Machine Learning (2022) 111:173–203
https://doi.org/10.1007/s10994-021-06103-6

SAMBA: safe model-based & active reinforcement learning

Alexander I. Cowen-Rivers[1,2] · **Daniel Palenicek**[1,2] · **Vincent Moens**[1] · **Mohammed Amin Abdullah**[1] · **Aivar Sootla**[1] · **Jun Wang**[1,3] · **Haitham Bou-Ammar**[1,3]

Received: 23 October 2020 / Revised: 6 September 2021 / Accepted: 11 October 2021 /
Published online: 4 January 2022
© The Author(s), under exclusive licence to Springer Science+Business Media LLC, part of Springer Nature 2022

Abstract
In this paper, we propose SAMBA, a novel framework for safe reinforcement learning that combines aspects from probabilistic modelling, information theory, and statistics. Our method builds upon PILCO to enable active exploration using novel acquisition functions for out-of-sample Gaussian process evaluation optimised through a multi-objective problem that supports conditional-value-at-risk constraints. We evaluate our algorithm on a variety of safe dynamical system benchmarks involving both low and high-dimensional state representations. Our results show orders of magnitude reductions in samples and violations compared to state-of-the-art methods. Lastly, we provide intuition as to the effectiveness of the framework by a detailed analysis of our acquisition functions and safety constraints.

Keywords Gaussian process · Safe reinforcement learning · Active learning

1 Introduction

Reinforcement learning (RL) has seen successes in many problems such as video and board games (Mnih et al. 2013; Silver et al. 2016, 2017; Mnih et al. 2015), and control of simulated robots (Ammar et al. 2014; Schulman et al. 2015, 2017). Though successful, these applications assume idealised simulators and require tens of millions of agent-environment interactions, typically performed by randomly exploring policies. However, on the time scales of physical (i.e., real-world) systems, sample-efficiency naturally becomes a more pressing concern due to time and cost burdens. Additionally, there are often important

A. I. Cowen-Rivers, D. Palenicek, V. Moens: The first three authors are joint first authors.
H. Bou-Ammar: Honorary position at UCL.
Editors: Dana Drachsler Cohen, Javier García, Mohammad Ghavamzadeh, Marek Petrik, Philip S. Thomas.

✉ Alexander I. Cowen-Rivers
alexander.cowen.rivers@huawei.com

[1] Huawei Noah's Ark Lab, London, United Kingdom

[2] Technical University Darmstadt, Darmstadt, Germany

[3] University College London, London, United Kingdom

safety considerations that one would like to integrate into the behaviour of the system. These can often be formalised through constraints on system trajectories (sequences of state and action pairs). For example, in autonomous drone navigation, one may desire to avoid sharp manoeuvres in cluttered environments, especially where there are people. A safety constraint can be integrated into the control optimisation formulation by quantifying the risk of safety violation by encoding a formal risk measure over the sequence of states (distance, velocity, acceleration, etc.), and actions (actuator control signals). Thus, taken together, the need for RL algorithms to integrate safety and sample-efficiency into a single coherent framework for real-world settings becomes clear.

To elaborate further on the issue of safety, there is a distinction to be made between requiring that a controlling agent/policy is safe (however that may be defined), and requiring that the learning process itself is safe. The latter generally requires assumptions to be made about the dynamics of the system as well as starting conditions (such as access to a safe initial policy). For example, safety defined by constraining trajectories to safe regions of the state-action space is studied in (Akametalu et al. 2014) and partially-known control-affine dynamics with Lipschitz regularity conditions are assumed. In Aswani et al. (2013) and Koller et al. (2018), strong assumptions on dynamics and initial control policies are required to give theoretical guarantees of safety. Safety in terms of Lyapunov stability (Khalil and Grizzle 2002) is studied in a model-free setting in Chow et al. (2018, 2019), which require a safe initial policy. Likewise, Lyapunov stability in a model-based setting is considered in Berkenkamp et al. (2016), which requires Lipschitz dynamics and a safe initial policy. In Dalal et al. (2018) a decision layer is added on top of the policy in order to perturb the actions chosen by the policy so as to remain within safety constraints. The safety layer relies on the dynamics being linearisable, and to be learnt by, e.g., a neural network prior to policy training.

Whilst theoretical guarantees of safety are clearly appealing, the burden of extra assumptions and requirements needed is one we seek to avoid in this paper. As a trade-off, we are willing to tolerate some constraint violation during the learning process with the benefit of not requiring stringent assumptions on dynamics or a safe initial policy.

With regard to the form of the constraint, expectation (i.e., risk-neutral) constraints, which have a long history in the MDP literature (Altman 1999), have been studied in a number of recent RL papers (e.g., Achiam et al. 2017; Dalal et al. 2018; Chow et al. 2018, 2019). Whilst expectation constraints are sufficient for some purposes, they are somewhat limited as a risk measure. A more general risk measure, and the one we exploit in this paper, is Conditional Value-at-Risk (CVaR) (Rockafellar and Uryasev 2000), which is used in a number of RL papers, such as Chow et al. (2017) (see discussion in Sect. 2.1). However, their approach is based on model-free methods for finite MDPs, and in particular, is not suitable for the continuous control use-cases considered in this paper.

Following the above discussion, in this paper we develop SAMBA, a model-based (deep) RL algorithm suitable for learning control in continuous state and action spaces, and which incorporates CVaR as a safety constraint. SAMBA exploits Gaussian processes (Rasmussen and Williams 2005) for dynamics learning, similar to PILCO (Deisenroth and Rasmussen 2011), but does not adopt the moment-matching approach that PILCO takes in order predict forward-dynamics. Instead SAMBA performs policy updates by sampling trajectories from learned dynamics and applying standard deep RL techniques (policy gradient updates in an actor-critic framework). In the context of safe GP-based RL we must mention Polymenakos et al. (2019, 2020), which still relies on moment matching to approximate the policy gradient, while our design relies on a stochastic gradient sampling strategy based upon simulations of the learnt dynamics. Furthermore, we also leverage *active* learning (Gal et al. 2017) to promote exploration near the starting state (Sect. 2.2), wherein the agent is influenced to acquire states reducing model uncertainty.

Successful application of active learning is very much predicated upon the chosen method of quantifying potential uncertainty reduction, i.e., what we refer to as the *acquisition function*. Common acquisition functions are those which identify points in the model with large entropy or large variance, or where those quantities would be most reduced in a posterior model if the given point were added to the data set (Krause and Guestrin 2007; Krause et al. 2008; Fedorov 2013; Settles 2009). However, our desire for safety adds a complication, since a safe learning algorithm will likely have greater uncertainty in regions where it is unsafe by virtue of not exploring those regions (Deisenroth and Rasmussen 2011; Kamthe and Deisenroth 2018). Attacking the above challenges, we propose two novel acquisition functions for Gaussian processes that allow for exploration in novel areas while remaining close to training data, thus avoiding unsafe regions. To enable effective and grounded introduction of active exploration and safety constraints, we define a novel constrained bi-objective formulation of RL and provide a policy multi-gradient solver that is proven effective on a variety of safety benchmarks.

In short, our contributions can be stated as follows: (1) We define a novel constrained bi-objective formulation that trades off cost minimisation, exploration maximisation, and constraint feasibility. (2) We present a model-based deep RL algorithm for this constrained bi-objective formulation that is suitable for continuous state and action spaces and which exploits Gaussian processes. (3) We define safety-aware acquisition functions for exploration. We test our algorithm on three stochastic dynamical systems after augmenting these with safety regions and demonstrate a significant reduction in sample and cost complexities compared to the state-of-the-art.

As a final note, an alternative approach to using reinforcement learning would be to use model predictive control (MPC) (Camacho and Alba 2013) instead of computing a policy (i.e., general map from states to actions), and this is the method of some of the aforementioned papers (Koller et al. 2018; Berkenkamp et al. 2017; Aswani et al. 2013; Kamthe and Deisenroth 2018). However, it is typically argued that an MPC controller is less robust than a policy controller (Mayne et al. 2005). In fact, robust MPC methods typically use a combination of a simple policy to ensure some form of robustness and an action plan to guarantee constraint satisfaction. Consequently, we chose to use a policy-based controller.

2 Background and notation

2.1 Reinforcement learning

We consider Markov decision processes (MDPs) with continuous states and action spaces; $\mathcal{M} = \langle \mathcal{X}, \mathcal{U}, \mathcal{P}, c, \gamma \rangle$, where $\mathcal{X} \subseteq \mathbb{R}^{d_{\text{state}}}$ denotes the state space, $\mathcal{U} \subseteq \mathbb{R}^{d_{\text{act}}}$ the action space, $\mathcal{P} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to [0, \infty)$ is a transition density function, $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the cost function and $\gamma \in [0, 1]$ is a discount factor. At each time step $t = 0, \dots, T$, the agent is in state $\boldsymbol{x}_t \in \mathcal{X}$ and chooses an action $\boldsymbol{u}_t \in \mathcal{U}$ transitioning it to a successor state $\boldsymbol{x}_{t+1} \sim \mathcal{P}(\boldsymbol{x}_{t+1} | \boldsymbol{x}_t, \boldsymbol{u}_t)$, and yielding a cost $c(\boldsymbol{x}_t, \boldsymbol{u}_t)$. Given a state $\boldsymbol{x}_t$, an action $\boldsymbol{u}_t$ is sampled from a policy $\pi : \mathcal{X} \times U \to [0, \infty)$, where we write $\pi(\boldsymbol{u}_t | \boldsymbol{x}_t)$ to represent the conditional density of an action. Upon subsequent interactions, the agent collects a trajectory $\boldsymbol{\tau} = [\boldsymbol{x}_{0:T}, \boldsymbol{u}_{0:T}]$, and aims to determine an optimal policy $\pi^\star$ by minimising total expected cost: $\pi^\star \in \arg\min_\pi \mathbb{E}_{\boldsymbol{\tau} \sim p_\pi(\boldsymbol{\tau})}[\mathcal{C}(\boldsymbol{\tau}) := \sum_{t=0}^{T} \gamma^t c(\boldsymbol{x}_t, \boldsymbol{u}_t)]$, where $p_\pi(\boldsymbol{\tau})$ denotes the trajectory density defined as: $p_{\pi}(\boldsymbol{\tau}) = \mu_0(\boldsymbol{x}_0)\prod_{t=0}^{T-1}\mathcal{P}(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t,\boldsymbol{u}_t)\pi(\boldsymbol{u}_t|\boldsymbol{x}_t)$, with $\mu_0(\cdot)$ being an initial state distribution.

Constrained MDPs The above can be generalised to include various forms of constraints, often motivated by the desire to impose some form of safety measures. Examples are expectation constraints (Achiam et al. 2017; Altman 1999) (which have the same form as the objective, i.e., expected discounted sum of costs), constraints on the variance of the return (Prashanth and Ghavamzadeh 2013), chance constraints (a.k.a. Value-at-Risk (VaR)) (Chow et al. 2017), and Conditional Value-at-Risk (CVaR) (Chow and Ghavamzadeh 2014; Chow et al. 2017; Prashanth 2014). The latter is the constraint we adopt in this paper for reasons that will be elucidated upon below. Adding constraints means we cannot directly apply standard algorithms like policy gradient (Sutton and Barto 2018), and different techniques are required, e.g., via Lagrange multipliers (Bertsekas 1997), as was done in (Chow and Ghavamzadeh 2014; Chow et al. 2017; Prashanth 2014) besides many others. Further, current methods only consider cost minimisation with no regard to exploration as we do in this paper.

Model-Based Reinforcement Learning Current solutions to the problem described above (constrained or unconstrained) can be split into model-free and model-based methods. Though effective, model-free algorithms are highly sample inefficient (Hessel et al. 2018). For sample-efficient solvers, we follow model-based strategies that we now detail. To reduce the number of interactions with the real environments, model-based solvers build surrogate models, $\mathcal{P}_{\text{surr}}$, to determine optimal policies. These methods, typically, run two main loops. The first gathers traces from the real environment to update $\mathcal{P}_{\text{surr}}$, while the second improves the policy using $\mathcal{P}_{\text{surr}}$ (Deisenroth and Rasmussen 2011; Hafner et al. 2020). Among various candidate models, e.g., world models (Ha and Schmidhuber 2018), in this paper, we follow PILCO (Deisenroth and Rasmussen 2011) and adopt Gaussian processes (GPs) as we believe that uncertainty quantification and sample efficiency are key for real-world considerations of safety. In this construction, one places a Gaussian process prior on a latent function f to map between input-output pairs. Such a prior is fully specified by a mean, $m(\boldsymbol{x}) = \mathbb{E}\left[f(\boldsymbol{x})\right]$, and a covariance function $k(\boldsymbol{x},\boldsymbol{x}') = \mathbb{E}\left[(f(\boldsymbol{x})-m(\boldsymbol{x}))(f(\boldsymbol{x}')-m(\boldsymbol{x}'))\right]$ (Rasmussen and Williams 2005). We write $f \sim \mathcal{GP}(m(\cdot),k(\cdot,\cdot))$ to emphasize that f is sampled from a GP (Rasmussen and Williams 2005). Given a data-set of n_1 input-output pairs $\{(\boldsymbol{x}^{(i)},y^{(i)})\}_{i=1}^{n_1}$, corresponding, respectively, to state-action and successor state tuples, one can perform predictions on a query set of n_2 test data points $\{\boldsymbol{x}_{\star}^{(j)}\}_{j=1}^{n_2}$. Such a distribution is Gaussian with predictive mean-vectors and covariance matrices given by: $\boldsymbol{\mu}_{\star} = \boldsymbol{K}_{n_2,n_1}\boldsymbol{A}_{n_1,n_1}\boldsymbol{y}_{n_1}$ and $\boldsymbol{\Sigma}_{n_2,n_2} = \boldsymbol{K}_{n_2,n_2} - \boldsymbol{K}_{n_2,n_1}\boldsymbol{A}_{n_1,n_1}\boldsymbol{K}_{n_1,n_2}$, where $\boldsymbol{A}_{n_1,n_1} = [\boldsymbol{K}_{n_1,n_1} + \sigma_{\omega}^2\boldsymbol{I}]^{-1}$ with σ_{ω} being the noise covariance that is assumed to be Gaussian. In the above, we also defined $\boldsymbol{y}_{n_1}$ as a vector concatenating all training labels, $\boldsymbol{K}_{n_1,n_1} = K(\boldsymbol{X},\boldsymbol{X}) = [k(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)})]_{ij}$, $\boldsymbol{K}_{n_1,n_2} = \boldsymbol{K}_{n_2,n_1}^{\top} = K(\boldsymbol{X},\boldsymbol{X}_{\star}) = [k(\boldsymbol{x}^{(i)},\boldsymbol{x}_{*}^{(j)})]_{ij}$, and $\boldsymbol{K}_{n_2,n_2}(\boldsymbol{X}_{\star},\boldsymbol{X}_{\star}) = [k(\boldsymbol{x}_{*}^{(i)},\boldsymbol{x}_{*}^{(j)})]_{ij}$, where $\boldsymbol{X}$ and $\boldsymbol{X}_{\star}$ are feature matrices with #input-dim $\times\, n_1$ and #input-dim $\times\, n_2$ sizes respectively. We executed training in GPyTorch (Gardner et al. 2018), and used multi-output-GPs as defined in (de Wolff et al. 2020).

2.2 Active learning in dynamical systems

In *active learning* (Fedorov 2013; Settles 2009), an agent chooses points to sample/query that best improve learning or model updates. This is often performed by optimising an *acquisition function*, which gives some quantification of how much a model would improve if a given data point were queried, e.g., points where the model has high entropy or where variance can be most reduced. Active learning with GPs has been studied in the static case, where points can be selected at will (see, e.g., (Krause and Guestrin 2007; Krause et al. 2008)). In the context of dynamical systems, however, added complications arise as one is not always able to directly drive the system into a desired state. Recent work has attempted to resolve this problem, e.g., in (Buisson-Fenet et al. 2019) and (Schultheis et al. 2019), receding horizon optimisation is used to iteratively update a model, and in (Buisson-Fenet et al. 2019), actions are favoured that maximise the sum of differential entropy terms at each point in the mean trajectory predicted to occur by those actions. Moreover, in (Schultheis et al. 2019), a sum of variance terms is optimised to improve Bayesian linear regression. Again, for computational tractability, the predicted mean of states is used as propagating state distributions in the model is difficult. Different to our paper, neither of these works deal with safety, nor do they have additional objectives to maximise/minimise. In (Jain et al. 2018) a GP model that is used for MPC is updated by greedily selecting points which maximise *information gain*, i.e., reduction in entropy, as is done in Krause et al. (2008). Only very recently, Ball et al. (2020) proposed an active learning approach coupled with MBRL. Similar to SAMBA, they use an adaptive convex combination of objectives, however their acquisition function is based on reward variance computed from a (finite) collection of models increasing the burden on practitioners who now need to predefine the collection of dynamics. They do not use GPs as we do, and do not consider safety. Compared to Ball et al. (2020), we believe SAMBA is more flexible supporting model-learning from scratch and enabling principled exploration coupled with safety consideration. Further afield from our work, active learning has been recently studied in the context of GP time-series in Zimmer et al. (2018), and for pure exploration in Shyam et al. (2019), which uses a finite collection of models. Our functions generalise the above to consider safe-regions and future information trade-off as we detail in Sect. 3.2.

3 SAMBA: framework & solution

3.1 Solution method

In designing SAMBA, we take PILCO (Deisenroth and Rasmussen 2011) as a template and introduce two novel ingredients allowing for active exploration and safety. Following PILCO, SAMBA runs a main loop that gathers traces from the real environment and updates a surrogate model, $\mathcal{P}_{\text{GP}}(\cdot) : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \rightarrow [0, \infty)$, encoded by a Gaussian process. Given $\mathcal{P}_{\text{GP}}(\cdot)$, PILCO and other model-based methods (Srinivas et al. 2020) attempt to obtain a policy that minimises total-expect cost with respect to traces, $\boldsymbol{\tau}$, sampled from the learnt model by solving

$$\min_{\pi} \mathbb{E}_{\boldsymbol{\tau} \sim p_{\text{surr}}(\boldsymbol{\tau})}[\mathcal{C}(\boldsymbol{\tau})]$$

with $\mathcal{C}(\boldsymbol{\tau}) = \sum_{t=0}^{T} \gamma^t c(\boldsymbol{x}_t, \boldsymbol{u}_t)$ and

$$p_{\text{surr}}(\boldsymbol{\tau}) = \mu_0(\boldsymbol{x}_0) \prod_{t=0}^{T-1} \mathcal{P}_{\text{GP}}(\boldsymbol{x}_{t+1} | \boldsymbol{x}_t, \boldsymbol{u}_t) \pi(\boldsymbol{u}_t | \boldsymbol{x}_t)$$

Given a (GP) model of an environment, we formalise our problem as a constrained MDP with an additional active learning loss. We define a cost function $c : \mathcal{X}\times\mathcal{U} \rightarrow \mathbb{R}$, constraint cost function (used to encode safety) $l : \mathcal{X}\times\mathcal{U} \rightarrow \mathbb{R}$, additional objective function $\zeta : \mathcal{X}\times\mathcal{U} \rightarrow \mathbb{R}$, and discount factor $\gamma \in [0, 1)$.[1] In our setting, for instance, l encodes the state-action's risk by measuring the distance to an unsafe region, while $\boldsymbol{\zeta}$ denotes an acquisition function from these described in Sect. 3.2. To finalise the problem definition, we need to consider an approachable constraint to describe safety considerations. In incorporating such constraints, we are chiefly interested in those that are flexible (i.e., can support different user-designed safety criteria) and allow us to quantify events occurring in tails of cost distributions. When surveying constrained MDPs, we realise that the literature predominately focuses on expectation-type constraints (Achiam et al. 2017; Ray et al. 2019) – a not so flexible approach restricted to being safe on average. Others, however, make use of conditional-value-at-risk (CVaR); a coherent risk measure (Chow et al. 2017) that provides powerful and flexible notions of safety (i.e., can support expectation, tail-distribution, or hard – unsafe state visitation – constraints) and quantifies tail risk in the worst $(1 - \alpha)$-quantile. Formally, given a random variable Z, $\text{CVaR}_\alpha(Z)$ is defined as: $\text{CVaR}_\alpha(Z) = \min_\nu[\nu + \frac{1}{1-\alpha}\mathbb{E}[(Z - \nu)^+]]$, where $(Z - \nu)^+ = \max(Z - \nu, 0)$.

With such a constraint, we can write the optimisation problem as:

$$\min_\pi \mathbb{E}_{\boldsymbol{\tau}\sim p_{\text{surr}}(\boldsymbol{\tau})}[\mathcal{C}(\boldsymbol{\tau}), \zeta(\boldsymbol{\tau})]^{\mathsf{T}} \text{ s.t. } \text{CVaR}_\alpha(\mathcal{L}(\boldsymbol{\tau})) \leq \xi, \tag{1}$$

with $\mathcal{L}(\boldsymbol{\tau}) = \sum_{t=0}^{T} \gamma^t l(\boldsymbol{x}_t, \boldsymbol{u}_t)$ being total accumulated safety cost along $\boldsymbol{\tau}$, and $\xi \in \mathbb{R}_+$ a safety threshold.

We transform this problem into a single objective one through a linear combination of $\mathcal{C}(\boldsymbol{\tau})$ and $\zeta_{\text{LOO}}(\boldsymbol{\tau})$. This relaxed yet constrained version is given by

$$\min_\pi \lambda_\pi \mathbb{E}_{\boldsymbol{\tau}\sim p_{\text{surr}}(\boldsymbol{\tau})}[\mathcal{C}(\boldsymbol{\tau})] - (1 - \lambda_\pi)\mathbb{E}_{\boldsymbol{\tau}\sim p_{\text{surr}}(\boldsymbol{\tau})}[\boldsymbol{\zeta}_{\text{LOO}}(\boldsymbol{\tau})] \text{ s.t. } \text{CVaR}_\alpha(\mathcal{L}(\boldsymbol{\tau})) \leq \xi \tag{2}$$

where λ_π is a tuneable hyper-parameter in [0, 1] and $\boldsymbol{\zeta}_{\text{LOO}}$ is an acquisition function defined in Sect. 3.2. Please note that the negative sign in the linear combination is due to the fact that we used $-\boldsymbol{\zeta}_{\text{LOO}}(\boldsymbol{\tau})$.

From Constrained to Unconstrained Objectives We write an unconstrained problem using a Lagrange multiplier λ_{CVaR}:

$$\begin{aligned}\min_\pi \lambda^\star_\pi \mathbb{E}_{\boldsymbol{\tau}\sim p_{\text{surr}}(\boldsymbol{\tau})}[\mathcal{C}(\boldsymbol{\tau})] - (1 - \lambda^\star_\pi)\mathbb{E}_{\boldsymbol{\tau}\sim p_{\text{surr}}(\boldsymbol{\tau})}[\boldsymbol{\zeta}_{\text{LOO}}(\boldsymbol{\tau})] \\ + \lambda_{\text{CVaR}}[\text{CVaR}_\alpha(\mathcal{L}(\boldsymbol{\tau})) - \xi]\end{aligned} \tag{3}$$

Due to non-convexity of the problem, we cannot assume strong duality holds, so in our experiments, we schedule λ_{CVaR} proportional to gradients using a technique similar to that in Schulman et al. (2017) that has proven effective.[2] To solve the above optimisation problem, we first fix ν and perform a policy gradient step in π.[3] To minimise the variance in the gradient estimator of the accumulated cost $\mathcal{C}(\boldsymbol{\tau})$ and the acquisition function $\boldsymbol{\zeta}_{\text{LOO}}(\boldsymbol{\tau})$

[1] It is worth noting that the acquisition functions we develop in Sect. 3.2 map to $\mathbb{R}_+$ instead of $\mathbb{R}$.

[2] Note that a primal dual-method as in (Chow et al. 2015) is not applicable due to non-convexity. In the future, we plan to study approaches from (Goh and Yang 2001) to ease determining λ_{CVaR}.

[3] We resorted to policy gradients for two reason: (1) cost functions are not necessarily differentiable, and (2) better experimental behaviour when compared to model back-prop especially on OpenAI's safety gym tasks.

(defined in Sect. 3.2), we build two neural network critics that we use as baselines. The first attempts to model the value of the standard cost, while the second learns information gain values. For the CVaR's gradient, we simply apply policy gradients. As CVaR is non-Markovian, it is difficult to estimate its separate critic. In our experiments, a heuristic where discounted safety losses as unbiased baselines was used and proved effective. In short, our main update equations when using a policy parameterised by a neural network with parameters $\boldsymbol{\theta}$ can be written as:

$$\begin{aligned}\boldsymbol{\theta}^{[j][k+1]} = \boldsymbol{\theta}^{[j][k]} - \eta_k\Big(\mathbb{E}_{\boldsymbol{\tau}}\Big[\sum_{t\geq 1}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\boldsymbol{u}_t|\boldsymbol{x}_t)\Big(\lambda_\pi^\star(Q_{\mathcal{C}}(\boldsymbol{x}_t,\boldsymbol{u}_t) - V_{\mathcal{C}}^{\phi_1}(\boldsymbol{x}_t))\\ - (1-\lambda_\pi^\star)(Q_\zeta(\boldsymbol{x}_t,\boldsymbol{u}_t) - V_\zeta^{\phi_2}(\boldsymbol{x}_t))\Big)\Big] + \lambda_{\text{CVaR}}\nabla_{\boldsymbol{\theta}}\text{CVaR}_\alpha(\mathcal{L}(\boldsymbol{\tau}))\Big),\end{aligned} \tag{4}$$

where η_k is a learning rate, and $V_{\mathcal{C}}^{\phi_1}(\boldsymbol{x}_t)$, $V_\zeta^{\phi_2}(\boldsymbol{x}_t)$ are neural network critics with parameters $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$. We present the main steps in Algorithm 1 and more details in the Appendix.

Algorithm 1 SAMBA: Safe Model-Based & Active Reinforcement Learning

```
1: **Inputs:** $\lambda_\pi^\star$, $\lambda_{\text{CVaR}}$ initialisation, initial random policy $\pi_1$, $\mathcal{D}_0 = \{\}$, $\alpha$-quantile, safety
   threshold $\xi$
2: **for** $j = 1 : \#\text{env-iterations}$ **do:**
3:     Sample traces from the real environment using $\pi_j$, concatenate all data and update
       $\mathcal{P}_{\text{GP}}^{[j]}(\cdot)$
4:     **for** $k = 1 : \#\text{control-iterations}$ **do:**
5:         Sample traces from $\mathcal{P}_{\text{GP}}^{[j]}(\cdot)$ and compute $\boldsymbol{\zeta}_{\text{LOO}}^{[j][k]}(\boldsymbol{\tau})$ (§ 3.2)
6:         Update $\phi_1^{[j][k]}$ and $\phi_2^{[j][k]}$ with $\mathbb{E}_{\boldsymbol{\tau}}[\mathcal{C}(\boldsymbol{\tau})]$ and $\mathbb{E}_{\boldsymbol{\tau}}[\boldsymbol{\zeta}_{\text{LOO}}(\boldsymbol{\tau})]$ as targets, and $\boldsymbol{\theta}^{[j][k]}$
           (§ 3.1)
7:     Set $\pi_{j+1} = \pi_{\#\text{control-iterations}}$
8: **Output:** Policy $\pi_{\#\text{env-iterations}}$
```

The updated policy is then used to sample new traces from the real system where the above process repeats. During this sampling process, model-based algorithms consider various acquisition functions in acquiring transitions that reveal novel information, which can be used to improve the surrogate model's performance. PILCO, for instance, makes use of the GP uncertainty, while ensemble models (Saphal et al. 2020; van Amersfoort et al. 2020) explore by their aggregated uncertainties. With sufficient exploration, this allows policies obtained from surrogate-models to control real-systems. Our safety considerations mean we would prefer agents that learn well-behaving policies with minimal sampling from unsafe regions of state-action spaces; a property we achieve later by incorporating CVaR constraints as we detail in Sect. 3.1. Requiring a reduced number of visits to unsafe regions, hence, lessens the amount of "unsafe" data gathered in such areas by definition. Therefore, model entropy is naturally increased in these territories and algorithms following such exploration strategies are, as a result, encouraged to sample from hazardous states. As such, a naive adaptation of entropy-based exploration can quickly become problematic by contradicting safety requirements. To circumvent these problems, we introduce two new acquisition functions in Sect. 3.2, that assess information beyond training-data availability and consider input-output data distributions. Our acquisition functions operate under the assumption that during any model update step, "safe" transition data (i.e., a set of state-action-successor states sampled from safe regions) is more abundant in number when compared to "unsafe" triplets. Considering such a skew between distributions, our acquisition functions yield increased values on test queries close to high-density training data. Given such an acquisition function, we enable novel model-based algorithms that attempts to minimise cost, while maximising active values and satisfying safety constraints.

Of course, the assumption of having skew towards safe regions in training data distribution is generally not true since solving the above only ensures good expected returns.

To frequently sample safe regions, we augment cost minimisation with a safety constraint that is encoded through the CVaR of a user-defined safety cost function with respect to *model* traces. Hence, SAMBA solves a constrained optimisation problem (Sect. 3.1) aimed at minimising cost, maximising active exploration, and meeting safety constraints.

3.2 ζ-functions for safe active exploration

In general, ζ can be any bounded objective that needs to be maximised/minimised in addition to standard cost. Here, we choose one that enables active exploration in safe state-action regions. To construct ζ, we note that a feasible policy – i.e., one abiding by CVaR constraints – of the problem in Eq. (1) samples tuples that mostly reside in safe regions. As such, the training data distribution is skewed in the sense that safe state-action pairs are more abundant than unsafe ones. Exploiting such skewness, we can indirectly encourage agents to sample safe transitions by maximising information gain functions that only grow in areas close-enough to training data.

ζ_{LOO}*: Leave-One-Out acquisition* Consider a GP dynamics model, $\mathcal{P}_{\text{GP}}$, that is trained on a state-action-successor-state data set $\mathcal{D} = \{\langle \tilde{\boldsymbol{x}}^{(i)}, y^{(i)}\rangle\}_{i=1}^{n_1}$ with $\tilde{\boldsymbol{x}}^{(i)} = (\boldsymbol{x}^{(i)}, \boldsymbol{u}^{(i)})$.[4] Such a GP induces a posterior allowing us to query predictions on n_2 test points $\tilde{\boldsymbol{x}}_\star = \{(\boldsymbol{x}_\star^{(j)}, \boldsymbol{u}_\star^{(j)})\}_{j=1}^{n_2}$. As noted in Sect. 2.1, the posterior is also Gaussian with the following mean vector and covariance matrix:[5] $\boldsymbol{\mu}_\star = \boldsymbol{K}_{n_2,n_1}\boldsymbol{A}_{n_1,n_1}\boldsymbol{y}_{n_1}$ and $\boldsymbol{\Sigma}_{n_2,n_2} = \boldsymbol{K}_{n_2,n_2} - \boldsymbol{K}_{n_2,n_1}\boldsymbol{A}_{n_1,n_1}\boldsymbol{K}_{n_1,n_2}$. Our goal is to design a measure that increases in regions with dense training-data (due to the usage of CVaR constraint) to aid agents in exploring novel yet safe tuples. To that end, we propose using an expected leave-one-out acquisition function between two Gaussian processes defined, for a one query data point $\tilde{\boldsymbol{x}}_\star$, as:

$$\zeta_{\text{LOO}}(\tilde{\boldsymbol{x}}_\star) = \mathbb{E}_{i\sim\text{Uniform}[1,n_1]}\left[\text{KL}\left(p(\boldsymbol{f}_\star|\mathcal{D}_{\neg i})||p\left(\boldsymbol{f}_\star|\mathcal{D}\right)\right)\right]$$

with $\mathcal{D}_{\neg i}$ being $\mathcal{D}$ with point i left-out. Importantly, such a measure will only grow in regions which are close-enough to sampled training data, as posterior mean and covariance of $p(\boldsymbol{f}_\star|\mathcal{D}_{\neg i})$ shift by a factor that scales linearly and quadratically, respectively, with the total covariance between $\tilde{\boldsymbol{x}}_\star$ and $\boldsymbol{X}_{\neg i}$ where $\boldsymbol{X}_{\neg i}$ denotes a feature matrix with the i^{th} row removed.[6] In other words, such a acquisition function fulfils our requirement in the sense that if a test query is distant (in distribution) from all training input data, it will achieve low ζ_{LOO} score. Though appealing, computing a full-set of ζ_{LOO} can be highly computationally expensive, of the order of $\mathcal{O}(n_1^4)$ – computing $\boldsymbol{A}_{n_{1\neg i},n_{1\neg i}}$ requires $\mathcal{O}(n_1^3)$ and this has to be repeated n_1 times. A major source contributing to this expense, well-known in GP literature, is related to the need to invert covariance matrices. Rather than following variational

[4] As $\mathcal{D}$ changes at every outer iteration, we simply concatenate all data in one larger data set; see Algorithm 1.

[5] For clarity, we describe a one-dimensional scenario.

[6] Though intuitive, we provide a formal treatment of the reason behind such growth properties in the Appendix.

approximations (which constitute an interesting direction), we prioritise sample-efficiency and focus on exact GPs. To this end, we exploit the already computed $\boldsymbol{A}_{n_1,n_1}$ during the model-learning step and make-use of the matrix inversion lemma (Petersen et al. 2008) to recursively update the mean and covariances of $p(\boldsymbol{f}_\star|\mathcal{D}_{\neg i})$ for all i (see Appendix): $\boldsymbol{\mu}_\star^{(i)} = \boldsymbol{\mu}_\star - \boldsymbol{K}_{n_2,n_1}\frac{\boldsymbol{a}_i^\mathsf{T}\boldsymbol{a}_i}{a_{i,i}}\boldsymbol{y}_{n_1}$ and $\boldsymbol{\Sigma}_{n_2,n_2}^{(i)} = \boldsymbol{\Sigma}_{n_2,n_2} + \boldsymbol{K}_{n_2,n_1}\frac{\boldsymbol{a}_i^\mathsf{T}\boldsymbol{a}_i}{a_{i,i}}\boldsymbol{K}_{n_1,n_2}$, with $\boldsymbol{a}_i$ being the i^{th} row of $\boldsymbol{A}_{n_1,n_1}$. Hence, updating the inverse covariance matrix only requires computing and adding the outer product of the i^{th} row $\boldsymbol{A}_{n_1,n_1}$, divided by the i^{th} diagonal element. This, in turn, reduces complexity from $\mathcal{O}(n_1^4)$ to $\mathcal{O}(n_1^3)$. Note that now the main contributors to the computational cost of LOO are repeated matrix-vector, vector-vector multiplications, which can be efficiently distributed. Moreover, $\mathcal{O}(n_1^3)$ is also the complexity of the exact GP we rely on for the surrogate model, with the notable difference that the matrix inversions that are required for inference with a GP cannot be distributed as efficiently as the operations detailed above. As a consequence, one would not expect the computational budget to increase dramatically with the use of this acquisition function.

$\zeta_{\text{Bootstrap}}$*: Bootstrapped symmetric acquisition* We also experimented with another acquisition function that quantifies posterior sensitivity to bi-partitions of the data as measured by symmetric KL-divergence[7] (also called Jensen-Shannon divergence), averaged over possible bi-partitions: $\zeta_{\text{Bootstrap}}(\tilde{\boldsymbol{x}}_\star) = \mathbb{E}_{\langle\mathcal{D}_1,\mathcal{D}_2\rangle}[\text{KL}_{\text{sym}}(p(\boldsymbol{f}_\star|\mathcal{D}_1)||p(\boldsymbol{f}_\star|\mathcal{D}_2))]$, where $\langle\mathcal{D}_1,\mathcal{D}_2\rangle$ is a random bi-partition of the data $\mathcal{D}$. In practice, we randomly split the data in half, and do this K times (where K is a tuneable hyper-parameter) to get a collection of K bi-partitions. We then average over that collection. Similar to ζ_{LOO}, $\zeta_{\text{Bootstrap}}$ also assigns low importance to query points far from the training inputs, and hence, can be useful for safe-decision making. In our experiments, ζ_{LOO} provided better-behaving exploration strategy, see Sect. 4.

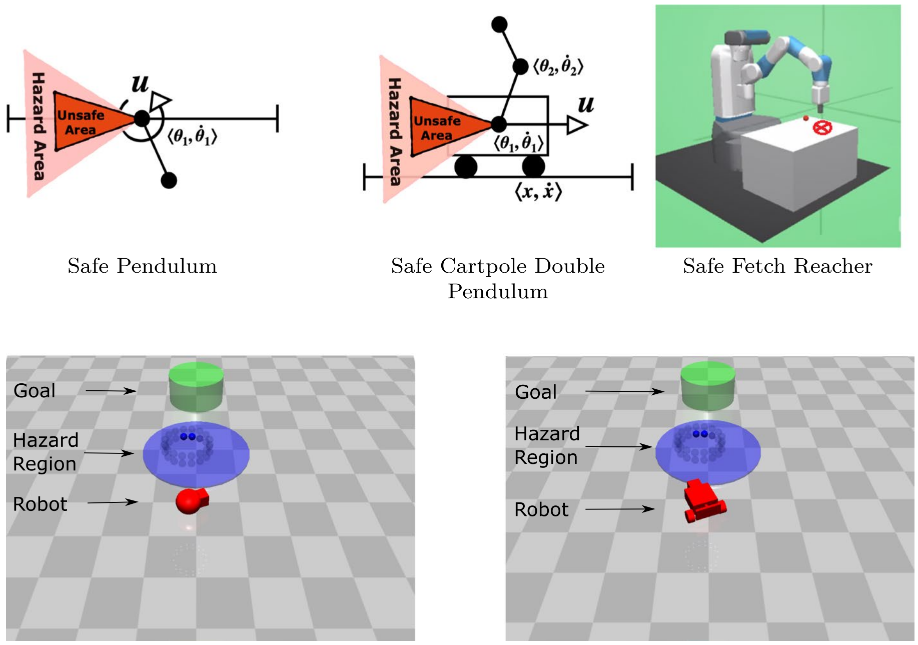
Transforming $\zeta_\cdot(\tilde{\boldsymbol{x}}_\star)$ *to* $\zeta_\cdot(\boldsymbol{\tau})$ Both introduced functions are defined in terms of query test points $\tilde{\boldsymbol{x}}_\star$. To incorporate in Eq. (1), we define trajectory-based expected total information gain as

$$\boldsymbol{\zeta}_{\text{LOO}}(\boldsymbol{\tau}) = \sum_{t=0}^{T}\gamma^t\zeta_{\text{LOO}}(\langle\boldsymbol{x}_t,\boldsymbol{u}_t\rangle),$$

$$\boldsymbol{\zeta}_{\text{Bootstrap}}(\boldsymbol{\tau}) = \sum_{t=0}^{T}\gamma^t\zeta_{\text{Bootstrap}}(\langle\boldsymbol{x}_t,\boldsymbol{u}_t\rangle).$$

Interestingly, this characterisation trades off long-term versus short-term information gain similar to how cost trades-off optimal greedy actions versus long-term decisions. In other words, it is not necessarily optimal to seek an action that maximises immediate information gain since such a transition can ultimately drive the agent to unsafe states (i.e., ones that exhibit low $\zeta_\cdot(\tilde{\boldsymbol{x}})$ values). In fact, such horizon-based definitions have also recently been shown to improve modelling of dynamical systems (Buisson-Fenet et al. 2019; Shyam et al. 2019). Of course, our problem is different in the sense that we seek safe policies in a safe decision-making framework, and thus require safely exploring acquisition functions.

[7] A symmetric KL-divergence between two distributions p, and q is defined as: $(\text{KL}(p\|q)+\text{KL}(q\|p))/2$.


Hazard Area
Unsafe Area
u
$\langle\theta_1,\dot{\theta}_1\rangle$
$\langle\theta_2,\dot{\theta}_2\rangle$
$\langle x,\dot{x}\rangle$
Goal
Hazard Region
Robot

Safe Pendulum

Safe Cartpole Double Pendulum

Safe Fetch Reacher

Safety point robot

Safety Car robot

Fig. 1 Environments with their respective unsafe regions visualised for evaluating safe algorithms

4 Experiments

We assess SAMBA in terms of both *safe learning* (train) and *safe final policies* (test) on five dynamical systems, two of which are adaptations of standard dynamical systems for MBRL (Safe Pendulum and Safe Cart-Pole Double Pendulum), while the third (Fetch Robot — optimally control end-effector to reach a 3D goal) we adapt from OpenAI's robotics environments (Brockman et al. 2016). Lastly, we use the car and point safe robotic task from OpenAI Safety Gym (Ray et al. 2019). All environments and respective unsafe (hazard) regions are visualised in Fig. 1. Total cost (TC) during training is the constraint cost the agent acquires throughout all training interactions with the environment, and similarly total cost during evaluation is the accumulated constraint cost at test time in the environment. Total violation (TV) similarly is the total constraint violations through training/testing. Note, constraints in general are not explicitly designed to minimise TV, and TV can be non-zero even in the optimality.

In each of the benchmark tasks, we define unsafe regions as areas in state space and design the safety loss (i.e., $\mathcal{L}(\boldsymbol{\tau})$) to correspond to the (linearly proportional) distance between the end-effector's position (when in the hazard region) to the centre of the unsafe region. SAMBA implemented a more stable proximal update of Eq. (4) following a similar method to (Schulman et al. 2017). We compare against algorithms from both model-free and model-based literature. Unconstrained model-free comparisons against TRPO (Schulman et al. 2015), PPO (Schulman et al. 2017), while expectation constrained model-free methods include CPO (Achiam et al. 2017), STRPO (safety-constrained TRPO) (Ray et al. 2019) and SPPO (safety-constrained PPO) (Ray et al. 2019). These expectation constrained

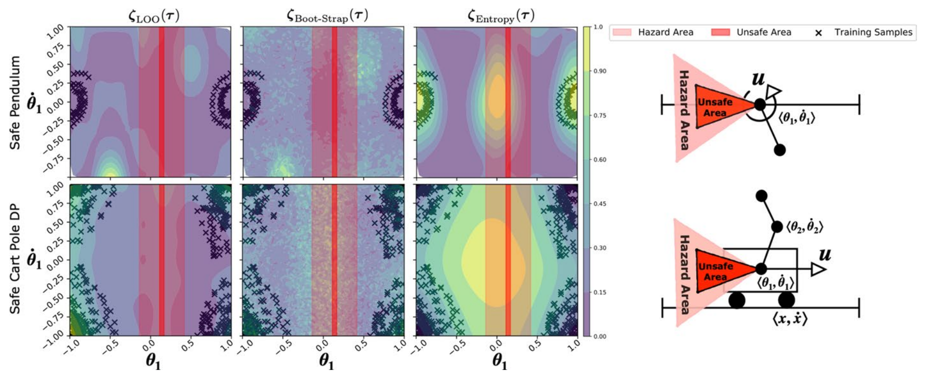


Fig. 2 Comparing acquisition function values across the entire state space. Note, policies used within this framework are encouraged to visit regions with higher (yellow) values. We first generate samples by random rollouts, as is typical in initial learning stages for most deep reinforcement learning algorithms. A GP dynamics model for each environment was then trained on 100 samples. The samples used to train the models are marked with black crosses. We then plot the varying acquisition function values using this GP for varying acquisition functions. More plots for different numbers of samples can be found in the Appendix (Color figure online)

and unconstrained algorithms enable us to determine if SAMBA improves upon the following: sample complexity during training (number of observations during training from the real system); total violations (TV), that is, the total number of timesteps spent inside the unsafe region; total accumulated safety cost.[8] Comparison with unconstrained model-based solvers (e.g., PlaNet (Hafner et al. 2019), PILCO (Deisenroth and Rasmussen 2011), MBPO (Janner et al. 2019)) sheds light on the importance of our acquisition functions for safe exploration. We use PlaNet and MBPO as generic model-based deep reinforcement learning algorithm that use neural network dynamic models, i.e., we do not anticipate a considerably higher sample efficiency from other methods with neural network dynamic models. It is important to note that when implementing PILCO, we preferred a flexible solution that does not assume moment-matching and specific radial-basis function controllers. Hence, we adapted PILCO to support proximal policy updates, referred to as PPILCO in our experiments. In our comparisons, we will also employ a variant of SAMBA, which does not use active learning (whilst the CVaR constraint is kept) and call it SAMBA w/o active learning. This algorithm can be seen as a safety-constrained adaptation of PPILCO (i.e., PPILCO with CVaR constraint). The adaptation of PPILCO with active learning (i.e., SAMBA without CVaR constraints) performed very similarly to PPILCO and therefore not presented. Note, we do not show time complexity analysis as we did not observe significant differences between algorithms overall run time, whilst the focus of this work remains on lowering the total unsafe interactions required by a safe algorithm at train and deployment (test). As SAMBA introduces exploration components to standard model-based learners, we analysed these independently before combining them and reporting TV and TC (see

[8] Note, we report safe learning process TC, which is the total incurred safety cost throughout all training environment interactions, and safe evaluation TC and TV, which similarly is the total incurred safety cost during evaluation, and the total violations from the timesteps spent inside the unsafe region during evaluation. Another metric used to compare safety algorithms is samples used for training, this is often referred to purely as samples, and is defined as the numbers of observations of the real system.

Table 1 and Table 3).[8] All policies are represented by two-hidden-layer (32 units each) neural networks with *tanh* non-linearities, which is one of the common choices for policy architectures in the RL literature. All hyper-parameters to reproduce our results can be found in the Appendix, we typically fixed all policy and critic parameters. The most crucial hyper-parameter to learning a safe policy was the cost constraint, whereby a simple random search over the range [0.01, 0.05] led to safe policies on all environments. We also invested resources in tuning the gamma used in PPO, searching over the values $\gamma \in [0.99, 0.98, 0.97, 0.96, 0.95]$, although this was not detrimental to algorithms safety performance. We set $\lambda^{\star}_{\pi} = 0.9$, which provided reasonable results for all experiments.

Acquisition Function Component Evaluating our (acquisition) functions, we conducted an analysis that reports a 2D projection view of the state space at various intervals in the data-collection process. We compare $\zeta_{\text{LOO}}(\boldsymbol{\tau})$ and $\zeta_{\text{Bootstrap}}(\boldsymbol{\tau})$ against a *Max Entropy* (Shannon 2001) acquisition function and report the results on two systems in Fig. 2. Our results suggest that LOO is much more conservative and will aim to guide the policy away from the dangerous area. At the same time, entropy encourages exploration into the unsafe area, additionally encouraged with the increase of number of samples - a highly undesirable property when safe active exploration is required. Similar results are demonstrated with the Fetch Reach robot task (in the Appendix). It is also worth noting that due to the high-dimensional nature of the tasks, visual analysis can only give indications. Still, empirical analysis supports our claim and performance improvements are clear; see Table 1.

Learning and Evaluation Having seen initial results which highlight the benefit of using our acquisition functions, we then conducted learning and evaluation experiments comparing SAMBA against state-of-the-art methods.

Results reported in Table 1 demonstrate that SAMBA reduces the amount of training TC, and samples [8] compared to others. This is, perhaps, expected while comparing SAMBA to model-free baselines (SPPO, CPO, STRPO), which are known to be sample inefficient, and PlaNet w RS, which was designed for high-dimensional environments. Interestingly, during safe evaluation (deploying learnt policy and evaluating performance), we see SAMBA's safety performance competitive with (if not significantly better than) policies trained for safety in terms of TC and TV. [8] Such results are interesting as SAMBA was never explicitly designed to minimise test TV, [8] but it was still able to acquire significant reductions. Of course, one might argue that these results do not convey a safe final policy, as violations are still non-zero. Recall, however, that we define safety in terms of CVaR constraints, which do not totally prohibit violations, rather, limit the average cost of excess violations beyond a (user-defined) risk level α. Indeed, as mentioned above, it is not possible to guarantee total safety without strong assumptions on dynamics and/or an initial safe policy (and of course, none of the algorithms in Table 1 achieve zero violations). Finally, note that SAMBA policies consistently deliver similar safety costs (TC and TV) as SPPO and STRPO, which can be explained by similar policy updates for all three methods.

Learning Reducing Constraint Cost & Sample Complexity. Figure 3 shows that SAMBA significantly reduces the total acquired constraint cost (top), and number of samples used to train an agent (bottom). This illustrates that sample efficiency does not come as a trade-off to safety (the values of TC and TV) at the test or the evaluation time. We can attribute this behaviour to use of a Gaussian Process being able to accurately model the transition

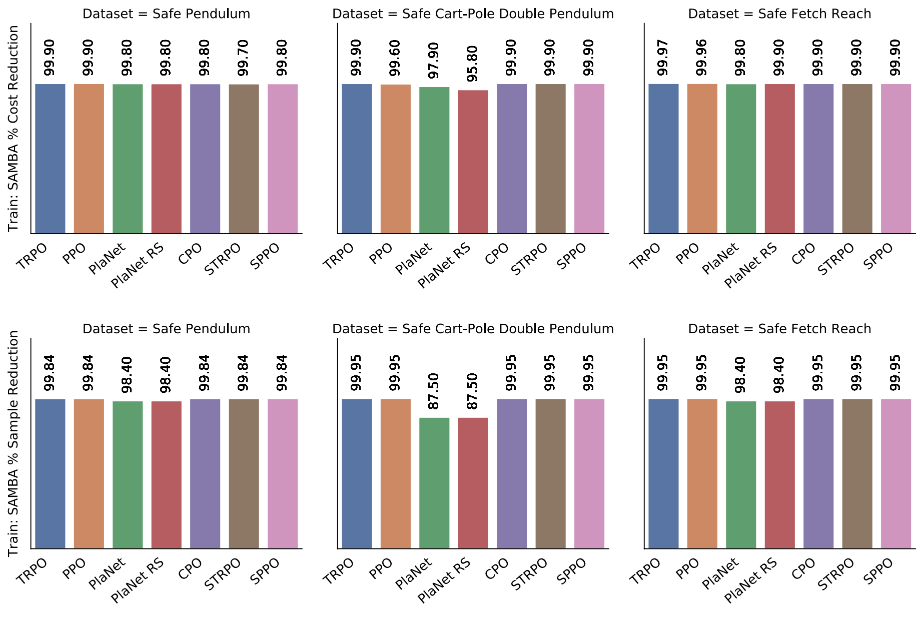


Fig. 3 (Top) Percentage reduction of total constraint cost [8] acquired to train SAMBA compared to baselines. (Bottom) Percentage reduction of samples [8] used to train SAMBA compared to baselines

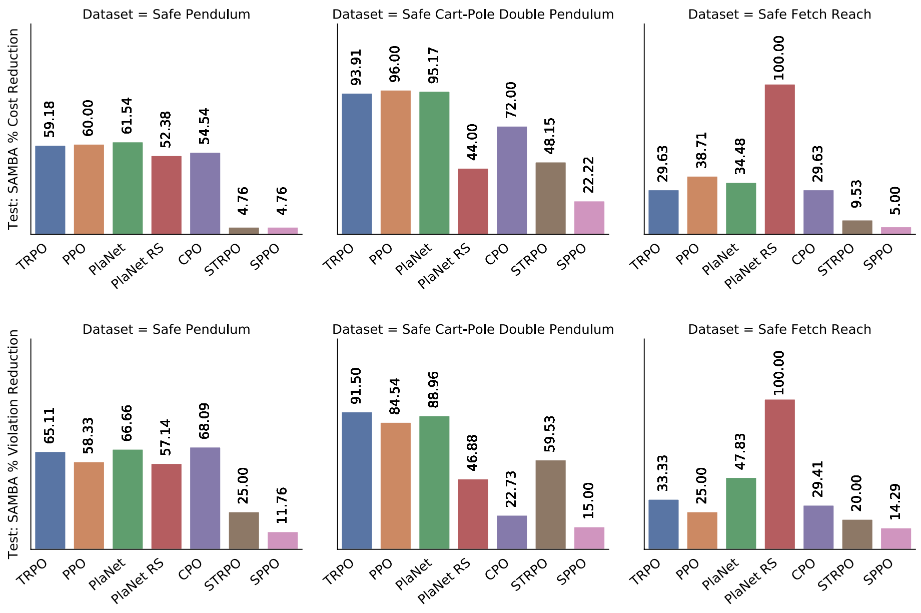


Fig. 4 (Top) Percentage reduction of total constraint cost acquired during evaluation of SAMBA compared to baselines. (Bottom) Percentage reduction of constraint violations during evaluation of SAMBA compared to baselines

Table 1 Safe learning and safe evaluation for Safe Pendulum, Safe Cart Pole Double Pendulum, and Safe Fetch Reach. We averaged each policy's safe evaluation over five random seeds, and collected 10000 evaluation samples per seed[8]

Learner	Safe pendulum				Safe cart pole double pendulum				Safe fetch reach			
	Learning		Evaluation		Learning		Evaluation		Learning		Evaluation	
	Samples	TC	TV	TC	Samples	TC	TV	TC	Samples	TC	TV	TC
TRPO	1000	109	4.3	4.9	1000	311	10	23	1000	793	1.8	2.7
PPO	1000	81	3.6	5	1000	144	5.5	35	1000	667	1.6	3.1
PlaNet	100	58	4.5	5.2	40	2.7	7.7	29	100	114	2.3	2.9
MBPO	90	50	4.3	5.6	50	3.3	6.9	31	80	102	2.7	3.1
PPILCO	2	0.04	2.8	4.5	6	0.097	7.1	33	5	0.49	1.9	3
PlaNet w RS	100	52	3.5	4.2	40	1.4	1.6	2.5	–	–	–	–
CPO	1000	61	4.7	4.4	1000	98	1.1	5	1000	169	1.9	2.7
STRPO	1000	33	2.0	2.1	1000	86	2.1	2.7	1000	169	1.7	2.1
SPPO	1000	67	1.7	2.4	1000	65	1	1.8	1000	296	1.5	2
SAMBA (w/o active learning)	1.8	0.02	1.7	2.0	6	0.062	1.2	1.7	5	0.38	1.4	2.2
SAMBA	**1.6**	**0.01**	**1.5**	**2.0**	**5**	**0.052**	**0.85**	**1.4**	**5**	**0.21**	**1.2**	**1.9**

The data is scaled by 10^3. PlaNet w RS on Safe Fetch Reach diverged during training. We have stopped training the model-free approaches (PPO, TRPO, CPO, SPPO, STRPO) after collecting $1000 \cdot 10^3$ samples.

function in few samples, allowing us to transfer the dangerous learning process from the real environment into the dynamics model.

Evaluation Reducing Constraint Cost & Violations Figure 4 (top) shows large percentage improvements of SAMBA's constraint costs during evaluation in the real environment, compared with baselines. From our ablation studies (Table 1), we can clearly see these gains are due to the addition of both the CVaR metric and the active learning component combined. Figure 4 (bottom) tells the same story as evaluation constraint cost, although the raw numbers for violations are significantly lower than the constraint costs acquired, to the violation region being much smaller than the constraint cost inducing region.

To evaluate risk performance, we conducted an in-depth study evaluating both expectation and CVaR constraints on the two most challenging environments, Safe Cart Pole Double Pendulum and Safe Fetch Reach. We would expect all constrained methods to satisfy their constraint at test time, which surprisingly is not what we observe for CPO, seen in Table 2. Secondly, we observe that CVaR constrained methods are able to meet their constraints, as we as the expectation constraints. Thirdly, it is clear that active learning can play a crucial role in producing a constraint satisfying policy, as is seen in the ablation study in Safe Fetch Reach. Overall, Table 2 demonstrates that SAMBA achieves lower safety cost quartiles and lower expected safety cost, using cost limits $\xi = 0.25$ and $\xi = 0.75$ (Table 3). Similarly, Table 4 demonstrates that SAMBA significantly outperforms others with respect to safety during evaluation, where we used a cost limit $\xi = 5$ and $\xi = 10$ for

Table 2 Constraint satisfaction (✓) or constraint violation (×) on Safe Cart Pole Double Pendulum and Safe Fetch of model-based (bold) and model-free (italics) solvers. We evaluated expectation constraints (Exp.) and CVaR over 5 seeds per algorithm. Results show SAMBA satisfied safety constraints, outperforming others in different quartiles. There is also a clear trade-off between returns and safety for all algorithms. Interestingly, SAMBA is safer yet acquiring acceptable expected returns – close to SPPO for instance. Note, all policies shown below solve the given tasks, i.e., reach the target states

Learner	Safe cart pole double pendulum						Safe fetch reach					
	Quartile			Constraint		$\mathbb{E}_\tau[\mathcal{C}(\tau)]$	Quartile			Constraint		$\mathbb{E}_\tau[\mathcal{C}(\tau)]$
	0.25	0.5	0.75	Exp.	CVaR$_\alpha$		0.25	0.5	0.75	Exp.	CVaR$_\alpha$	
PPO	5.1	5.5	9.6	×	×	**– 19**	0.94	1.26	1.88	×	×	**-0.26**
PPILCO	5.2	5.4	8.7	×	×	– 21	0.91	1.22	1.45	×	×	– 0.36
PlaNet	5.2	6.1	8.7	×	×	– 21	0.62	1.13	1.87	×	×	– 0.39
MBPO	4.5	5.5	9.9	×	×	– 21	0.68	1.57	2.09	×	×	– 0.37
TRPO	5.0	5.7	7.4	×	×	– 22	0.68	0.83	0.94	×	×	– 0.57
CPO	0.44	0.49	0.94	×	×	– 23	0.86	0.90	1.01	×	×	– 0.43
PlaNet w RS	0.82	0.99	1.07	×	×	-24	–	–	–	–	–	–
SAMBA (w/o active learning)	0.20	0.27	0.31	×	×	– 25	0.44	0.75	1.11	✓	×	– 0.57
STRPO	0.23	0.26	0.33	✓	×	– 28	0.13	0.47	0.87	✓	×	– 0.98
SPPO	0.20	0.23	0.32	✓	×	– 27	0.29	0.43	0.72	✓	✓	– 1.51
SAMBA	0.14	0.20	0.23	✓	✓	– 27	**0.00**	**0.05**	**0.18**	✓	✓	– 2.27

PlaNet w RS on Safe Fetch Reach diverged during training.

Table 3 Safe learning and safe evaluation for Safety Gym Car and Safety Gym Point. We averaged each policy's safe evaluation over five random seeds, and collected 10000 evaluation samples per seed. [8]. Note, we tested SAMBA against all other algorithms with a two sided T-test and highlighted a statistic in bold if SAMBA outperformed *all* others significantly. Where significance is defined by all tests having a p-value of ≤ 0.05

Learner	Safety gym car				Safety gym point			
	Learning		Evaluation		Learning		Evaluation	
	Samples	TC	TV	TC	Samples	TC	TV	TC
TRPO	1000	202	3.7	5.6	1000	26.8	8.6	13
PPO	1000	205	7.4	11	1000	9	6.1	9.2
PlaNet	100	3.5	3.2	4.9	100	2.6	7.7	16
MBPO	90	3.4	4.3	5.6	100	3.2	6.9	21
PPILCO	2	0.47	7.4	11	8	0.31	5.4	8.9
PlaNet w RS	100	3.4	3.1	3.9	100	2.5	7.9	13
CPO	1000	49	4.7	4.4	1000	14	1.1	5
STRPO	1000	64	2.0	2.1	1000	17	2.1	2.7
SPPO	1000	66	1.7	2.4	1000	10	1	1.8
SAMBA (w/o active learning)	0.8	0.13	1.4	4.8	0.7	0.56	1.8	2.5
SAMBA	0.6	**0.01**	1.3	**2.2**	0.5	**0.04**	**1.3**	**1.5**

The data is scaled by 10^3. We have stopped training the model-free approaches (PPO, TRPO, CPO, SPPO, STRPO) after collecting $1000 \cdot 10^3$ samples.

Table 4 Constraint satisfaction (✓) or constraint violation (×) on Safety Gym Point and Car tasks (Ray et al. 2019). and Safe Fetch of model-based (bold) and model-free (italic) solvers. We evaluated expectation constraints (Exp.) and CVaR. Note, all policies shown below solve the given tasks, i.e., reach the target states. Note, we tested SAMBA against all other algorithms with a two sided T-test and highlighted a statistic in bold if SAMBA outperformed *all* others significantly. Where significance is defined by all tests having a p-value of ≤ 0.05

Learner	Safety gym car						Safety gym point					
	Quartile			Constraint		$\mathbb{E}_\tau[\mathcal{C}(\tau)]$	Quartile			Constraint		$\mathbb{E}_\tau[\mathcal{C}(\tau)]$
	0.25	0.5	0.75	Exp.	CVaR$_\alpha$		0.25	0.5	0.75	Exp.	CVaR$_\alpha$	
PPO	7.1	7.8	8.0	×	×	2.4	0.79	1.15	18.3	×	×	0.24
PPILCO	7.5	7.7	8.1	×	×	2.1	0.83	1.01	14.9	×	×	0.13
PlaNet	2.7	6.0	6.5	×	×	1.03	9.4	11.6	22.9	×	×	0.44
TRPO	3.5	6.7	7.3	×	×	2.14	0.02	12	23.3	×	×	0.43
CPO	2.8	4.3	4.8	✓	×	2.8	0.0	2.3	13.5	✓	×	0.12
PlaNet w RS	1.3	5.2	6.7	×	×	0.9	7.5	13.6	20.1	×	×	0.4
SAMBA (w/o active learning)	1.9	3.4	4.0	✓	✓	1.1	1.8	2.2	5.4	✓	×	0.6
STRPO	2.8	3.0	6.3	✓	×	1.9	0.17	1.16	5.20	✓	✓	0.13
SPPO	0.8	7.8	8.1	×	×	1.1	0	1.02	4.49	✓	✓	0.51
MBPO	1.7	7.4	8.2	×	×	1.15	8.5	12.1	15.2	×	×	0.39
SAMBA	0.7	**2.7**	**3.5**	✓	✓	0.92	0.00	**0.62**	**3.87**	✓	✓	0.58

Safety Gym Car and Safety Gym Point environments, respectively. These results suggest that SAMBA produces safe-final policies in terms of its objective in Eq. (1).

5 Conclusion and future work

We proposed SAMBA, a safe and active model-based learner that makes use of GP models and solves an objective constraint problem to trade-off cost minimisation, exploration, and safety. We evaluated our method on three benchmarks, including ones from Open AI's safety gym and demonstrated significant reduction in training cost and sample complexity, as well as safe-final policies in terms of CVaR constraints compared to the state-of-the-art.

One of the strengths of our approach is the active learning component, which explores by sampling around the safe areas. This, however, can lead to overly cautious sampling impeding the task learning process. This may potentially be very restrictive in complex environments, such as, a humanoid robot learning to stand, walk and eventually run. In this case, a bold exploration outside the seen data may be needed. While overcoming this limitation is outside the scope of this work, we plan to address it in future.

In future, we plan to generalise to variational GPs (Damianou et al. 2011) in order to scale to larger number systems, and to apply our method in real-world robotics. Additionally, we want to study the theoretical guarantees of SAMBA to demonstrate convergence to well-behaving, exploratory, and safe policies.

A Algorithm

In this section we provide a more detailed description of Algorithm 1. We shall use the following short-hand for the advantage function: $A_{\mathcal{C}}^{\phi_1}(\boldsymbol{x},\boldsymbol{u}) := Q_{\mathcal{C}}(\boldsymbol{x},\boldsymbol{u}) - V_{\mathcal{C}}^{\phi_1}(\boldsymbol{x})$, $A_{\boldsymbol{\zeta}}^{\phi_2}(\boldsymbol{x},\boldsymbol{u}) := Q_{\boldsymbol{\zeta}}(\boldsymbol{x},\boldsymbol{u}) - V_{\boldsymbol{\zeta}}^{\phi_2}(\boldsymbol{x})$. Note that in our implementation we use $\lambda_\pi^\star$ as a hyper-parameter, but we have also experimented with its automatic determination as presented in Algorithm 3. In this case, $\lambda_\pi^\star$ is determined using the estimates of $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{\tau}}[\mathcal{C}(\boldsymbol{\tau})]$ and $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{\tau}}[\boldsymbol{\zeta}_{\text{LOO}}]$ and scheduled at every iteration. However, since the constant value showed a good performance, we opted to use $\lambda_\pi^\star$ as a hyper-parameter.

Algorithm 2 SAMBA: Safe Model-Based & Active Reinforcement Learning

```
1: Inputs: Risk-level $\alpha$, safety threshold $\xi$,
2: Hyper-parameters: Coefficient $\lambda_\pi^\star$, Data batch size $B$, model traces batch size $N$, #env-iterations $J$, #control-iterations $K$
3: Initialisation: GP hyper-parameters (§ 2.1), $\mathcal{D} = \emptyset$, $\lambda_{\text{CVaR}} \geq 0$ arbitrarily, policy parameter $\boldsymbol{\theta}^{[0][0]}$ and critic parameters $\phi_1^{[0][0]}$, $\phi_2^{[0][0]}$ randomly
4: for $j = 0 : J-1$ do:
5:     Sample batch of $B$ new traces $\mathcal{N}_j = \{\boldsymbol{\tau}_\ell\}_{\ell=1}^{\ell=B}$ from the real environment using $\boldsymbol{\theta}^{[j][0]}$,
6:     Transform the traces $\mathcal{N}_j$ and append to data set $\mathcal{D}$:
7:         for each $\boldsymbol{\tau}_\ell \in \mathcal{N}_j$ do
8:             for each $(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1}) \in \boldsymbol{\tau}_\ell$ do
9:                 $\mathcal{D} = \mathcal{D} \cup \{\langle \tilde{\boldsymbol{x}}^{(\ell,t)} = (\boldsymbol{x}_t, \boldsymbol{u}_t), \boldsymbol{y}^{(\ell,t)} = \boldsymbol{x}_{t+1}\rangle\}$ (§ 3.2)
10:    Update $\mathcal{P}_{\text{GP}}^{[j]}(\cdot)$ using $\mathcal{D}$ and GP equations (§ 2.1)
11:    for $k = 0 : K-1$ do:
12:        Sample $N$ traces $\mathcal{S} = \{\boldsymbol{\tau}_\kappa\}_{\kappa=1}^{N}$ from $\mathcal{P}_{\text{GP}}^{[j]}(\cdot)$ using $\boldsymbol{\theta}^{[j][k]}$
13:        Compute estimates of $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{\tau}}[\mathcal{C}(\boldsymbol{\tau})]$ and $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{\tau}}[\boldsymbol{\zeta}_{\text{LOO}}(\boldsymbol{\tau})]$ using $\mathcal{S}$:
14:            $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{\tau}}[\mathcal{C}(\boldsymbol{\tau})] \approx \frac{1}{N}\sum_{\boldsymbol{\tau}_\kappa \in \mathcal{S}}\sum_{t=0}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\boldsymbol{u}_t^{(\kappa)}|\boldsymbol{x}_t^{(\kappa)})A_{\mathcal{C}}^{\phi_1^{[j][k]}}(\boldsymbol{x}_t^{(\kappa)}, \boldsymbol{u}_t^{(\kappa)})$
15:            $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{\tau}}[\boldsymbol{\zeta}_{\text{LOO}}(\boldsymbol{\tau})] \approx \frac{1}{N}\sum_{\boldsymbol{\tau}_\kappa \in \mathcal{S}}\sum_{t=0}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\boldsymbol{u}_t^{(\kappa)}|\boldsymbol{x}_t^{(\kappa)})A_{\boldsymbol{\zeta}}^{\phi_2^{[j][k]}}(\boldsymbol{x}_t^{(\kappa)}, \boldsymbol{u}_t^{(\kappa)})$
16:        Update $\phi_1^{[j][k]}$
               $\phi_1^{[j][k+1]} = \phi_1^{[j][k]} - \frac{\eta_k^{\phi_1}}{NT}\sum_{\boldsymbol{\tau}_\kappa \in \mathcal{S}}\sum_{t=0}^{T}\left(V_{\mathcal{C}}^{\phi_1^{[j][k]}}(\boldsymbol{x}_t^{(\kappa)}) - \sum_{t'=t}^{T}\gamma^{t'-t}c(\boldsymbol{x}_{t'}^{(\kappa)}, \boldsymbol{u}_{t'}^{(\kappa)})\right)\nabla_{\phi_1}V_{\mathcal{C}}^{\phi_1^{[j][k]}}(\boldsymbol{x}_t^{(\kappa)})$
17:        Update $\phi_2^{[j][k]}$
               $\phi_2^{[j][k+1]} = \phi_2^{[j][k]} - \frac{\eta_k^{\phi_2}}{NT}\sum_{\boldsymbol{\tau}_\kappa \in \mathcal{S}}\sum_{t=0}^{T}\left(V_{\boldsymbol{\zeta}}^{\phi_2^{[j][k]}}(\boldsymbol{x}_t^{(\kappa)}) - \sum_{t'=t}^{T}\gamma^{t'-t}\zeta_{\text{LOO}}(\boldsymbol{x}_{t'}^{(\kappa)}, \boldsymbol{u}_{t'}^{(\kappa)})\right)\nabla_{\phi_1}V_{\boldsymbol{\zeta}}^{\phi_2^{[j][k]}}(\boldsymbol{x}_t^{(\kappa)})$
18:        Update $\boldsymbol{\theta}^{[j][k]}$ (§ 3.1)
               $\boldsymbol{\theta}^{[j][k+1]} = \boldsymbol{\theta}^{[j][k]} - \frac{\eta_k^{\boldsymbol{\theta}}}{N}\sum_{\boldsymbol{\tau}_\kappa \in \mathcal{S}}\Big[\sum_{t=0}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\boldsymbol{u}_t^{(\kappa)}|\boldsymbol{x}_t^{(\kappa)})\Big(\lambda_\pi^\star A_{\mathcal{C}}^{\phi_1^{[j][k]}}(\boldsymbol{x}_t^{(\kappa)}, \boldsymbol{u}_t^{(\kappa)})$
                   $- (1-\lambda_\pi^\star)A_{\boldsymbol{\zeta}}^{\phi_2^{[j][k]}}(\boldsymbol{x}_t^{(\kappa)}, \boldsymbol{u}_t^{(\kappa)})\Big) + \lambda_{\text{CVaR}}\nabla_{\boldsymbol{\theta}}\text{CVaR}_\alpha(\mathcal{L}(\boldsymbol{\tau}_\kappa))\Big]$
19:    Set $\boldsymbol{\theta}^{[j+1][0]} = \boldsymbol{\theta}^{[j][K]}$, $\phi_1^{[j+1][0]} = \phi_1^{[j][K]}$, $\phi_2^{[j+1][0]} = \phi_2^{[j][K]}$
20: Return $\boldsymbol{\theta}^{[J][0]}$
```

Algorithm 3 Scheduling $\lambda_\pi^\star$

```
Solve for $\lambda_\pi^{[j][k],\star} = \arg\min_{\lambda_\pi} ||\lambda_\pi \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}}[\mathcal{C}(\boldsymbol{\tau})] - (1-\lambda_\pi)\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}}[\boldsymbol{\zeta}_{\mathrm{LOO}}(\boldsymbol{\tau})]||_2^2$:
if $\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}}[\mathcal{C}(\boldsymbol{\tau})]^\mathsf{T} \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}}[\boldsymbol{\zeta}_{\mathrm{LOO}}(\boldsymbol{\tau})] \geq ||\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}}[\mathcal{C}(\boldsymbol{\tau})]||_2^2$ then $\lambda_\pi^{[j][k],\star} = 1$
else if $\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}}[\mathcal{C}(\boldsymbol{\tau})]^\mathsf{T} \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}}[\boldsymbol{\zeta}_{\mathrm{LOO}}(\boldsymbol{\tau})] \geq ||\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}}[\boldsymbol{\zeta}_{\mathrm{LOO}}(\boldsymbol{\tau})]||_2^2$ then $\lambda_\pi^{[j][k],\star} = 0$
else
    $\lambda_\pi^{[j][k],\star} = \frac{(\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}}[\boldsymbol{\zeta}_{\mathrm{LOO}}(\boldsymbol{\tau})] - \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}}[\mathcal{C}(\boldsymbol{\tau})])^\mathsf{T} \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}}[\boldsymbol{\zeta}_{\mathrm{LOO}}(\boldsymbol{\tau})]}{||\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}}[\boldsymbol{\zeta}_{\mathrm{LOO}}(\boldsymbol{\tau})] - \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}}[\mathcal{C}(\boldsymbol{\tau})]||_2^2}$
```

B Acquisition functions

B.1 Acquisition function intuition

We implement our code in GPyTorch (Gardner et al. 2018), which computes a Cholesky decomposition of $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^\mathsf{T}$ during training. The use of the triangular matrix $\boldsymbol{L}$ allows the problem to be divided in such a way that memory requirements are reduced, as most matrix multiplications can be pre-computed and re-used when needed. Note also that the diagonal elements of $\boldsymbol{A}$ can be readily computed as $a_{i,i} = \sum_{j=1}^{i} L_{i,j}^2$. All these operations considered, the total computational burden of a full-set LOO distribution is reduced from $\mathcal{O}(\mathrm{N}^4)$ to $\mathcal{O}(\mathrm{N}^3)$. The number $\mathcal{O}(\mathrm{N}^3)$ is a result of N matrix multiplications (each per sample) costing $\mathcal{O}(\mathrm{N}^2)$ each. Note that we have eliminated matrix inversion contribution to the cost as we show in Appendix B.2, and these computations can be distributed fairly easily. Furthermore, as we mentioned in the main text $\mathcal{O}(N^3)$ is also the complexity of the exact GP, which comes from matrix inversion and cannot be easily distributed. The space complexity of our batch operations, however, can increase due to the need of storing a large tensor. Rather than doing so, we implement a batched version that eliminates such a need and achieves desirable results.

To illustrate what information the LOO acquisition function brings, we analyse under which conditions its value is null. Coming back to our definition, we can write explicitly the KL-divergence from $p(\boldsymbol{f}_\star|\mathcal{D}_{\neg i})$ to $p(\boldsymbol{f}_\star|\mathcal{D})$, for a single test data-point (i.e., $n_2 = 1$):

$$\mathrm{KL}\big(p(\boldsymbol{f}_\star|\mathcal{D}_{\neg i})||p(\boldsymbol{f}_\star|\mathcal{D})\big) = \frac{1}{2}\left(\underbrace{\left(\frac{\mu_\star - \mu_\star^{(i)}}{\sigma_{n_2,n_2}}\right)^2}_{\geq 0} + \underbrace{\left(\frac{\sigma_{n_2,n_2}^{(i)}}{\sigma_{n_2,n_2}}\right)^2 - \log\left(\frac{\sigma_{n_2,n_2}{}^{(i)}}{\sigma_{n_2,n_2}}\right)^2 - 1}_{\geq 0}\right), \tag{5}$$

where we use non-bold and non-capitalised fonts in order to signify that $\sigma_{n_2,n_2}^{(i)}$, σ_{n_2,n_2}, $\mu_\star^{(i)}$ and $\mu_\star$ are scalars. Note that the second expression is non-negative since $\log(x) \leq x - 1$ for all x and the equality is achieved if and only if $x = 1$. Therefore for the KL divergence in Equation (5) to be zero, two main conditions need to be met: $\mu_\star^{(i)} = \mu_\star$ and $\sigma_{n_2,n_2}^{(i)} = \sigma_{n_2,n_2}$. Since in general $\sigma_{n_2,n_2}^{(i)} = \sigma_{n_2,n_2} + \boldsymbol{K}_{n_2,n_1}\boldsymbol{a}_i^\mathsf{T}\boldsymbol{a}_i\boldsymbol{K}_{n_2,n_1}/a_{i,i}$ we obtain another necessary condition for the KL divergence being equal to zero: $\boldsymbol{K}_{n_2,n_1}\boldsymbol{a}_i^\mathsf{T} = 0$. Now we will show that $\boldsymbol{K}_{n_2,n_1}\boldsymbol{a}_i^\mathsf{T} = 0$ (and hence $\sigma_{n_2,n_2}^{(i)} = \sigma_{n_2,n_2}$) is also sufficient. Developing the condition $\mu_\star^{(i)} = \mu_\star$ we have $\boldsymbol{K}_{n_2,n_1}\boldsymbol{a}_i^\mathsf{T}\boldsymbol{a}_i\boldsymbol{y}_{n_1} = 0$ since $\mu_\star^{(i)} = \mu_\star - \boldsymbol{K}_{n_2,n_1}\boldsymbol{a}_i^\mathsf{T}\boldsymbol{a}_i\boldsymbol{y}_{n_1}$. As both $\boldsymbol{K}_{n_2,n_1}\boldsymbol{a}_i^\mathsf{T}$ and $\boldsymbol{a}_i\boldsymbol{y}_{n_1}$ are scalars this equivalently amounts to $\boldsymbol{K}_{n_2,n_1}\boldsymbol{a}_i^\mathsf{T} = 0$ or $\boldsymbol{a}_i\boldsymbol{y}_{n_1} = 0$. Therefore if $\boldsymbol{K}_{n_2,n_1}\boldsymbol{a}_i^\mathsf{T} = 0$ (or equivalently $\sigma_{n_2,n_2}^{(i)} = \sigma_{n_2,n_2}$) then $\mu_\star = \mu_\star^{(i)}$ and the KL divergence in Equation (5) is equal to zero. In other words, the KL divergence between the two distribution will be zero if and only if the covariance of the test data and the target training data is zero when all other points are observed. As a consequence, the LOO acquisition function can be seen as a proxy to the expected reduction of uncertainty about training output values conditioned on the observation of the queried datapoint.

The intuition behind these conditions is that the expected KL divergence between LOO and the full distributions answers a completely different question when compared to entropy bonus: whereas entropy bonus indicates how much "absolute" uncertainty there is about a *test* output – which we always expect to be high for points situated at a distance of the training inputs – the LOO acquisition function indicates how much, on average, we can learn (i.e. how big would be the reduction in uncertainty would be) about the *training* outputs if we were to observe the queried test data. Therefore, the LOO acquisition function truly answers the question of how much we can learn about the current model when observing the queried point.

B.2 LOO computation

Without loss of generality, assume that we leave the last sample out, i.e., we will set $i = n_1$. We have the following expressions for the means:

$$\begin{aligned} \boldsymbol{\mu}_\star &= \boldsymbol{K}_{n_2,n_1}\boldsymbol{A}_{n_1,n_1}\boldsymbol{y}_{n_1}, \\ \boldsymbol{\mu}_\star^{(n_1)} &= \boldsymbol{K}_{n_2,n_1-1}\boldsymbol{A}_{n_1-1,n_1-1}\boldsymbol{y}_{n_1-1}, \end{aligned} \tag{6}$$

and the covariance matrices:

$$\begin{aligned} \boldsymbol{\Sigma}_{n_2,n_2} &= \boldsymbol{K}_{n_2,n_2} - \boldsymbol{K}_{n_2,n_1}\boldsymbol{A}_{n_1,n_1}\boldsymbol{K}_{n_1,n_2}, \\ \boldsymbol{\Sigma}_{n_2,n_2}^{(n_1)} &= \boldsymbol{K}_{n_2,n_2} - \boldsymbol{K}_{n_2,n_1-1}\boldsymbol{A}_{n_1-1,n_1-1}\boldsymbol{K}_{n_1-1,n_2}, \end{aligned} \tag{7}$$

where

$$\begin{aligned} \boldsymbol{A}_{n_1,n_1} &= \left[\boldsymbol{K}_{n_1,n_1} + \sigma_\omega^2\boldsymbol{I}\right]^{-1}, \\ \boldsymbol{A}_{n_1-1,n_1-1} &= \left[\boldsymbol{K}_{n_1-1,n_1-1} + \sigma_\omega^2\boldsymbol{I}\right]^{-1}. \end{aligned}$$

The main difficulty in computing the updates $\boldsymbol{\mu}_\star^{(n_1)} - \boldsymbol{\mu}_\star$, $\boldsymbol{\Sigma}_{n_2,n_2}^{(n_1)} - \boldsymbol{\Sigma}_{n_2,n_2}$ is actually dealing with the update involving the matrices $\boldsymbol{A}_{n_1-1,n_1-1}$ and $\boldsymbol{A}_{n_1,n_1}$.

Note that there exist $\boldsymbol{b}_0$, b_1, c_0 such that:

$$\begin{aligned} \boldsymbol{K}_{n_2,n_1} &= \begin{bmatrix}\boldsymbol{K}_{n_2,n_1-1} & c_0\end{bmatrix} \\ \boldsymbol{K}_{n_1,n_1} &= \begin{bmatrix}\boldsymbol{K}_{n_1-1,n_1-1} & \boldsymbol{b}_0^\top \\ \boldsymbol{b}_0 & b_1\end{bmatrix} \end{aligned} \tag{8}$$

We have

$$\boldsymbol{A}_{n_1,n_1} = \left[\boldsymbol{K}_{n_1,n_1} + \sigma_\omega^2\boldsymbol{I}\right]^{-1} = \begin{bmatrix}\boldsymbol{A}_{\neg n_1,\neg n_1} & \boldsymbol{a}_{n_1,\neg n_1}^\top \\ \boldsymbol{a}_{n_1,\neg n_1} & a_{n_1,n_1}\end{bmatrix}$$

As $\boldsymbol{A}_{n_1,n_1}^{-1} = \boldsymbol{K}_{n_1,n_1} + \sigma_\omega^2\boldsymbol{I}$, using the block inversion lemma it is straightforward to show that:

$$\boldsymbol{K}_{n_1-1,n_1-1} + \sigma_\omega^2 \boldsymbol{I} = \left[\boldsymbol{A}_{\neg n_1,\neg n_1} - \boldsymbol{a}^{\top}_{n_1,\neg n_1} a^{-1}_{n_1,n_1} \boldsymbol{a}_{n_1,\neg n_1}\right]^{-1},$$

and consequently:

$$\boldsymbol{A}_{n_1-1,n_1-1} = \left[\boldsymbol{K}_{n_1-1,n_1-1} + \sigma_\omega^2 \boldsymbol{I}\right]^{-1} = \boldsymbol{A}_{\neg n_1,\neg n_1} - \boldsymbol{a}^{\top}_{n_1,\neg n_1} a^{-1}_{n_1,n_1} \boldsymbol{a}_{n_1,\neg n_1},$$

For simplicity, let us introduce the following matrix:

$$\begin{aligned} &\boldsymbol{\Delta} = \boldsymbol{A}_{n_1,n_1} - \begin{bmatrix} \boldsymbol{A}_{n_1-1,n_1-1} & 0 \\ 0 & 0 \end{bmatrix} = \\ &\begin{bmatrix} \boldsymbol{A}_{\neg n_1,\neg n_1} & \boldsymbol{a}^{\top}_{n_1,\neg n_1} \\ \boldsymbol{a}_{n_1,\neg n_1} & a_{n_1,n_1} \end{bmatrix} - \begin{bmatrix} \boldsymbol{A}_{\neg n_1,\neg n_1} - \boldsymbol{a}^{\top}_{n_1,\neg n_1} a^{-1}_{n_1,n_1} \boldsymbol{a}_{n_1,\neg n_1} & 0 \\ 0 & 0 \end{bmatrix} = \\ &\begin{bmatrix} \boldsymbol{a}^{\top}_{n_1,\neg n_1} a^{-1}_{n_1,n_1} \boldsymbol{a}_{n_1,\neg n_1} & \boldsymbol{a}^{\top}_{n_1,\neg n_1} \\ \boldsymbol{a}_{n_1,\neg n_1} & a_{n_1,n_1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}^{\top}_{n_1,\neg n_1} \\ a_{n_1,n_1} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_{n_1,\neg n_1} & a_{n_1,n_1} \end{bmatrix} / a_{n_1,n_1} = \frac{\boldsymbol{a}^{\top}_{n_1} \boldsymbol{a}_{n_1}}{a_{n_1,n_1}}, \end{aligned}$$

where $\boldsymbol{a}_{n_1} = \left[\boldsymbol{a}_{n_1,\neg n_1} \;\; a_{n_1,n_1}\right]$ is the n_1-th row of the matrix $\boldsymbol{A}_{n_1,n_1}$.

Now:

$$\begin{aligned} &\boldsymbol{\Sigma}^{(n_1)}_{n_2,n_2} - \boldsymbol{\Sigma}_{n_2,n_2} = \boldsymbol{K}_{n_2,n_1} \boldsymbol{A}_{n_1,n_1} \boldsymbol{K}_{n_1,n_2} - \boldsymbol{K}_{n_2,n_1-1} \boldsymbol{A}_{n_1-1,n_1-1} \boldsymbol{K}_{n_1-1,n_2} = \\ &\left[\boldsymbol{K}_{n_2,n_1-1} \;\; c_0\right] \boldsymbol{\Delta} \begin{bmatrix} \boldsymbol{K}_{n_1-1,n_2} \\ c_0 \end{bmatrix} = \boldsymbol{K}_{n_2,n_1} \frac{\boldsymbol{a}^{\top}_{n_1} \boldsymbol{a}_{n_1}}{a_{n_1,n_1}} \boldsymbol{K}_{n_1,n_2}. \end{aligned}$$

Similarly,

$$\begin{aligned} &\boldsymbol{\mu}^{(n_1)}_{\star} - \boldsymbol{\mu}_{\star} = \boldsymbol{K}_{n_2,n_1-1} \boldsymbol{A}_{n_1-1,n_1-1} \boldsymbol{y}_{n_1-1} - \boldsymbol{K}_{n_2,n_1} \boldsymbol{A}_{n_1,n_1} \boldsymbol{y}_{n_1} = \\ &- \boldsymbol{K}_{n_2,n_1} \boldsymbol{\Delta} \boldsymbol{y}_{n_1} = -\boldsymbol{K}_{n_2,n_1} \frac{\boldsymbol{a}^{\top}_{n_1} \boldsymbol{a}_{n_1}}{a_{n_1,n_1}} \boldsymbol{y}_{n_1} \end{aligned}$$

Hence, we have that

$$\begin{aligned} \boldsymbol{\mu}^{(i)}_{\star} &= \boldsymbol{\mu}_{\star} - \boldsymbol{K}_{n_2,n_1} \frac{\boldsymbol{a}^{\top}_{i} \boldsymbol{a}_{i}}{a_{i,i}} \boldsymbol{y}_{n_1}, \\ \boldsymbol{\Sigma}^{(i)}_{n_2,n_2} &= \boldsymbol{\Sigma}_{n_2,n_2} + \boldsymbol{K}_{n_2,n_1} \frac{\boldsymbol{a}^{\top}_{i} \boldsymbol{a}_{i}}{a_{i,i}} \boldsymbol{K}_{n_1,n_2} \end{aligned}$$

for any i(Table 5).

C Environmental settings

We limit all environments to have a maximum horizon/ trajectory length of 100 steps. We implemented a safety cost function with the following settings; The unsafe region started at $USR_{min} = 20\pi/180$ and ended at $USR_{max} = 30\pi/180$, therefore if θ_1 was in $USR_{min} \leq \theta_1 \leq USR_{max}$ we would refer to this as a violation and safety cost would be incurred. The Hazard Region contained within $HZ_{min} = USR_{min} - \pi/4 \leq HZ \leq USR_{max} + \pi/4 = HZ_{max}$. Where θ_1 is transformed to

Table 5 Experiment hyper-parameters

		Safe pendulum	Safe cartpole double pendulum	Safe fetch reacher
Dynamics Model	N samples	30	30	300
	N init samples	30	30	1000
	Optimizer	FullBatchLBFGS (Hjmshi 2018)	FullBatchLBFGS (Hjmshi 2018)	FullBatchLBFGS (Hjmshi 2018)
	Predict delta states	True	True	True
	Normalize states	True	True	True
	GP model	Exact	Exact	Exact
	Objective function	Exact Marginal Log Likelihood	Exact Marginal Log Likelihood	Exact Marginal Log Likelihood
	kernel	RBF	RBF	RBF
	CG tolerance	0.00001	0.00001	0.00001
	Max preconditioner size	200	30	150
	Max cg iterations	15000	15000	15000
	Optimization iterations	300	300	100
	Learning rate	0.1	0.1	0.0001
Agent	N update epochs	80	80	80
	Mini batch size	30000	30000	90000
	N neurons	32	32	32
	Policy learning rate	3e-4	3e-4	3e-4
	Value function learning rate	1e-3	1e-3	1e-3
	Penalty learning rate	5e-2	5e-2	5e-2
	Clipping	0.2	0.2	0.2
	Max gradiant norm	0.5	0.5	1.0
	Gamma	0.99	0.95	0.99
	Lambda	0.97	0.97	0.95
	Value function tragets	Monte Carlo	Monte Carlo	Monte Carlo
	Value function loss fn	Squared loss	Squared loss	Clipped loss
	CVaR cost limit	0.025	0.025	0.0166666667
	CVaR risk	$\alpha \in (0.5, 1.0)$	$\alpha \in (0.5, 1.0)$	$\alpha \in (0.5, 1.0)$

Table 5 (continued)

		Safe pendulum	Safe cartpole double pendulum	Safe fetch reacher
Runner	Max trajectory length	30	30	30
	N training epochs	100	100	100
	N epoch samples	30000	30000	90000

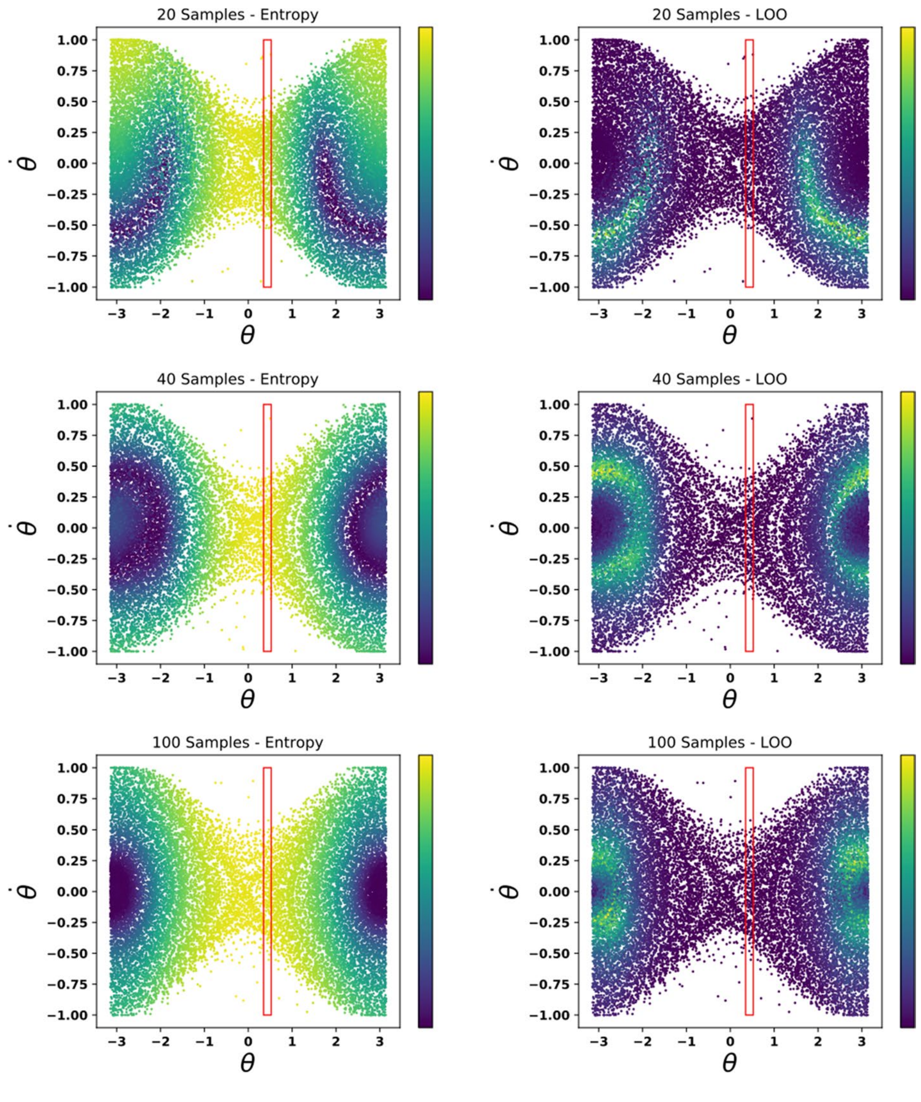


Fig. 5 Comparison between Entropy vs. LOO values across the entire state space of Safe Pendulum across true sampled data points. Yellow corresponds to a higher acquisition function values, while blue to a lower. Note that PPO (or any policy used within this framework) is encouraged to visit regions with higher (yellow) weight. The red box indicates the unsafe region of the state space (Color figure online)

always remain in the region $\theta_1 \in [-\pi, \pi]$. Therefore, safety cost is linearly proportional to the distance from the edge of the hazard region (HZ_{min} or HZ_{max}) to the centre of the hazard region $(HZ_{max} + HZ_{min})/2$. We also implemented batched versions of the pendulum reward function as well as batched versions of the pendulum safety function using torch. We also implement the unsafe region centre as the middle between the fetchers starting position and its goal (with respect to the x position only), then the hazard region expands around this point with respect to 1/4 of the total distance between the fetchers start state and goal

Fig. 6 Comparison Entropy vs. LOO values across the entire state space of Safe Pendulum and Safe Cartpole Double Pendulum. Yellow corresponds to a higher acquisition function value, while blue to a lower. Note that PPO (or any policy used within this framework) is encouraged to visit regions with higher (yellow) weight. The red box indicates the unsafe region of the state space (Color figure online) ▶

state. Note, we terminated Safe Pendulum once the last 5 rewards in a trajectory were all ≤ -0.01, as this provides an adequately stabilised pendulum.

For the safety gym robots we take the following state values: velocities, magnetometer measurements, centre of mass position of the robot (all in the x and y axes) and gyroscope measurement (only in z axis). This choice is made to increase the dimension of the observation space, while also making the environment more GP-friendly. For instance, we remove the velocity, magnetometer and position readouts in the z axis as our car moves in a two dimensional plane. Similarly, we include only the measurement of the yaw angle rate, which is provided by the gyroscope measurement in the z axis. The robot's initial position is at (0, 0) coordinates, while the goal is placed at (3, 3) and marked in green in Fig. 1. The centre of the hazard region of radius 0.5 (marked in blue in Fig. 1), which the robot should avoid, is placed halfway between the robot's initial position and the goal. Note that we do not place other objects in the safety gym such as pillars, buttons etc.

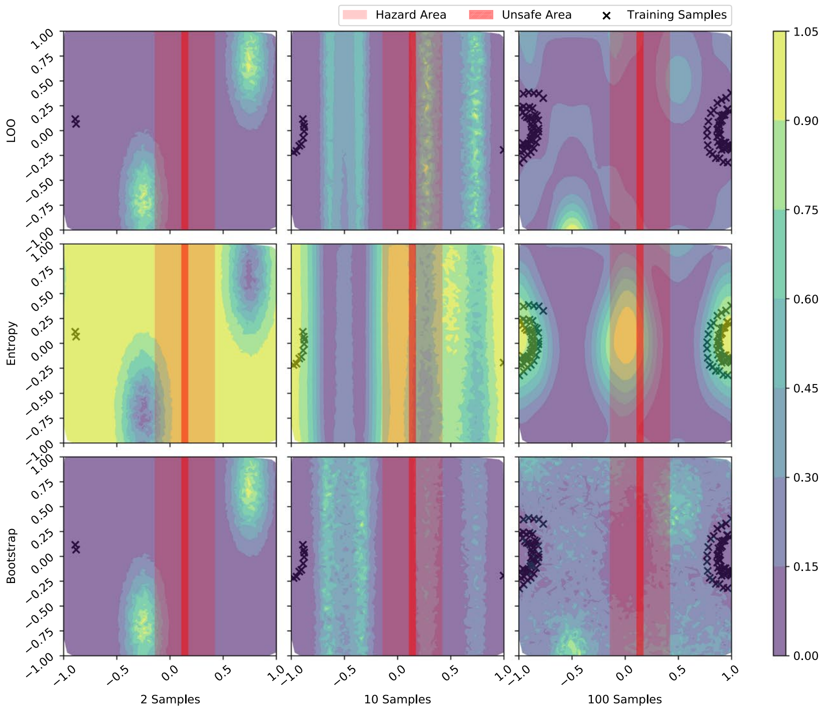
D Experimental evaluation

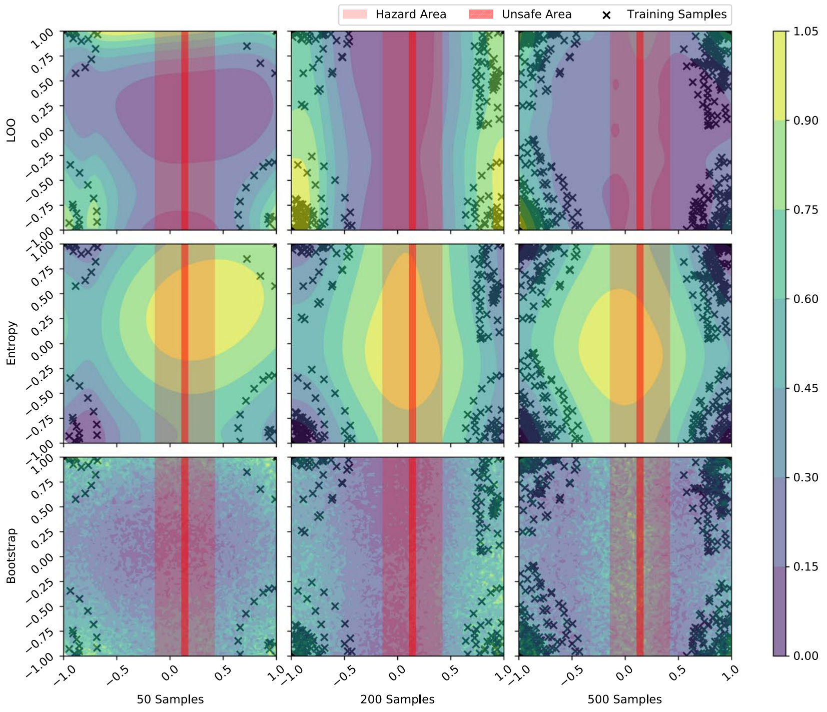
D.1 LOO performance analysis

This section empirically compares the typical entropy-based method with LOO. For comparison, we collected N random rollouts from Safe Pendulum to train our GP. Next, we calculated the entropy/LOO acquisition function for unseen random rollouts totaling 20, 000 test samples. Figure 5 shows the result of this evaluation, where we compared Entropy vs. LOO for differing sizes of training data N for the GP ($N = 20$, $N = 40$ and $N = 100$). The red box contains the USR (Unsafe Region). The experiment suggests that LOO is much more conservative and will aim to guide the policy away from the dangerous area (Fig. 6). At the same time, entropy encourages exploration into the unsafe area, additionally encouraged with the increase of N - a highly undesirable property when safe active exploration is required (Fig. 7).

D.2 Dynamics model analysis

This section presents and compares sample traces between different GP dynamics models trained on a different number of data points (20, 40, and 100 samples). The procedure for


Hazard Area
Unsafe Area
Training Samples
LOO
Entropy
Bootstrap
2 Samples
10 Samples
100 Samples
1.05
0.90
0.75
0.60
0.45
0.30
0.15
0.00


Hazard Area
Unsafe Area
Training Samples
LOO
Entropy
Bootstrap
50 Samples
200 Samples
500 Samples
1.05
0.90
0.75
0.60
0.45
0.30
0.15
0.00

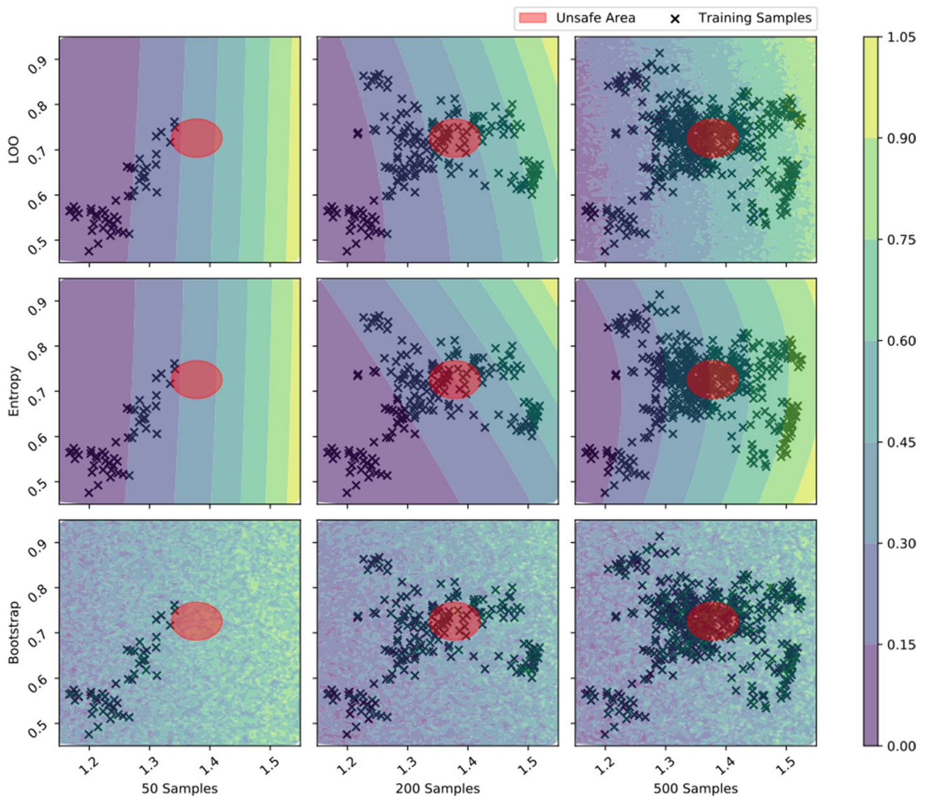


Fig. 7 Comparison Entropy vs. LOO values across the entire state space of Safety Fetcher. Yellow corresponds to a higher acquisition function value, while blue to a lower. Note that PPO (or any policy used within this framework) is encouraged to visit regions with higher (yellow) weight. The red box indicates the unsafe region of the state space (Color figure online)

the comparison is the following: We rollout the real environment for one full trajectory with random actions and record states and actions. For each GP, we show three sample traces of length 100 time steps (dashed blue line). The sampled traces are compared to the true trajectories by concatenating the true action taken (from the true trajectory) to the open-loop predicted states. We can see in Fig. 8 how even when the dynamical system differences to the model, there is still many similarities to the transition dynamics (Fig. 9).

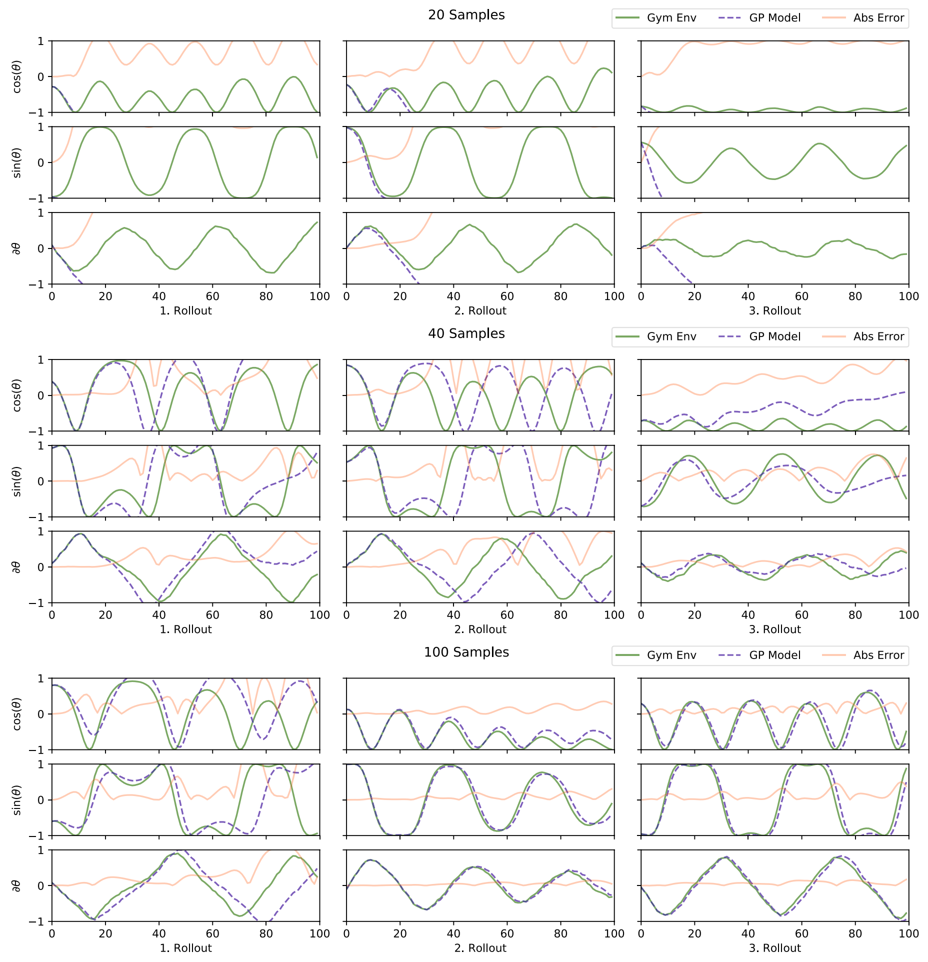


Fig. 8 Samples trace from Pendulum

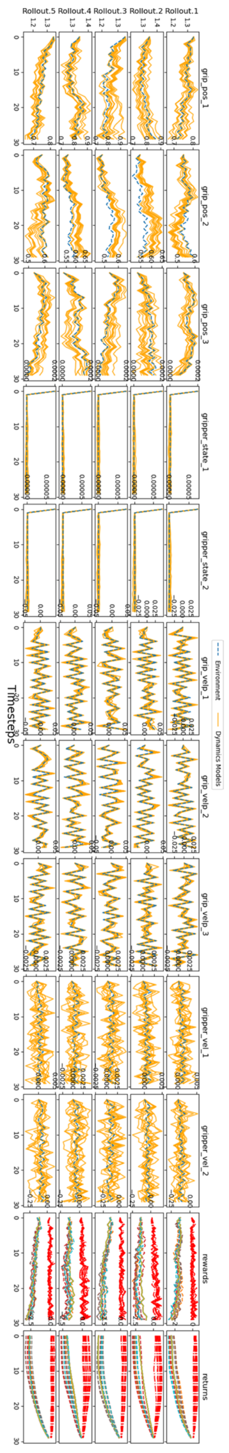


Fig. 9 Sampled traces from Safe Fetch Reacher

Author Contributions AI.C-R - Worked on algorithmic design and experiment design and analysis. DP - Worked on algorithmic and environment design. VM - Worked on developing novel acquisition functions LOO and bootstrap. MAA - Worked on problem formulation and algorithmic design. AS - Worked on problem formulation and algorithmic design. JW - Worked on problem formulation and analysis. HB-A - Worked on problem formulation and analysis.

Funding This study was funded by Huawei R&D London.

Data availability Simulators are freely available online https://github.com/openai/safety-gym and https://gym.openai.com/.

Declarations

Conflict of interest All authors received a salary from Huawei while the research was conducted.

Code availability While all baselines can be readily found online such as https://github.com/openai/safety-starter-agents, https://github.com/facebookresearch/mbrl-lib and https://github.com/Kaixhin/PlaNet. We are currently working on releasing the SAMBA agents.

References

Achiam, J., Held, D., Tamar, A., & Abbeel, P. (2017). Constrained policy optimization. *In: International conference on machine learning*, pp 22–31.

Akametalu, A.K., Fisac, J.F., Gillula, J.H., Kaynama, S., Zeilinger, M.N., & Tomlin, C.J. (2014). Reachability-based safe learning with gaussian processes. *In: IEEE conference on decision and control*, pp 1424–1431.

Altman, E. (1999). *Constrained markov decision processes*. CRC Press.

Ammar, H.B., Eaton, E., Ruvolo, P., & Taylor, M. (2014). Online multi-task learning for policy gradient methods. *In: International conference on machine learning*, pp 1206–1214.

Aswani, A., Gonzalez, H., Sastry, S. S., & Tomlin, C. (2013). Provably safe and robust learning-based model predictive control. *Automatica, 49*(5), 1216–1226.

Ball, P., Parker-Holder, J., Pacchiano, A., Choromanski, K., & Roberts, S. (2020). Ready policy one: World building through active learning. arXiv preprint arXiv:200202693

Berkenkamp, F., Moriconi, R., Schoellig, A.P., & Krause, A. (2016). Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes. *In: 2016 IEEE 55th conference on decision and control (CDC), IEEE*, pp 4661–4666.

Berkenkamp, F., Turchetta, M., Schoellig, A., & Krause, A. (2017). Safe model-based reinforcement learning with stability guarantees. *In: Advances in neural information processing systems*, pp 908–918.

Bertsekas, D. P. (1997). Nonlinear programming. *Journal of the Operational Research Society, 48*(3), 334.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. arXiv preprint arXiv:160601540

Buisson-Fenet, M., Solowjow, F., & Trimpe, S. (2019). Actively learning gaussian process dynamics. arXiv preprint arXiv:191109946

Camacho, E.F., & Alba, C.B. (2013). Model predictive control. *Springer Science & Business Media.*

Chow, Y., & Ghavamzadeh, M. (2014). Algorithms for cvar optimization in mdps. *In: Advances in neural information processing systems*, pp 3509–3517

Chow, Y., Tamar, A., Mannor, S., & Pavone, M. (2015). Risk-sensitive and robust decision-making: a cvar optimization approach. *In: Advances in neural information processing systems*, pp 1522–1530.

Chow, Y., Ghavamzadeh, M., Janson, L., & Pavone, M. (2017). Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research, 18*(1), 6070–6120.

Chow, Y., Nachum, O., Duenez-Guzman, E., & Ghavamzadeh, M. (2018). A lyapunov-based approach to safe reinforcement learning. *In: Advances in neural information processing systems*, pp 8092–8101.

Chow, Y., Nachum, O., Faust, A., Ghavamzadeh, M., & Duenez-Guzman, E. (2019). Lyapunov-based safe policy optimization for continuous control. arXiv preprint arXiv:190110031

Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., & Tassa, Y. (2018). Safe exploration in continuous action spaces. arXiv preprint arXiv:180108757

Damianou, A., Titsias, M.K., Lawrence, N.D. (2011). Variational gaussian process dynamical systems. *In: Advances in neural information processing systems*, pp 2510–2518.

de Wolff, T., Cuevas, A., & Tobar, F. (2020). Mogptk: The multi-output gaussian process toolkit. arXiv preprint arXiv:200203471

Deisenroth, M., & Rasmussen, C.E. (2011). Pilco: A model-based and data-efficient approach to policy search. *In: International conference on machine learning*, pp 465–472.

Fedorov, V. V. (2013). *Theory of optimal experiments*. Elsevier.

Gal, Y., Islam, R., & Ghahramani, Z. (2017). Deep bayesian active learning with image data. arXiv preprint arXiv:170302910

Gardner, J., Pleiss, G., Weinberger, K.Q., Bindel, D., & Wilson, A.G. (2018). Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *In: Advances in neural information processing systems*, pp 7576–7586

Goh, C., & Yang, X. (2001). Nonlinear lagrangian theory for nonconvex optimization. *Journal of Optimization Theory and Applications, 109*(1), 99–121.

Ha, D., & Schmidhuber, J. (2018). World models. arXiv preprint arXiv:180310122

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., & Davidson, J. (2019). Learning latent dynamics for planning from pixels. *In: International conference on machine learning*.

Hafner, D., Lillicrap, T., Ba, J., & Norouzi, M. (2020). Dream to control: Learning behaviors by latent imagination. In: International conference on learning representations.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. *In: AAAI conference on artificial intelligence*.

Hjmshi. (2018). hjmshi/pytorch-lbfgs. https://githubcom/hjmshi/PyTorch-LBFGS

Jain, A., Nghiem, T., Morari, M., & Mangharam, R. (2018). Learning and control using gaussian processes. *In: ACM/IEEE international conference on cyber-physical systems*, pp 140–149.

Janner, M., Fu, J., Zhang, M., & Levine, S. (2019). When to trust your model: Model-based policy optimization. arXiv preprint arXiv:190608253

Kamthe, S., & Deisenroth, M.P. (2018). Data-efficient reinforcement learning with probabilistic model predictive control. *In: International conference on artificial intelligence and statistics*.

Khalil, H. K., & Grizzle, J. W. (2002). *Nonlinear systems* (Vol. 3). Prentice hall Upper Saddle River.

Koller, T., Berkenkamp, F., Turchetta, M., & Krause, A. (2018). Learning-based model predictive control for safe exploration. *In: IEEE conference on decision and control*, pp 6059–6066.

Krause, A., & Guestrin, C. (2007). Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. *In: International conference on machine learning*, pp 449–456.

Krause, A., Singh, A., & Guestrin, C. (2008). Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research, 9*(Feb), 235–284. 9(Feb):235–284.

Mayne, D. Q., Seron, M. M., & Raković, S. (2005). Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica, 41*(2), 219–224.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:13125602

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533.

Petersen, K., & Pedersen, M., et al. (2008). The matrix cookbook, vol. 7. Technical University of Denmark 15.

Polymenakos, K., Abate, A., & Roberts, S. (2019). Safe policy search using gaussian process models. *In: Proceedings of the 18th international conference on autonomous agents and multiagent systems*, pp 1565–1573.

Polymenakos, K., Rontsis, N., Abate, A., & Roberts, S. (2020). SafePILCO: A software tool for safe and data-efficient policy synthesis. In D. N. Jansen & A. Remke (Eds.), *Gribaudo M* (pp. 18–26). Quantitative Evaluation of Systems: Springer International Publishing.

Prashanth, L. (2014). Policy gradients for cvar-constrained mdps. In: International conference on algorithmic learning theory, Springer, pp 155–169.

Prashanth, L., & Ghavamzadeh, M. (2013). Actor-critic algorithms for risk-sensitive mdps. *In: Advances in neural information processing systems*, pp 252–260.

Rasmussen, C. E., & Williams, C. K. I. (2005). *Gaussian processes for machine learning (Adaptive computation and machine learning)*. The MIT Press.

Ray, A., Achiam, J., & Amodei, D. (2019). Benchmarking safe exploration in deep reinforcement learning. https://cdn.openai.com/safexp-short.pdf.

Rockafellar, R. T., Uryasev, S., et al. (2000). Optimization of conditional value-at-risk. *Journal of Risk, 2*, 21–42.

Saphal, R., Ravindran, B., Mudigere, D., Avancha, S., & Kaul, B. (2020). Seerl: Sample efficient ensemble reinforcement learning. arXiv preprint arXiv:200105209.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. *In: International conference on machine learning*, pp 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:170706347.

Schultheis, M., Belousov, B., Abdulsamad, H., & Peters, J. (2019). Receding horizon curiosity. *In: Conference on robot learning*.

Settles, B. (2009). *Active learning literature survey*. Tech. rep.: University of Wisconsin-Madison Department of Computer Sciences.

Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review, 5*(1), 3–55.

Shyam, P., Jaśkowski, W., & Gomez, F. (2019). Model-based active exploration. *In: International conference on machine learning*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature, 529*(7587), 484–489.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., et al. (2017). Mastering the game of go without human knowledge. *Nature, 550*(7676), 354–359.

Srinivas, A., Laskin, M., & Abbeel, P. (2020). Curl: Contrastive unsupervised representations for reinforcement learning. arXiv preprint arXiv:200404136.

Sutton, R.S., & Barto, A.G. (2018). Reinforcement learning: An introduction. *MIT press*

van Amersfoort, J., Smith, L., Teh, Y.W., & Gal, Y. (2020). Simple and scalable epistemic uncertainty estimation using a single deep deterministic neural network. arXiv preprint arXiv:200302037

Zimmer, C., Meister, M., & Nguyen-Tuong, D. (2018). Safe active learning for time-series modeling with gaussian processes. *In: Advances in neural information processing systems*, pp 2730–2739.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.