Fingerprint Policy Optimisation for Robust Reinforcement Learning

Supratik Paul [1] **Michael A. Osborne** [2] **Shimon Whiteson** [1]

Abstract

Policy gradient methods ignore the potential value of adjusting *environment variables*: unobservable state features that are randomly determined by the environment in a physical setting, but are controllable in a simulator. This can lead to slow learning, or convergence to suboptimal policies, if the environment variable has a large impact on the transition dynamics. In this paper, we present *fingerprint policy optimisation* (FPO), which finds a policy that is optimal in expectation across the distribution of environment variables. The central idea is to use Bayesian optimisation (BO) to actively select the distribution of the environment variable that maximises the improvement generated by each iteration of the policy gradient method. To make this BO practical, we contribute two easy-to-compute low-dimensional *fingerprints* of the current policy. Our experiments show that FPO can efficiently learn policies that are robust to significant rare events, which are unlikely to be observable under random sampling, but are key to learning good policies.

1. Introduction

Policy gradient methods have demonstrated remarkable success in learning policies for various continuous control tasks (Lillicrap et al., 2016; Mordatch et al., 2015; Schulman et al., 2016). However, the expense of running physical trials, coupled with the high sample complexity of these methods, pose significant challenges in directly applying them to a physical setting, e.g., to learn a locomotion policy for a robot. Another problem is evaluating the robustness of a learned policy; it is difficult to ensure that the policy performs as expected, as it is usually infeasible to test it across all possible settings. Fortunately, policies can often be trained and tested in a simulator that exposes key *environment variables* – state features that are unobservable to the agent and randomly determined by the environment in a physical setting, but that are controllable in the simulator.

Environment variables are ubiquitous in real-world settings. For example, localisation errors may mean that a robot is much nearer to an obstacle than expected, increasing the risk of a collision. While an actual robot cannot directly trigger such errors, they are trivial to introduce in simulation. Similarly, if we want to train a helicopter to fly robustly, we can easily simulate different wind conditions, even though we cannot control weather in the real world.

A naïve application of a policy gradient method updates a policy at each iteration by using a batch of trajectories sampled from the original distribution over environment variables irrespective of the current policy or the training iteration. Thus, it does not explicitly take into account how environment variables affect learning for different policies. Furthermore, this approach is not robust to *significant rare events* (SREs), i.e., it fails any time there are rare events that have significantly different rewards thereby affecting expected performance.

For example, even if robot localisation errors are rare, coping with them properly may still be key to maximising expected performance, since the collisions they can trigger are so catastrophic. Similarly, handling certain wind conditions properly may be essential to robustly flying a helicopter, even if those conditions are rare, since doing so may avoid a crash. In such cases, the naïve approach will not see such rare events often enough to learn an appropriate response.

These problems can be avoided by learning *off environment* (Frank et al., 2008; Ciosek & Whiteson, 2017; Paul et al., 2018), i.e., exploiting the ability to adjust environment variables in simulation to trigger SREs more often and thereby improve the efficiency and robustness of learning. However, existing off environment approaches have significant disadvantages. They either require substantial prior knowledge about the environment and/or dynamics (Frank et al., 2008; Ciosek & Whiteson, 2017), can only be applied to low dimensional policies (Paul et al., 2018), or are highly sample

[1] Department of Computer Science, University of Oxford, UK [2] Department of Engineering Science, University of Oxford, UK. Correspondence to: Supratik Paul <supratik.paul@cs.ox.ac.uk>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

inefficient (Rajeswaran et al., 2017a).

In this paper, we propose a new off environment approach called *fingerprint policy optimisation* (FPO) that aims to learn policies that are robust to rare events while addressing the disadvantages mentioned above. At its core, FPO uses a policy gradient method as the policy optimiser. However, unlike the naïve approach, FPO explicitly models the effect of the environment variable on the policy updates, as a function of the policy. Using *Bayesian optimisation* (BO), FPO actively selects the environment distribution at each iteration of the policy gradient method in order to maximise the improvement that one policy gradient update step generates. While this can yield biased gradient estimates, FPO implicitly optimises the bias-variance tradeoff in order to maximise its one-step improvement objective.

A key design challenge in FPO is how to represent the current policy, in cases where the policy is a large neural network with thousands of parameters. To this end, we propose two low-dimensional policy *fingerprints* that act as proxies for the policy. The first approximates the stationary distribution over states induced by the policy, with a size equal to the dimensionality of the state space. The second approximates the policy's marginal distribution over actions, with a size equal to the dimensionality of the action space.

We apply FPO to different continuous control tasks and show that it can outperform existing methods, including those for learning in environments with SREs. We show that both fingerprints work equally well in practice, which implies that, for a given problem, the lower dimensional fingerprint can be chosen without sacrificing performance.

2. Problem Setting and Background

A *Markov decision process* (MDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}_\theta, r_\theta, s_0, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the set of actions, $\mathcal{P}$ the transition probabilities, r the reward function, s_0 the probability distribution over the initial state, and $\gamma \in [0, 1)$ the discount factor. We assume that the transition and reward functions depend on some environmental variables θ. At the beginning of each episode, the environment randomly samples θ from some (known) distribution $p(\theta)$. The agent's goal is to learn a policy $\pi(a|s)$ mapping states $s \in \mathcal{S}$ to actions $a \in \mathcal{A}$ that maximises the expected return $J(\pi) = \mathbb{E}_\theta[R(\theta, \pi)] = \mathbb{E}_{\theta \sim p(\theta), a_t \sim \pi, s_t \sim \mathcal{P}}[\sum_t \gamma^t r_\theta(s_t, a_t)]$. Note that π does not condition on θ at any point. With a slight abuse of notation, we use π to denote both the policy and its parameters. We consider environments characterised by significant rare events (SREs), i.e., there exist some low probability values of θ that generate large magnitude returns (positive or negative), yielding a significant impact on $J(\pi)$.

We assume that learning is undertaken in a simulator, or under laboratory conditions where θ can be actively set. Simulators are typically imperfect, which can lead to a *reality gap* that inhibits transfer to the real environment. Nonetheless, in most practical settings, e.g., autonomous vehicles and drones, training in the real environment is prohibitively expensive and dangerous, and simulators play an indispensable role.

2.1. Policy Gradient Methods

Starting with some policy π_n at iteration n, gradient based batch policy optimisation methods like REINFORCE (Williams, 1992), NPG (Kakade, 2001), and TRPO (Schulman et al., 2015) compute an estimate of the gradient $\nabla J(\pi_n)$ by sampling a batch of trajectories from the environment while following π_n, and then use this estimate to approximate gradient ascent in J, yielding an updated policy π_{n+1}. REINFORCE uses a fixed learning rate to update the policy; NPG and TRPO use the Fisher information matrix to scale the gradient and constrain the KL divergence between consecutive policies, which makes the updates independent of the policy parametrisation.

A major problem for such methods is that the estimate of $\nabla J(\pi)$ can have high variance due to stochasticity in the policy and environment, yielding slow learning (Williams, 1992; Glynn, 1990; Peters & Schaal, 2006). In settings with SREs, this problem is compounded by the variance due to θ, which the environment samples for each trajectory in the batch. Furthermore, the SREs may not be observed during learning since the environment samples $\theta \sim p(\theta)$, which can cause convergence to a highly suboptimal policy. Since these methods do not explicitly consider the environment variable's effect on learning, we call them *naïve* approaches.

2.2. Bayesian Optimisation

A *Gaussian process* (GP) (Rasmussen & Williams, 2005) is a distribution over functions. It is fully specified by its mean $m(\mathbf{x})$ (often assumed to be 0 for convenience) and covariance functions $k(\mathbf{x}, \mathbf{x}')$ which encode any prior belief about the function. A posterior distribution is generated by using observed values to update the belief about the function in a Bayesian way. The squared exponential kernel is a popular choice for the covariance function, and has the form $k(\mathbf{x}, \mathbf{x}') = \sigma_0^2 \exp[-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \Sigma^{-1} (\mathbf{x} - \mathbf{x}')]$, where Σ is a diagonal matrix whose diagonal gives lengthscales corresponding to each dimension of $\mathbf{x}$. By conditioning on the observed data, predictions for any new points can be computed analytically as a Gaussian $\mathcal{N}\big(f(\mathbf{x}); \mu(\mathbf{x}), \sigma^2(\mathbf{x})\big)$:

$$\mu(\mathbf{x}) = k(\mathbf{x}, \mathbf{X})(\mathbf{K} + \sigma^2_{\text{noise}}\mathbf{I})^{-1} f(\mathbf{X}) \tag{1a}$$

$$\sigma^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - k(\mathbf{x}, \mathbf{X})(\mathbf{K} + \sigma^2_{\text{noise}}\mathbf{I})^{-1} k(\mathbf{X}, \mathbf{x}), \tag{1b}$$

where $\mathbf{X}$ is the design matrix, $f(\mathbf{X})$ is the corresponding function values, and $\mathbf{K}$ is the covariance matrix with elements $k(\mathbf{x_i}, \mathbf{x_j})$. Probabilistic modelling of the predictions makes GPs well suited for optimising $f(\mathbf{x})$ using BO. Given a set of observations, the next point for evaluation is chosen as the $\mathbf{x}$ that maximises an *acquisition function*, which uses the posterior mean and variance to balance exploitation and exploration. The choice of acquisition function can significantly impact performance and numerous acquisition functions have been suggested in the literature, ranging from confidence or expectation based methods to entropy based methods. We consider two acquisition functions: *upper confidence bound* (UCB) (Cox & John, 1992; 1997), and *fast information-theoretic Bayesian optimisation* (FITBO) (Ru et al., 2017).

Given a dataset $\mathcal{D}_{1:n} = \{(\mathbf{x}_i, f(\mathbf{x}_i))\}_{i=1}^n$, UCB directly incorporates the prediction uncertainty by defining an upper bound: $\alpha_{\text{UCB}}(\mathbf{x} \mid \mathcal{D}_{1:n}) = \mu(f(\mathbf{x}) \mid \mathcal{D}_{1:n}) + \kappa\, \sigma(f(\mathbf{x}) \mid \mathcal{D}_{1:n})$, where κ controls the exploration-exploitation tradeoff. By contrast, FITBO aims to reduce the uncertainty about the global optimum $f(\mathbf{x}^*)$ by selecting the $\mathbf{x}$ that minimises the entropy of the distribution $p(f(\mathbf{x}^*) \mid \mathbf{x}, \mathcal{D}_{1:n})$:

$$\alpha_{\text{FITBO}}(\mathbf{x} \mid \mathcal{D}_{1:n}) = H[p(f(\mathbf{x}) \mid \mathbf{x}, \mathcal{D}_{1:n})] \\ - \mathbb{E}_{p(f(\mathbf{x}^*)|\mathcal{D}_{1:n})}\left[H[p(f(\mathbf{x}) \mid f(\mathbf{x}^*), \mathbf{x}, \mathcal{D}_{1:n})]\right],$$

which cannot be computed analytically, but can be approximated efficiently following Ru et al. (2017).

BO mainly minimises simple regret: The acquisition function suggests the next point $\mathbf{x}_i$ for evaluation at each timestep, but then the algorithm suggests what it believes to be the optimal point $\hat{\mathbf{x}}_i^*$, and the regret is defined as $\sum_{i=1}^N f(\mathbf{x}^*) - f(\hat{\mathbf{x}}_i^*)$. This is different from a bandit setting where the cumulative regret is defined as $\sum_{i=1}^N f(\mathbf{x}^*) - f(\mathbf{x}_i)$. Krause & Ong (2011) show that the UCB acquisition function is also a viable strategy to minimise cumulative regret in a contextual GP bandit setting, where selection of $\mathbf{x}_i$ conditions on some observed context.

3. Fingerprint Policy Optimisation

To address the challenges posed by environments with SREs, we introduce *fingerprint policy optimisation* (FPO). The main idea is to sample θ from a parametrised distribution $q_{\psi_n}(\theta)$ where the parameters ψ_n are conditioned on a fingerprint of the current policy, such that it helps the policy optimisation routine learn a policy that takes into account any SREs. Concretely, FPO executes the following steps at each iteration n. First, it selects ψ_n by approximately solving an optimisation problem defined below. Second, it samples trajectories from the environment using the current policy π_n, where each trajectory uses a value for θ sampled from a distribution parameterised by ψ_n: $\theta \sim q_{\psi_n}(\theta)$. Third, these trajectories are fed to a policy optimisation routine, e.g., a policy gradient algorithm, which uses them to compute an updated policy, $\pi_{n+1} = \textsc{PolOpt}(\psi, \pi_n)$. Fourth, new, independent trajectories are generated from the environment with π_{n+1} and used to estimate $J(\pi_{n+1})$. Fifth, a new point is added to a dataset $\mathcal{D}_{1:n} = \{\big((\psi_{i-1}, \pi_{i-1}), J(\pi_i)\big)\}_{i=1}^n$, which is input to a GP. The process then repeats, with BO using the GP to select the next ψ.

The key insight behind FPO is that at each iteration n, FPO should select the ψ_n that it expects will maximise the performance of the *next* policy, π_{n+1}:

$$\begin{aligned} \psi_n|\pi_n &= \underset{\psi}{\operatorname{argmax}}\, J(\pi_{n+1}) \\ &= \underset{\psi}{\operatorname{argmax}}\, J\big(\textsc{PolOpt}(\psi, \pi_n)\big). \end{aligned} \tag{2}$$

In other words, FPO chooses the ψ_n that it thinks will help $\textsc{PolOpt}$ maximise the improvement to π_n that can be made in a single policy update. By modelling the relationship between ψ_n, π_n, and $J(\pi_{n+1})$ with a GP, FPO can learn from experience how to select an appropriate ψ for the current π. Modelling $J(\pi_{n+1})$ directly also bypasses the issue of modelling π_{n+1} and its relationship to $J(\pi_{n+1})$, which is infeasible when π_{n+1} is high dimensional. Note that while $\textsc{PolOpt}$ has inputs (ψ, π_n), the optimisation is performed over ψ only, with π_n fixed. FPO is summarised in Algorithm 1 and illustrated in Figure 1. The remainder of this section describes in more detail elements of FPO that are essential to make it work in practice.

3.1. Selecting ψ_n

The optimisation problem in (2) is difficult for two reasons. First, solving it requires calling $\textsc{PolOpt}$, which is expensive in both computation and samples. Second, the observed $J(\pi)$ can be noisy due to the inherent stochasticity in the policy and the environment.

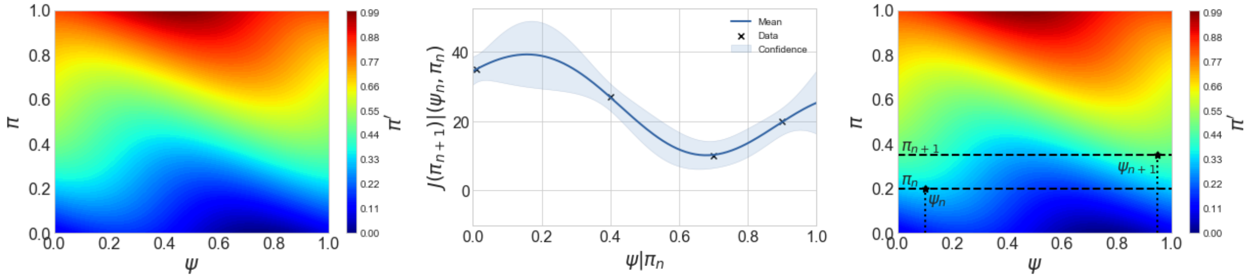
BO is particularly suited to such settings as it is sample efficient, gradient free, and can work with noisy observations. In this paper, we consider both the UCB and FITBO acquisition functions to select $\psi_n|\pi_n$ in (2), and compare their performance. Formally, we model the returns $J(\pi_n)$ as a GP with inputs (ψ_{n-1}, π_{n-1}). Given a dataset $\mathcal{D}_{1:n} = \{((\psi_{i-1}, \pi_{i-1}), J(\pi_i))\}_{i=1}^n$, ψ_n is selected by maximising the UCB or FITBO acquisition function:

$$\alpha_{\text{UCB}}(\psi_n|\pi_n) = \mu(J(\pi_{n+1})|\pi_n) + \kappa\sigma(J(\pi_{n+1})|\pi_n) \tag{3}$$

$$\alpha_{\text{FITBO}}(\psi_n|\pi_n) = H[p(J(\pi_{n+1})|\psi, \pi_n)] \\ - \mathbb{E}_{p(J(\pi_{n+1}^*)|\pi_n)}[H[p(J(\pi_{n+1})|\psi, \pi_n, J(\pi_{n+1}^*))]], \tag{4}$$

where we drop conditioning on $\mathcal{D}_{1:n}$ for ease of notation. See Figure 1b for an illustration.

Estimating the gradient using trajectories sampled from the environment with $\theta \sim q_\psi(\theta)$ introduces bias. While importance sampling methods (Frank et al., 2008; Ciosek &



(a)

(b)

(c)

Figure 1. (a) The policy optimisation routine PolOpt takes input (ψ, π) and updates the policy to π'; (b) FPO directly models $J(\pi')$ as a function of (ψ, π). At iteration n, π_n is fixed and FPO selects $\psi_n|\pi_n$ based on the UCB or FITBO acquisition function, (c) after which PolOpt(ψ_n, π_n) updates the policy to π_{n+1}, and this process repeats.

Algorithm 1 Fingerprint Policy Optimisation

```
input Initial policy $\pi_0$, original distribution $p(\theta)$, randomly
    initialised $q_{\psi_0}(\theta)$, policy optimisation method PolOpt,
    number of policy iterations $N$, dataset $\mathcal{D}_0 = \{\}$
for n = 1, 2, 3, . . . , N do
    Sample $\theta_{1:k}$ from $q_{\psi_{n-1}}(\theta)$, and with $\pi_{n-1}$ sample trajectories $\tau_{1:k}$ corresponding to each $\theta_{1:k}$
    Compute $\pi_n$ = PolOpt$(\tau_{1:k})$ = PolOpt$(\psi_{n-1}, \pi_{n-1})$
    Compute $J(\pi_n)$ using numerical quadrature as described in Section 3.2. Use the sampled trajectories to compute the policy fingerprint as described in Section 3.3.
    Set $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{((\psi_{n-1}, \pi_{n-1}), J(\pi_n))\}$ and update the GP to condition on $\mathcal{D}_n$
    Use either the UCB (3) or FITBO (4) acquisition functions to select $\psi_n$.
end for
```

Whiteson, 2017) could correct for this bias, FPO does not explicitly do so. Instead FPO lets BO implicitly optimise a bias-variance tradeoff by selecting ψ to maximise the one-step improvement objective.

3.2. Estimating $J(\pi_n)$

Estimating $J(\pi_n)$ accurately in the presence of SREs can be challenging. A Monte Carlo estimate using samples of trajectories from the original environment requires prohibitively many samples. One alternative would be to apply an IS correction to the trajectories generated from $q_{\psi_n}(\theta)$ for the policy optimisation routine. However, this is not feasible since it would require computing the IS weights $\frac{p(\tau|\theta)}{q_\psi(\tau|\theta)}$, which depend on the unknown transition function. Furthermore, even if the transition function is known, there is no reason why ψ_n should yield a good IS distribution since it is selected with the objective of maximising $J(\pi_{n+1})$.

Instead, FPO applies exhaustive summation for discrete θ and numerical quadrature for continuous θ to estimate $J(\pi_n)$. That is, if the support of θ is discrete, FPO simply samples a trajectory from each environment defined by θ and estimates $J(\pi_n) = \sum_{l=1}^{L} p(\theta = \theta_l) R(\theta_l, \pi_n)$. To reduce the variance due to stochasticity in the policy and the environment, we can sample multiple trajectories from each θ_l. For continuous θ, we apply an adaptive Gauss-Kronrod quadrature rule to estimate $J(\pi_n)$. While numerical quadrature may not scale to high dimensions, in practice θ is usually low dimensional, making this a practical design choice.

Since for discrete θ we evaluate $J(\pi_n)$ through exhaustive summation, it is natural to consider a variation of the naïve approach, wherein $\nabla J(\pi_n)$ is also evaluated in the same manner during training, i.e., $\nabla J(\pi_n) = \sum_{l=1}^{L} p(\theta = \theta_l) \nabla J(\theta_l, \pi_n)$. We call this the 'Enum' baseline since the gradient is estimated by enumerating over all possible values of θ. Our experiments in Section 5 show that this baseline is unable to match the performance of FPO.

3.3. Policy Fingerprints

True global optimisation is limited by the curse of dimensionality to low-dimensional inputs, and BO has had only rare successes in problems with more than twenty dimensions (Wang et al., 2013). In FPO, many of the inputs to the GP are policy parameters and in practice, the policy may be a neural network with thousands of parameters. While Rajeswaran et al. (2017b) show that linear and radial basis function policies can perform as well as neural networks in some simulated continuous control tasks, even these policies have hundreds of parameters, far too many for a GP.

Thus, we need to develop a policy *fingerprint*, i.e., a representation that is low dimensional enough to be treated as an input to the GP but expressive enough to distinguish the policy from others. Foerster et al. (2017) showed that a surprisingly simple fingerprint, consisting only of the training iteration, suffices to stabilise multi-agent Q-learning. Such a fingerprint is insufficient for FPO, as the GP fails to model the response surface and treats all observed $J(\pi)$ as noise. However, the principle still applies: a simplistic fingerprint that discards much information about the policy can still be sufficient for decision making, in this case to select ψ_n.

In this spirit, we propose two fingerprints. The first, the *state fingerprint*, augments the training iteration with an estimate of the stationary state distribution induced by the policy. In particular, we fit an anisotropic Gaussian to the set of states visited in the trajectories sampled while estimating $J(\pi)$ (see Section 3.2). The size of this fingerprint grows linearly with the dimensionality of the state space, instead of the number of parameters in the policy.

In many settings, the state space is high dimensional, but the action space is low dimensional. Therefore, our second fingerprint, the *action fingerprint*, is a Gaussian approximation of the marginal distribution over actions induced by the policy: $a(\pi) = \int \pi(a|s)s(\pi)ds$ (here $s(\pi)$ is the stationary state distribution induced by π), sampled from trajectories as with the state fingerprint.

Of course, neither the stationary state distribution nor the marginal action distribution are likely to be Gaussian and could in fact be multimodal. Furthermore, the state distribution is estimated from samples used to estimate $J(\pi)$, and not from $p(\theta)$. However, as our results show, these representations are nonetheless effective, as they do not need to accurately describe each policy, but instead just serve as low dimensional fingerprints on which FPO conditions.

3.4. Covariance Function

Our choice of policy fingerprints means that one of the inputs to the GP is a probability distribution. Thus for our GP prior we use a covariance that is the product of three terms, $k_1(\psi, \psi')$, $k_2(n, n')$ and $k_3\big(\text{FGP}(\pi), \text{FGP}(\pi')\big)$, where each of k_1, k_2, and k_3 is a squared exponential covariance function and $\text{FGP}(\pi)$ is the state or action fingerprint of π. Similar to Malkomes et al. (2016), we use the Hellinger distance to replace the Euclidean in k_3: this covariance remains positive-semi-definite as the Hellinger is effectively a modified Euclidean.

4. Related Work

Various methods have been proposed for learning in the presence of SREs. These are usually off environment and either based on learning a good IS distribution from which to sample the environment variable (Frank et al., 2008; Ciosek & Whiteson, 2017), or Bayesian active selection of the environment variable during learning (Paul et al., 2018).

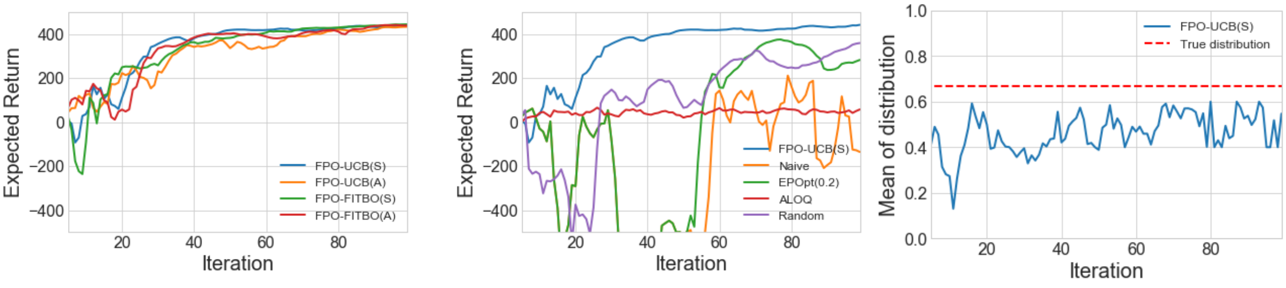
Frank et al. (2008) propose a temporal difference based method that uses IS to efficiently evaluate policies whose expected value may be substantially affected by rare events. However, their method assumes prior knowledge of the SREs, such that they can directly alter the probability of such events during policy evaluation. By contrast, FPO does not require any such prior knowledge about SREs, or the environment variable settings that might trigger them. It only assumes that the original distribution of the environment variable is known, and that the environment variable is controllable during learning.

OFFER (Ciosek & Whiteson, 2017) is a policy gradient method that uses observed trials to gradually change the IS distribution over the environment variable. Like FPO, it makes no prior assumptions about SREs. However, at each iteration it updates the environment distribution with the objective of minimising the variance of the gradient estimate, which may not lead to the distribution that optimises the learning of the policy. Furthermore, OFFER requires a full transition model of the environment to compute the IS weights. It can also lead to unstable IS estimates if the environment variable affects any transitions besides the initial state.

ALOQ (Paul et al., 2018) is a Bayesian optimisation and quadrature method that models the return as a GP with the policy parameters and environment variable as inputs. At each iteration it actively selects the policy and then the environment variable in an alternating fashion and, as such, performs the policy search in the parameter space. As a BO based method, it does not assume the observations are Markov and is highly sample efficient. However, it can only be applied to settings with low dimensional policies. Furthermore, its computational cost scales cubically with the number of iterations, and is thus limited to settings where a good policy can be found within relatively few iterations. By contrast, FPO uses policy gradients to perform policy optimisation while the BO component generates trajectories, which when used by the policy optimiser are expected to lead to a larger improvement in the policy.

In the wider BO literature, Williams et al. (2000) suggested a method for settings where the objective is to optimise expensive integrands. However, their method does not specifically consider the impact of SREs and, as shown by Paul et al. (2018), is unsuitable for such settings. Toscano-Palmerin & Frazier (2018) suggest BQO, another BO based method for expensive integrands. Their method also does not explicitly consider SREs. Finally, both these methods suffer from all the disadvantages of BO based methods mentioned earlier.

EPOpt(ϵ) (Rajeswaran et al., 2017a) is a different off-environment approach that learns robust policies by maximising the ϵ-percentile conditional value at risk (CVaR) of the policy. First, it randomly samples a set of simulator settings; then trajectories are sampled for each of these settings. A policy optimisation routine (e.g., TRPO (Schulman et al., 2015)) then updates the policy based on only



(a) Different versions of FPO

(b) FPO vs baselines

(c) FPO vs baselines

Figure 2. Results for the Cliff Walker environment. (a) Comparison of the performance of different versions of FPO. (b) Comparison of FPO-UCB(S) against the baselines. (c) The mean of the true distribution of the cliff location ('True distribution'), and the mean of the distribution selected by FPO-UCB(S) during learning (FPO-UCB(S)). Information on variation in performance due to the random starts is presented in Appendix A.2.

those trajectories with returns lower than the ϵ percentile in the batch. A fundamental difference to FPO is that it finds a risk-averse solution based on CVaR, while FPO finds a risk neutral policy. Also, while FPO actively changes the distribution for sampling the environment variable at each iteration, EPOpt samples them from the original distribution, and is thus unlikely to be suitable for settings with SREs, since it will not generate SREs often enough to to learn an appropriate response. Finally, EPOpt discards all sampled trajectories in the batch with returns greater than ϵ percentile for use by the policy optimisation routine, making it highly sample inefficient, especially for low values of ϵ.

RARL (Pinto et al., 2017) learns robust policies by training in a simulator where an adversary applies destabilising forces, with both the agent and the adversary trained simultaneously. RARL requires significant prior knowledge in setting up the adversary to ensure that the environment is easy enough to be learnable but difficult enough that the learned policy will be robust. Like EPOpt, it does not consider settings with SREs.

By learning an optimal distribution for the environment variable conditioned on the policy fingerprint, FPO also has some parallels with meta-learning. Methods like MAML (Finn et al., 2017), and Reptile (Nichol et al., 2018) seek to find a good policy representation that can be adapted quickly to a specified task. Andrychowicz et al. (2016); Chen et al. (2017) seek to optimise neural networks by learning an automatic update rule based on transferring knowledge from similar optimisation problems. To maximise the performance of a neural network across a set of discrete tasks, Graves et al. (2017) propose a method for automatically selecting a curriculum during learning. Their method treats the problem as a multi-armed bandit and uses Exp3 (Auer et al., 2002) to find the optimal curriculum. Unlike these methods, which seek to quickly adapt to a new task after training on some related task, FPO seeks to maximise the expected return across a family of tasks characterised by different settings of the environment variable.

5. Experiments

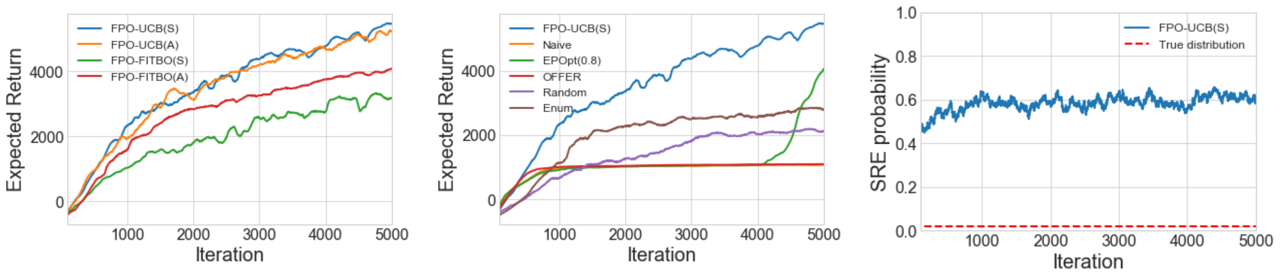
To evaluate the empirical performance of FPO, we start by applying it to a simple problem: a modified version of the cliff walker task (Sutton & Barto, 1998), with one dimensional state and action spaces. We then move on to simulated robotics problems based on the MuJoCo simulator (Brockman et al., 2016) with much higher dimensionalities. These were modified to include SREs. We aim to answer two questions: (1) How do the different versions of FPO (UCB vs. FITBO acquisition functions, state (S) vs. action (A) fingerprints) compare with each other? (2) How does FPO compare to existing methods (Naïve, Enum, OFFER, EPOpt, ALOQ), and ablated versions of FPO. We use TRPO as the policy optimisation method combined with neural net policies. We also include θ as an input to the baseline (but not to the policy) since it is observable during training.

We repeat all our experiments across 10 random starts. For ease of viewing we present only the median of the expected returns in the plots. Further experimental details are provided in Appendix A.1, and information about the variation due to the random starts is presented in Appendix A.2.

Due to the disadvantages of ALOQ mentioned in Section 4, we were able to apply it only on the cliff walker problem. The policy dimensionality and the total number of iterations for the simulated robotic tasks were far too high. Note also that, while we compare FPO to EPOpt, these methods optimise for different objectives.

5.1. Cliff Walker

We start with a modified version of the cliff walker problem where instead of a gridworld we consider an agent moving in a continuous state space $\mathbb{R}$; the agent starts randomly near the state 0, and at each timestep can take an action $a \in \mathbb{R}$. The environment then transitions the agent to a new location $s' = s + 0.025\text{sign}(a) + 0.005\epsilon$, where ϵ is standard Gaussian noise. The location of the cliff is given by $1 + \theta$, where $p(\theta) \sim \textit{Beta}(2, 1)$. If the agent's current

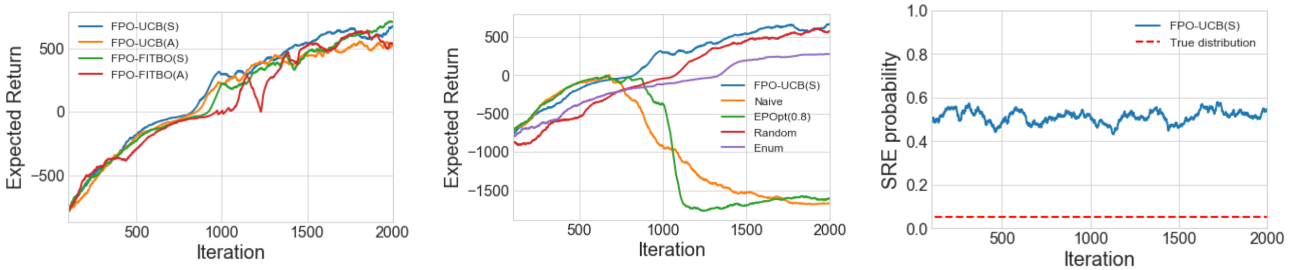


(a) Different versions of FPO

(b) FPO vs baselines

(c) Learnt ψ

Figure 3. Results for the Half Cheetah environment. (a) Comparison of the performance of different versions of FPO. (b) Comparison of FPO-UCB(S) against the baselines. (c) The actual probability of velocity target being 4 ('True distribution'), and the probability selected by FPO-UCB(S) during learning (FPO-UCB(S)). Information on variation in performance due to the random starts is presented in Appendix A.2.



(a) Different versions of FPO

(b) FPO vs baselines

(c) Learnt ψ

Figure 4. Results for the Ant environment. (a) Comparison of the performance of different versions of FPO. (b) Comparison of FPO-UCB(S) against the baselines. (c) The actual probability of velocity target being 4 ('True distribution'), and the probability selected by FPO-UCB(S) during learning (FPO-UCB(S)). Information on variation in performance due to the random starts is presented in Appendix A.2.

state is lower than the cliff location, it gets a reward equal to its state; otherwise it falls off the cliff and gets a reward of -5000, terminating the episode. Thus the objective is to learn to walk as close to the cliff edge as possible without falling over.

Figure 2a shows that all versions of FPO do equally well on the task. Figure 2b shows that FPO-UCB(S) learns a policy with a higher expected return than all other baselines. This is not surprising since, as discussed in Section 2.1, the gradient estimates without active selection of ψ are likely to have high variance due to the presence of SREs. For EPOpt we set $\epsilon = 0.2$ and perform rejection sampling after 50 iterations. The poor performance of ALOQ is expected since even in this simple problem, the policy dimensionality of 47 is high for BO. We could not run OFFER since an analytical solution of $\nabla_\psi q_\psi(\theta)$ does not exist.

Figure 2c compares the mean of sampling distribution for the cliff location chosen by FPO-UCB(S), i.e., $\mathbb{E}_{q_{\psi_n}}[\theta]$, against the mean of true distribution of the cliff location ($\mathbb{E}_p[\theta]$). FPO-UCB(S) selects sampling distributions for the cliff location that are more conservative than the true distribution. This is expected as falling off the cliff has a significant cost and closer cliff locations need to be sampled more frequently to learn a policy that avoids the low probability events where the cliff location is close under the true distribution.

The large negative reward for falling off the cliff acts like an SRE in this setting, so we can modify it to have no SRE by setting the reward to 0, but retain the randomised location of the cliff as an environment variable that affects the dynamics. In Appendix A.4 we show that the active selection of the environment variable enables FPO to outperform the naïve approach in this setting as well.

5.2. Half Cheetah

Next we consider simulated robotic locomotion tasks using the Mujoco simulator. In the original OpenAI Gym HalfCheetah task, the objective is to maximise forward velocity. We modify the original problem such that in 98% of the cases the objective is to achieve a target velocity of 2, with rewards decreasing linearly with the distance from the target. In the remaining 2%, the target velocity is set to 4, with a large bonus reward, which acts as an SRE.

Figure 3a shows that FPO with UCB outperforms FPO with FITBO. We suspect that this is because FITBO tends to over-explore. Also, as mentioned in Section 2.2, it was developed with the aim of minimising simple regret and it is not known how efficient it is at minimising cumulative regret. As in cliff walker, both the action and state fingerprints perform equally well with UCB.

Figure 3b shows that the Naïve method and OFFER converge to a locally optimal policy that completely ignores the SRE, while the random selection of ψ does slightly better. The Enum baseline performs better than random, but is still far worse than FPO. This shows that the key to learning a good policy is the active selection of ψ_n, which also includes the implicit bias-variance tradeoff performed by FPO. We set $\epsilon = 0.8$ for EPOpt, but in this case we use the trajectories with returns exceeding the threshold for the policy optimisation since the SRE has a large positive return. Although its performance increases after iteration 4,000, it is extremely sample inefficient, requiring about five times the samples of FPO. We did not run ALOQ as it is entirely infeasible given the policy dimensionality ($> 20,000$).

Finally, Figure 3c presents the schedule of ψ, (i.e., the probability of the target velocity being 4) as selected by FPO-UCB(S) across different iterations. FPO learns to vary ψ, starting with 0.5 initially, to hovering around 0.6 once the cheetah can consistently reach velocities greater than 2. Further assessments of the learnt policy is presented in Appendix A.3.

5.3. Ant

The ant environment is much more difficult than half cheetah since the agent moves in 3D, and the state space has 111 dimensions compared to 17 for half cheetah. The larger state space also makes learning difficult due to the higher variance in the gradient estimates. We modify the original problem such that velocities greater than 2 carry a 5% chance of damage to the ant. On incurring damage, which we treat as the SRE, the agent receives a large negative reward, and the episode terminates.

Figure 4a compares the performance of the UCB versions of FPO. There is no significant difference between the performance of the state or action fingerprint, or between the UCB and FITBO acquisition functions. Thus, the lower dimensional fingerprint (in this case the action fingerprint) can be chosen without compromising performance.

Figure 4b shows that for the Naïve method and EPOpt, performances drops significantly after about 750 iterations. This is because the learned policies generate velocities beyond 2, which after factoring in the effect of the SRE, yield much lower expected returns. Since the SREs are not seen often enough, these methods do not learn that higher velocities actually yield lower expected returns. The Enum baseline once again performs better than the naïve approach, but still worse than FPO. The random baseline performs better, and eventually matches the performance of FPO. We could not run OFFER in this setting as computing the IS weights would require knowledge of the transition model.

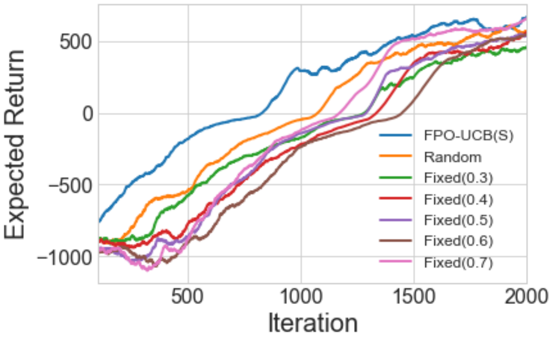


Figure 5. Comparison of FPO-UCB(S) against the Fixed baselines

Figure 4c shows that the optimal schedule for ψ, the probability of damage for velocities greater than 2, as learnt by FPO-UCB(S), hovers around 0.5. This helps explain the relatively good performance of the random baseline: since the baseline samples $\psi \sim U(0,1)$ at each iteration, the expected value of ψ is 0.5, and thus we can expect it to find a good policy eventually. Of course, active selection of ψ still matters, as evidenced by FPO outperforming it initially.

A natural question that arises based on the above result, is whether the schedule learnt by FPO-UCB(S) is better than a fixed schedule wherein the probability of breakage is fixed to some value throughout training. We investigate this by running another set of baselines wherein the probability of breakage is fixed to each of $\{0.3, 0, 4, 0.5, 0.6, 0.7\}$ throughout training. The learning curves are presented in Figure 5, with ‘Fixed(x)’ referring to the baseline with the probability of breakage fixed at x. While all of the baselines finally converge to a policy that performs on par with FPO-UCB(S), they learn much slower initially. This goes to show that learning a schedule that takes into account the current policy is key to faster learning. Further assessments of the learnt policy is presented in Appendix A.3.

6. Conclusions & Future Work

Environment variables can have a significant impact on the performance of a policy and are pervasive in real-world settings. This paper presented FPO, a method based on the insight that active selection of environment variables during learning can lead to policies that take into account the effect of SREs. We introduced novel state and action fingerprints that can be used by BO with a one-step improvement objective to make FPO scalable to high dimensional tasks irrespective of the policy dimensionality. We applied FPO to a number of continuous control tasks of varying difficulty and showed that FPO can efficiently learn policies that are robust to significant rare events, which are unlikely to be observable under random sampling but are key to learning good policies. In the future we would like to develop fingerprints for discrete state and action spaces, and explore using a multi-step improvement objective for the BO component.

Acknowledgements

We would like to thank Binxin Ru for sharing the code for FITBO, and Yarin Gal for the helpful discussions. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement #637713). The experiments were made possible by a generous equipment grant from NVIDIA.

References

Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In *Neural Information Processing Systems (NIPS)*, 2016.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Chen, Y., Hoffman, M. W., Colmenarejo, S. G., Denil, M., Lillicrap, T. P., Botvinick, M., and de Freitas, N. Learning to learn without gradient descent by gradient descent. In *International Conference on Machine Learning (ICML)*, 2017.

Ciosek, K. and Whiteson, S. Offer: Off-environment reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2017.

Cox, D. D. and John, S. A statistical method for global optimization. In *IEEE International Conference on Systems, Man and Cybernetics*, 1992.

Cox, D. D. and John, S. SDO: A statistical method for global optimization. In *in Multidisciplinary Design Optimization: State-of-the-Art*, pp. 315–329, 1997.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning (ICML)*, 2016.

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

Foerster, J., Nardelli, N., Farquhar, G., Torr, P., Kohli, P., and Whiteson, S. Stabilising experience replay for deep multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

Frank, J., Mannor, S., and Precup, D. Reinforcement learning in the presence of rare events. In *International Conference on Machine Learning (ICML)*, 2008.

Glynn, P. W. Likelihood ratio gradient estimation for stochastic systems. In *Communications of the ACM*, 1990.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. Automated curriculum learning for neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Kakade, S. A natural policy gradient. In *Neural Information Processing Systems (NIPS)*, 2001.

Krause, A. and Ong, C. S. Contextual gaussian process bandit optimization. In *Neural Information Processing Systems (NIPS)*, 2011.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

Malkomes, G., Schaff, C., and Garnett, R. Bayesian optimization for automated model selection. In *Neural Information Processing Systems (NIPS)*. 2016.

Mordatch, I., Lowrey, K., Andrew, G., Popovic, Z., and Todorov, E. V. Interactive control of diverse complex characters with neural networks. In *Neural Information Processing Systems (NIPS)*. 2015.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018.

Paul, S., Chatzilygeroudis, K., Ciosek, K., Mouret, J.-B., Osborne, M., and Whiteson, S. Alternating optimisation and quadrature for robust control. In *AAAI Conference on Artificial Intelligence*, 2018.

Peters, J. and Schaal, S. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

Rajeswaran, A., Ghotra, S., Levine, S., and Ravindran, B. EPOpt: Learning robust neural network policies using model ensembles. *International Conference on Learning Representations (ICLR)*, 2017a.

Rajeswaran, A., Lowrey, K., Todorov, E. V., and Kakade, S. M. Towards generalization and simplicity in continuous control. In *Neural Information Processing Systems (NIPS)*. 2017b.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

Ru, B., McLeod, M., Granziol, D., and Osborne, M. A. Fast Information-theoretic Bayesian Optimisation. In *International Conference on Machine Learning (ICML)*, 2017.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

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

Toscano-Palmerin, S. and Frazier, P. I. Bayesian Optimization with Expensive Integrands. *ArXiv e-prints*, March 2018.

Wang, Z., Zoghi, M., Hutter, F., Matheson, D., and De Freitas, N. Bayesian Optimization in High Dimensions via Random Embeddings. In *IJCAI*, pp. 1778–1784, 2013.

Williams, B. J., Santner, T. J., and Notz, W. I. Sequential design of computer experiments to minimize integrated response functions. *Statistica Sinica*, 2000.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

A. Appendices

A.1. Hyperparameter Settings

Our implementation is based on rllab (Duan et al., 2016) and as such most of the hyperparameter settings were kept to their default settings. The details are provided in Table 1.

Table 1. Experimental details

	Cliff Walker	HalfCheetah	Ant
PolOpt	TRPO	TRPO	TRPO
KL constraint	0.01	0.01	0.01
Discount rate	0.99	0.99	0.99
GAE λ	1.0	1.0	1.0
Batch size	10,000	12,500	12,500
Policy layers	(5,5)	(100,100)	(100,100)
Policy units	Tanh	ReLU	ReLU
FPO-UCB κ	2	2	2

A.2. Detailed Experimental Results

In Table 2c we present the quartiles of the expected return of the final learnt policy for each method across the 10 random starts.

Table 2. Quartiles of expected return across 10 random starts

(a) Cliff Walker

	Q1	Median	Q2
FPO-UCB(S)	427.1	441.5	450.0
FPO-UCB(A)	335.2	432.6	440.4
FPO-FITBO(S)	428.1	443.6	453.1
FPO-FITBO(A)	372.2	438.2	451.5
Naïve	-1478.7	-135.5	243
EPOpt	-44.4	282.1	354.4
ALOQ	33.5	57.2	77.2
Random	345.8	358.9	373.4

(b) HalfCheetah

	Q1	Median	Q2
FPO-UCB(S)	3913.7	5464.0	5905.5
FPO-UCB(A)	4435.6	5231.8	5897.6
FPO-FITBO(S)	2973.9	3187.2	3923.7
FPO-FITBO(A)	3686.2	4091.1	7247.3
Naïve	1059.9	1071.1	1086.0
EPOpt	803.6	4066.0	4421.0
OFFER	1093.9	1097.4	1111.2
Random	1722.3	2132.6	2645.5
Enum	2442.8	2796.0	3428.4

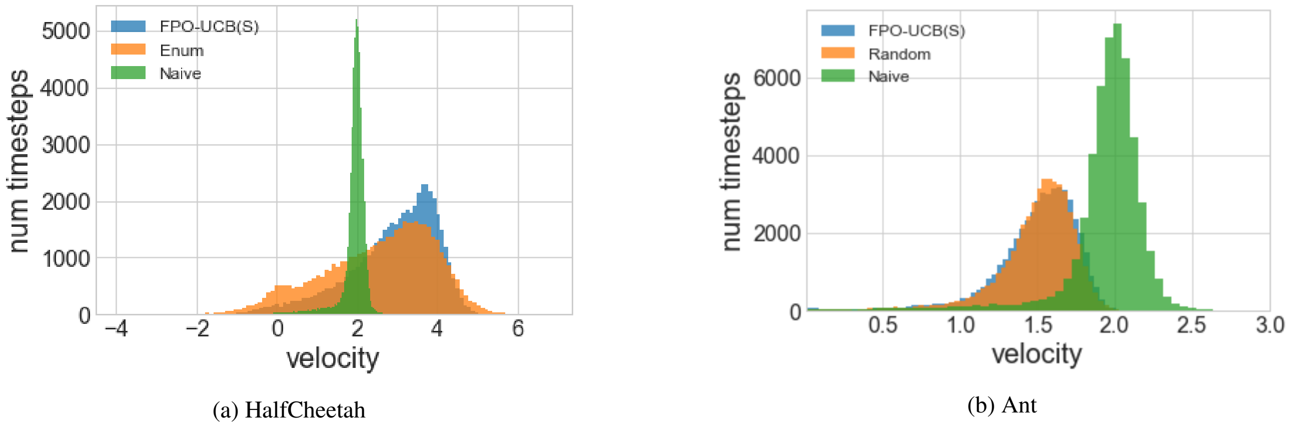
(c) Ant

	Q1	Median	Q2
FPO-UCB(S)	490.6	674.2	713.3
FPO-UCB(A)	408.1	519.4	629.7
FPO-FITBO(S)	626.8	704.2	770.0
FPO-FITBO(A)	455.7	533.1	707.0
Naïve	-1746.9	-1669.4	-1585.2
EPOpt	-1732.3	-1606.6	-1454.5
Random	460.3	575.6	640.4
Enum	255.7	273.4	285.6

A.3. Further Examination of the Learnt Policies

As explained in the Experiments section, the SREs for the HalfCheetah and Ant experiments are based on the velocities achieved by the agent; for HalfCheetah the SRE is defined as the velocity target being 4 (and carrying a large bonus reward) instead of 2 with a 2% probability of occurrence, while for Ant velocities greater than 2 has a 5% probability of incurring a large cost. Here we compare the performance of FPO-UCB(S) against the next best baseline (Enum for HalfCheetah and Random for Ant) and the Naïve baseline by visualising the velocity profiles of the final learnt policy. For each random start for each method we sampled 10 trajectories (for a total of 100 trajectories per method) and plot the histogram of the velocity at each timestep. This is presented in Figure 6.

For the HalfCheetah task, from Figure 6a we can see that the velocity profile for the Naïve approach is highly concentrated around 2. This goes to show that the Naïve approach learns a policy that does not take into account the SRE at all. On the other hand, both Enum and FPO-UCB(S) have velocity profiles with much higher variance with a lot of mass spread between 2 and 4. This goes to show that both of them take into account the effect of SREs. However, FPO-UCB(S) manages to better balance the SRE/non-SRE rewards and has slightly higher mass concentrated on 4,



(a) HalfCheetah

(b) Ant

Figure 6. Histogram of the velocity profile of the final learnt policies for each method.

which in turn leads to it significantly outperforming Enum.

For the Ant task, unsurprisingly once again the Naïve approach completely ignores the SREs, and exhibits a velocity profile that is greater than 2 roughly 50% of the time. The velocity profiles of the Random baseline and FPO-UCB(S) are almost exactly the same. This is unexpected as there is no significant difference between the expected return of the final policies learnt by these two methods, as shown in Section 5. As noted earlier, the good performance of Random is not unsurprising since the schedule for ψ chosen by FPO-UCB(S) is close to 0.5, which is also the mean of ψ under the random baseline as $\psi \sim U(0, 1)$.

A.4. Performance in settings without SREs

FPO considers the setting where environments are characterised by SREs. A natural question to ask is how does its performance compare to the naïve method in settings where there are no SREs. To investigate this we applied FPO-UCB(S), and the naïve baseline, to the cliff walker problem presented in Section 5.1, with the modification that falling off the cliff now carries 0 reward instead of -5000. This removes the SRE, but the environment variable (the location of the cliff) is still relevant since it has a significant effect on the dynamics. The results are presented in Figure 7. Note that the performance of the Naïve method is far more stable than in the setting with SRE. However, while it is able to learn a good policy, FPO-UCB(S) still performs better since it takes into account the effect of the environment variable.

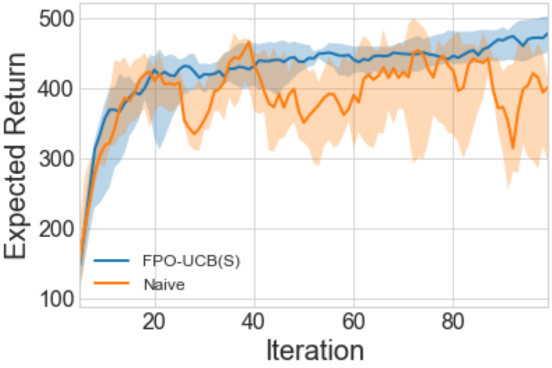


Figure 7. Results for the Cliff Walker environment without any SRE. Solid line shows the median and the shaded region the quartiles across 10 random starts.