbrace{p\left(\boldsymbol{\alpha} \mid \hat{\mathbf{S}}\right)}_{\text{gating network}} \left[\prod_{k=1}^{K} \underbrace{p\left(\Delta\mathbf{S}_k \mid \hat{\mathbf{S}}_k, \boldsymbol{\theta}_k\right)}_{\text{expert } k}\right] \tag{27}$$

where the gating network assigns observations to experts. Note that there is a joint distribution corresponding to every possible combination of assignments of observations to experts. Hence, Eq. (27) is a sum over exponentially many (K^N) sets of assignments, where $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_N\}$ represents a set of assignments for all observations. This distribution factors into the product over experts, where each expert models the joint Gaussian distribution over the observations assigned to it. **??** shows its graphical model representation.

Gating network The gating network governs which expert is responsible for predicting at a given input location. That is, it models which dynamics mode governs the system at different regions of the input-space. Motivated by synergising model learning and control, we formulate our gating network using input-dependent functions – known as gating functions – and place GP priors over them. In **??** we exploit our GP-based gating network to encode explorative behaviour into our trajectory optimisation algorithm. Similar to Tresp, 2000, our gating network resembles a GP classification model,

$$p\left(\boldsymbol{\alpha} \mid \hat{\mathbf{S}}\right) = \mathbb{E}_{\underbrace{p\left(\mathbf{h}(\hat{\mathbf{S}})\right)}_{\text{GP prior(s)}}} \left[\prod_{n=1}^{N} \underbrace{p\left(\alpha_t \mid \mathbf{h}(\hat{\mathbf{s}}_t)\right)}_{\text{Softmax/Bernoulli likelihood}}\right] \tag{28}$$

where $p\left(\alpha_t \mid \mathbf{h}(\hat{\mathbf{s}}_t)\right)$ represents a classification likelihood parameterised by K gating functions $\mathbf{h} : \mathcal{X} \to \mathbb{R}^K$. We use a Bernoulli likelihood when $K = 2$ and a softmax when $K > 2$. GP priors are placed on each of the latent gating functions $p(\mathbf{h}(\hat{\mathbf{S}})) = \prod_{k=1}^{K} \mathcal{N}\left(p(\mathbf{h}(\hat{\mathbf{S}}) \mid \hat{\mu}_k(\hat{\mathbf{S}}), \hat{k}_k(\hat{\mathbf{S}}, \hat{\mathbf{S}})\right)$, where $\hat{\mu}_k(\cdot)$ and $\hat{k}_k(\cdot, \cdot)$ represent the mean and covariance functions associated with the k^{th} gating function.

Each mode's mixing probability $\Pr\left(\alpha_t = k \mid \hat{\mathbf{s}}_t, \boldsymbol{\phi}\right)$ is then obtained by marginalising **all** of the gating functions. In the general case where $\Pr\left(\alpha_t = k \mid \mathbf{h}(\hat{\mathbf{s}}_t)\right)$ uses the softmax function (**??**) this integral is intractable, so it is approximated with Monte Carlo quadrature.

D.2 Approximate Inference

Exact inference in our model is intractable due to the marginalisation over the set of expert indicator variables. For this reason, we resort to a variational approximation based on inducing variables, that provides scalability by utilising stochastic gradient-based optimisation. The marginalisation over the set of expert indicator variables $\boldsymbol{\alpha}$ in Eq. (27) is prohibitive to SVI (Hoffman et al., 2013). This is because it requires a set of local variables factorised over data but the marginalisation considers the sets of assignments for the entire data set. Following the approach by Titsias, 2009, we augment the probability space with a set of M inducing variables for each GP. However, instead of collapsing these inducing variables, we explicitly represent them as variational distributions and use them to lower bound the marginal likelihood, similar to Hensman et al., 2013, 2015. Fig. 6 shows the graphical model of the augmented joint probability space.

Augmented experts We sidestep the hard assignment of observations to experts by augmenting each expert with a set of separate independent inducing points $(\boldsymbol{\zeta}_k, f_k(\boldsymbol{\zeta}_k))$. Each expert's inducing points are assumed to be from its GP prior,

$$p(f_k(\boldsymbol{\zeta}_k)) = \mathcal{N}\left(f_k(\boldsymbol{\zeta}_k) \mid \mu_k(\boldsymbol{\zeta}_k), k_k(\boldsymbol{\zeta}_k, \boldsymbol{\zeta}_k)\right), \tag{29}$$

and the distribution over all experts is given by $p(\mathbf{f}(\boldsymbol{\zeta})) = \prod_{k=1}^{K} p(f_k(\boldsymbol{\zeta}_k))$, where the set of all inducing inputs associated with the experts has been denoted $\boldsymbol{\zeta}$ and the set of all inducing variables as $\mathbf{f}(\boldsymbol{\zeta})$. Note that the dependence on the inducing inputs has been dropped for notational conciseness. Introducing separate inducing points from each expert's GP can loosely be seen as "partitioning" the observations between experts. However, as the assignment of observations to experts is *not known a priori*, the inducing inputs $\boldsymbol{\zeta}_k$ and variables $f_k(\boldsymbol{\zeta}_k)$, must be inferred from observations.

Augmented gating network Following a similar approach for the gating network, each gating function is augmented with a set of M inducing points from its corresponding GP prior,

$$p(h_k(\boldsymbol{\xi})) = \mathcal{N}\left(h_k(\boldsymbol{\xi}) \mid \hat{\mu}_k(\boldsymbol{\xi}), \hat{k}_k(\boldsymbol{\xi}, \boldsymbol{\xi})\right), \tag{30}$$

where $h_k(\boldsymbol{\xi})$ are the inducing variables associated with the k^{th} gating function. The distribution over all gating functions is denoted $p(\mathbf{h}(\boldsymbol{\xi})) = \prod_{k=1}^{K} p(h_k(\boldsymbol{\xi}))$. Again, the dependence on the inducing inputs has been dropped for notational conciseness. The set of all inducing variables associated with the gating network has been denoted $\mathbf{h}(\boldsymbol{\xi})$. Partitioning of the data set is not desirable for the gating function GPs as each gating function should depend on all of the training observations. For this reason, the gating functions share the same inducing inputs $\boldsymbol{\xi}$.

Marginal likelihood These inducing points are used to approximate the marginal likelihood with a factorisation over observations that is favourable for constructing a GP-based gating network. Our approximate marginal likelihood is given by,

$$p\left(\Delta\mathbf{S} \mid \hat{\mathbf{S}}\right) \approx \mathbb{E}_{p(\mathbf{h}(\boldsymbol{\xi}))p(\mathbf{f}(\boldsymbol{\zeta}))}\left[\prod_{n=1}^{N}\sum_{k=1}^{K}\Pr(\alpha_t = k \mid \mathbf{h}(\boldsymbol{\xi})) \right. \\ \left. p(\Delta\mathbf{s}_{t+1} \mid f_k(\boldsymbol{\zeta}_k))\right], \tag{31}$$

where the conditional distributions $p(\Delta\mathbf{s}_{t+1} \mid f_k(\boldsymbol{\zeta}_k))$ and $\Pr(\alpha_t = k \mid \mathbf{h}(\boldsymbol{\xi}))$ follow from standard sparse GP methodologies. Fig. 6 shows the graphical model of this augmented joint probability model.

Our approximation assumes that given the inducing points, the marginalisation over every possible assignment of data points to experts can be factorised over data. In a similar spirit to the Fully Independent Training Conditional (FITC) approximation (Naish-guzman and Holden, 2008; Quiñonero-Candela and Rasmussen, 2005), this can be viewed as a likelihood approximation,

$$p(\Delta\mathbf{S} \mid \mathbf{f}(\boldsymbol{\zeta})) \approx \prod_{n=1}^{N}\sum_{k=1}^{K}\Pr\left(\alpha_t = k \mid \hat{\mathbf{s}}_t, \boldsymbol{\phi}\right)\prod_{k=1}^{K} p(\Delta\mathbf{s}_{t+1} \mid f_k(\boldsymbol{\zeta}_k)). \tag{32}$$

Importantly, the factorisation over observations has been moved outside of the marginalisation over the expert indicator variable, i.e. the expert indicator variable can be marginalised for each data point separately. This approximation assumes that the inducing variables, $\{f_k(\boldsymbol{\zeta}_k)\}_{k=1}^{K}$, are a sufficient statistic for their associated latent function values, $\{f_k(\hat{\mathbf{S}}_k)\}_{k=1}^{K}$ and the set of assignments $\boldsymbol{\alpha}$. This approximation becomes exact in the limit $KM = N$, if each expert's inducing points represent the true data partition $\{\boldsymbol{\zeta}_k, f_k(\boldsymbol{\zeta}_k)\}_{k=1}^{K} = \{\hat{\mathbf{S}}_k, f_k(\hat{\mathbf{S}}_k)\}_{k=1}^{K}$. It is also worth noting that Eq. (31) captures a rich approximation of each expert's covariance but as $KM \ll N$ the computational complexity is much lower. This approximation efficiently couples the gating network and the experts by marginalising the expert indicator variable for each data point separately.

Our approximate marginal likelihood captures the joint distribution over the data and assignments through the inducing variables $\mathbf{f}(\boldsymbol{\zeta})$. As such, information regarding the assignment of observations to experts must pass through the bottleneck of the inducing variables. This approximation induces a local factorisation over observations and a set of global variables – the necessary conditions for SVI. This approach can loosely be viewed as parameterising the full nonparametric model in **??** to obtain a desirable factorisation for 1) constructing a GP-based gating network and 2) deriving an ELBO that can be optimised with stochastic gradient methods, whilst still capturing the complex dependencies between the gating network and experts.

D.2.1 Evidence Lower Bound

Instead of collapsing the inducing variables as seen in Titsias, 2009, they can be explicitly represented as variational distributions and used to obtain a variational lower bound, aka Evidence Lower Bound (ELBO). Following a similar approach to Hensman et al., 2013, 2015, a lower bound on Eq. (31) can be obtained,

$$\mathcal{L} = \sum_{n=1}^{N}\mathbb{E}_{q(\mathbf{h}(\hat{\mathbf{s}}_t))q(\mathbf{f}(\boldsymbol{\zeta}))}\left[\log\sum_{k=1}^{K}\Pr\left(\alpha_t = k \mid \mathbf{h}(\hat{\mathbf{s}}_t)\right)p(\Delta\mathbf{s}_{t+1} \mid f_k(\boldsymbol{\zeta}_k))\right] \\ -\sum_{k=1}^{K}\mathrm{KL}\left(q(f_k(\boldsymbol{\zeta}_k)) \mid\mid p(f_k(\boldsymbol{\zeta}_k))\right) - \sum_{k=1}^{K}\mathrm{KL}\left(q(h_k(\boldsymbol{\xi})) \mid\mid p(h_k(\boldsymbol{\xi}))\right)$$

where each experts' variational posterior is given by $q(f_k(\boldsymbol{\zeta}_k)) = \mathcal{N}\left(f_k(\boldsymbol{\zeta}_k) \mid \mathbf{m}_k, \mathbf{S}_k\right)$ and the gating network's variational posterior is given by $q(h_k(\boldsymbol{\xi})) = \int p(h_k(\hat{\mathbf{s}}_t) \mid h_k(\boldsymbol{\xi}))\mathcal{N}\left(h_k(\boldsymbol{\xi}) \mid \hat{\mathbf{m}}_k, \hat{\mathbf{S}}_k\right)\mathrm{d}\mathbf{h}(\boldsymbol{\xi})$.

$$q(\mathbf{f}(\boldsymbol{\zeta})) = \prod_{k=1}^{K} q(f_k(\boldsymbol{\zeta}_k)) = \prod_{k=1}^{K} \mathcal{N}\left(f_k(\boldsymbol{\zeta}_k) \mid \mathbf{m}_k, \mathbf{S}_k\right) \tag{33}$$

$$q(\mathbf{h}(\hat{\mathbf{s}}_t)) = \prod_{k=1}^{K} \mathbb{E}_{q(h_k(\boldsymbol{\xi}))}\left[p(h_k(\hat{\mathbf{s}}_t) \mid h_k(\boldsymbol{\xi}))\right] \tag{34}$$

$$q(\mathbf{h}(\hat{\mathbf{s}}_t)) = \prod_{k=1}^{K} \int p(h_k(\hat{\mathbf{s}}_t) \mid h_k(\boldsymbol{\xi})) q(h_k(\boldsymbol{\xi})) \mathrm{d}\mathbf{h}(\boldsymbol{\xi}). \tag{35}$$

w the functional form of the gating networks variational posterior is given by$q(h_k(\boldsymbol{\xi}))q(h_k(\boldsymbol{\xi})) = \mathcal{N}\left(h_k(\boldsymbol{\xi}) \mid \hat{\mathbf{m}}_k, \hat{\mathbf{S}}_k\right)$. Moving the marginalisation over the latent gating functions $\mathbf{h}(\hat{\mathbf{s}}_t)$ outside of the marginalisation over the expert indicator variable is possible because the mixing probabilities are dependent on **all** of the gating functions and not just their associated gating function. In contrast, moving the marginalisation over each expert's latent function $f_k(\hat{\mathbf{s}}_t)$ outside of the marginalisation over the expert indicator variable corresponds to changing the underlying model, in particular, the likelihood approximation in Eq. (32).

The bound in **??** meets the necessary conditions to perform stochastic gradient methods on $q(\mathbf{f}(\boldsymbol{\zeta}))$ and $q(\mathbf{h}(\boldsymbol{\xi}))$, as the expected log likelihood (first term) is written as a sum over input-output pairs. However, this expectation cannot be calculated in closed form and must be approximated. The joint distribution over the inducing variables for each expert GP $q(f_k(\boldsymbol{\zeta}_k))$ are M dimensional multivariate normal distributions. Therefore, each expectation requires an M dimensional integral to be approximated. In contrast, the gating network's variational posterior marginalises the inducing variables in closed form, removing K of the undesirable approximate M dimensional integrals from **??**. The variational expectation in **??** still requires approximation, however, the gating network integrals are now only one-dimensional. We approximate all of the integrals with Gibbs sampling and in practice use only single samples because the added stochasticity helps the optimisation.

Optimisation The bound in **??** meets the necessary conditions to perform stochastic gradient methods on $q(\mathbf{f}(\boldsymbol{\zeta}))$ and $q(\mathbf{h}(\boldsymbol{\xi}))$. Firstly, they contain a sum of N terms corresponding to input-output pairs, enabling optimisation with mini-batches. Secondly, the expectations over the log-likelihood are calculated using Monte Carlo samples. We treat the inducing inputs $\boldsymbol{\xi}, \{\boldsymbol{\zeta}_k\}_{k=1}^{K}$, kernel hyperparameters and noise variances, as variational hyperparameters and optimise them alongside the variational parameters, using Adam (Kingma and Ba, 2017).